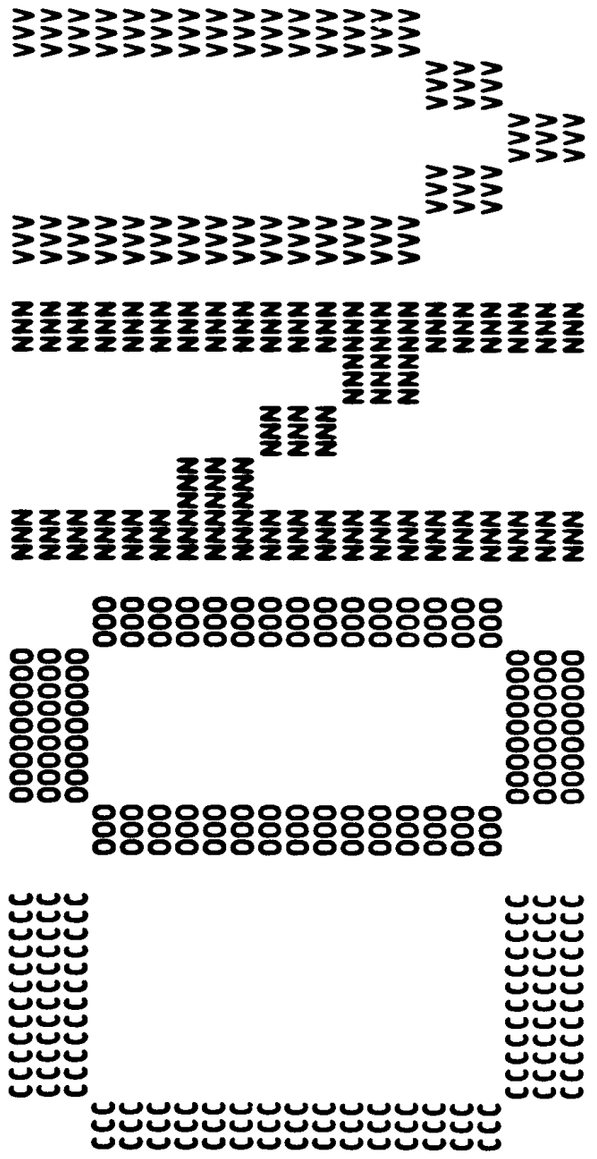
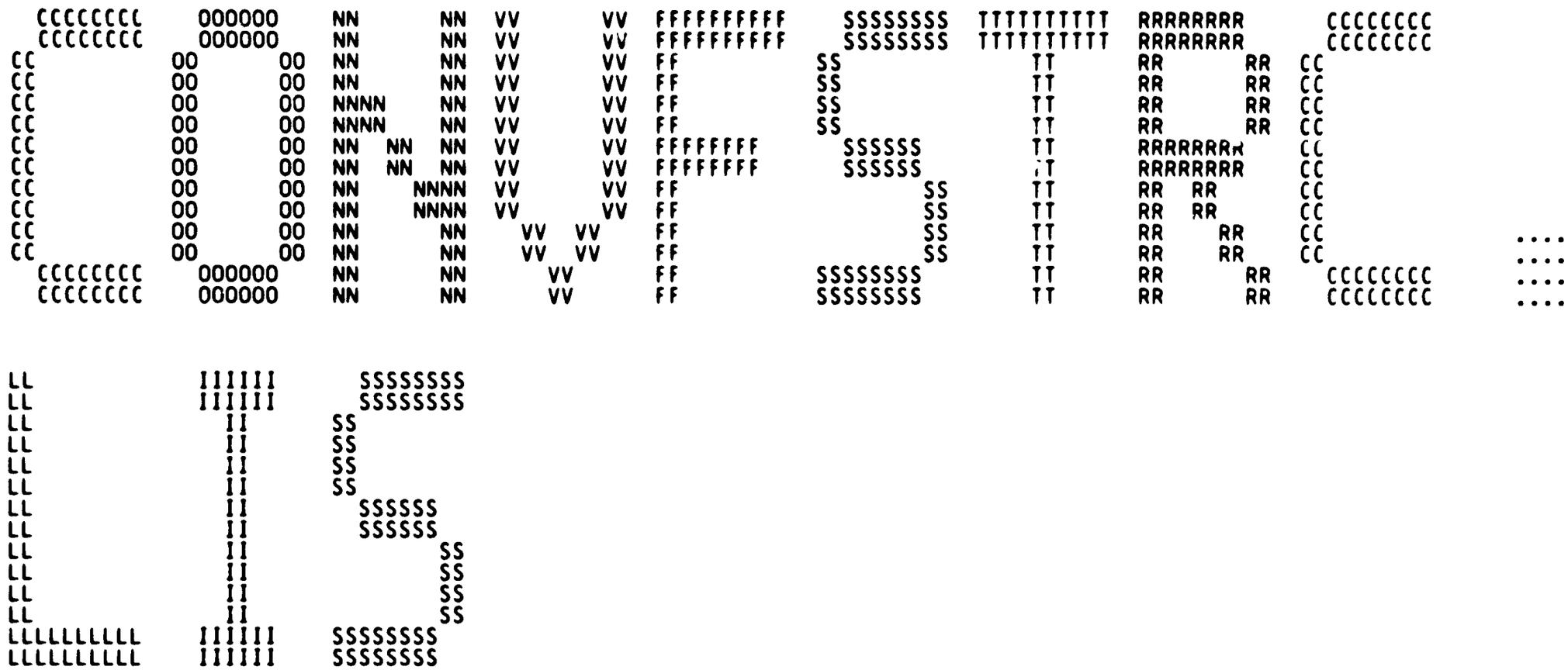


FILEID**CONVFSTRC



```
;  1          0001  0 %TITLE  'VAX-11 CONVERT'
;  2          0002  0 MODULE  CONV$FSTRC      ( IDENT='V04-000'
;  3          0003  0                          ,OPTLEVEL=3
;  4          0004  0                         ) =
;  5          0005  0
;  6          0006  1 BEGIN
;  7          0007  1
;  8          0008  1 !**************************************************************************
;  9          0009  1 !*                                                                        *
; 10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                               *
; 11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
; 12          0012  1 !*  ALL RIGHTS RESERVED.                                                  *
; 13          0013  1 !*                                                                        *
; 14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
; 15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
; 17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
; 18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
; 19          0019  1 !*  TRANSFERRED.                                                           *
; 20          0020  1 !*                                                                        *
; 21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
; 22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
; 23          0023  1 !*  CORPORATION.                                                           *
; 24          0024  1 !*                                                                        *
; 25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
; 26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
; 27          0027  1 !*                                                                        *
; 28          0028  1 !*                                                                        *
; 29          0029  1 !**************************************************************************
```

```
;    31          0030  1 !++
;    32          0031  1 !
;    33          0032  1 ! Facility:     VAX-11 CONVERT
;    34          0033  1 !
;    35          0034  1 ! Abstract:     CONVERT index record processing routines
;    36          0035  1 !
;    37          0036  1 ! Contents:
;    38          0037  1 !                CHECK_S_DUP
;    39          0038  1 !                CHECK_NULL
;    40          0039  1 !                SPLIT_DATA
;    41          0040  1 !                IN_SEGMENT
;    42          0041  1 !                COMPRESS_KEY
;    43          0042  1 !                COMPRESS_INDEX
;    44          0043  1 !                MAKE_INDEX
;    45          0044  1 !                WRITE_VBN
;    46          0045  1 !                COPY_REY
;    47          0046  1 !                CREATE_HIGH_KEY
;    48          0047  1 !
;    49          0048  1 ! Environment:
;    50          0049  1 !
;    51          0050  1 !                VAX/VMS Operating System
;    52          0051  1 !
;    53          0052  1 !
;    54          0053  1 ! Author:       Keith B Thompson        Creation date:          August-1980
;    55          0054  1 !
;    56          0055  1 !
;    57          0056  1 ! Modified by:
;    58          0057  1 !
;    59          0058  1 !       V03-011 JWT0144         Jim Teague              17-Nov-1983
;    60          0059  1 !               Fix compare-packed for null keys.
;    61          0060  1 !
;    62          0061  1 !       V03-010 KBT0553         Keith B. Thompson       25-Aug-1983
;    63          0062  1 !               Packed decimal high key fix
;    64          0063  1 !
;    65          0064  1 !       V03-009 KBT0480         Keith B. Thompson       31-Jan-1983
;    66          0065  1 !               Change rec_buf_ptr to rfa_buffer
;    67          0066  1 !
;    68          0067  1 !       V03-008 KBT0440         Keith B. Thompson       16-Dec-1982
;    69          0068  1 !               Add quadword key support
;    70          0069  1 !
;    71          0070  1 !       V03-007 KBT0403         Keith B. Thompson       19-Nov-1982
;    72          0071  1 !               Fix one byte bug in compress_key, change make_index and
;    73          0072  1 !               check_s_dup
;    74          0073  1 !
;    75          0074  1 !       V03-006 KBT0383         Keith B. Thompson       26-Oct-1982
;    76          0075  1 !               Add support for prologue 3 sidrs
;    77          0076  1 !
;    78          0077  1 !       V03-005 KBT0378         Keith B. Thompson       20-Oct-1982
;    79          0078  1 !               Check for keys out of order in split_data
;    80          0079  1 !
;    81          0080  1 !       V03-004 KBT0351         Keith B. Thompson       4-Oct-1982
;    82          0081  1 !               Use new linkage definitions
;    83          0082  1 !
;    84          0083  1 !       V03-003 KBT0038         Keith Thompson          3-Apr-1982
;    85          0084  1 !               Correct the vbn_size correction
;    86          0085  1 !
;    87          0086  1 !       V03-002 KBT0023         Keith Thompson          24-Mar-1982
```

```
;   88          0087  1 !                   Correct calculation of vbn_size in write_vbn
;   89          0088  1 !
;   90          0089  1 !       V03-001 KBT0013         Keith Thompson          16-Mar-1982
;   91          0090  1 !                   Fix the call to compress_data
;   92          0091  1 !****
```

```
;    94          0092  1
;    95          0093  1 PSECT
;    96          0094  1         OWN     = _CONV$FAST_D  (PIC),
;    97          0095  1         GLOBAL  = _CONV$FAST_D  (PIC),
;    98          0096  1         PLIT    = _CONV$PLIT    (SHARE,PIC),
;    99          0097  1         CODE    = _CONV$FAST_S  (SHARE,PIC);
;   100          0098  1
;   101          0099  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;   102          0100  1 LIBRARY 'SRC$:CONVERT';
;   103          0101  1
;   104          0102  1 DEFINE_ERROR_CODES;
;   105          0103  1
;   106          0104  1 LINKAGE
;   107          0105  1
;   108          0106  1         CL$COMPARE_KEYS = JSB : GLOBAL( CTX = 10,
;   109          0107  1                                         KEY_DESC = 11 ),
;   110          0108  1
;   111          0109  1         CL$IN_SEGMENT   = JSB : GLOBAL( BYTE_OFS = 6,
;   112          0110  1                                         NEXT_SEG = 7,
;   113          0111  1                                         NEXT_LEN = 8,
;   114          0112  1                                         KEY_DESC = 11 )
;   115          0113  1                                         NOTUSED( 3 );
;   116          0114  1
;   117          0115  1 EXTERNAL ROUTINE
;   118          0116  1         CONV$$CVT_TO_ASC        : CL$CVT_TO_ASC,
;   119          0117  1         CONV$$COMPRESS_DATA     : CL$JSB_LINK;
;   120          0118  1
;   121          0119  1 FORWARD ROUTINE
;   122          0120  1         COMPARE_KEYS    : CL$COMPARE_KEYS,
;   123          0121  1         IN_SEGMENT      : CL$IN_SEGMENT;
;   124          0122  1
;   125          0123  1 EXTERNAL
;   126          0124  1         CONV$GW_OUT_REC_SIZ     : SIGNED WORD,          ! Output Rec. Size
;   127          0125  1
;   128          0126  1         CONV$GL_RFA_BUFFER,
;   129          0127  1
;   130          0128  1         CONV$AB_OUT_FAB         : $FAB_DECL,
;   131          0129  1
;   132          0130  1         CONV$GB_PROL_V3         : BYTE,
;   133          0131  1
;   134          0132  1 !       Data Decl. for Fast Load routines
;   135          0133  1 !
;   136          0134  1         CONV$GL_RECORD_PTR,
;   137          0135  1         CONV$GL_DUP_BUF;
;   138          0136  1
;   139          0137  1 MACRO
;   140          0138  1 !       Some needed macros to define the data record for a bucket
;   141          0139  1 !
;   142          0140  1         IRC$L_RRV_VBN   = 3,0,32,0%,    ! RRV VBN Pointer
;   143          0141  1         IRC$L_RRV_VBN3  = 5,0,32,0%,    ! RRV VBN Pointer (Prologue 3)
;   144          0142  1         IRC$W_VAR_SIZ   = 7,0,16,0%,    ! Var. Rec. Format Size field
;   145          0143  1         IRC$L_DUPCOUNT  = 2,0,32,0%,    ! Duplicate count field
;   146          0144  1         IRC$W_DUPSZ     = 6,0,16,0%,    ! Size field when dup. are allowed
;   147          0145  1         IRC$W_NODUPSZ   = 2,0,16,0%;    ! Size field when dup. are not allowed
```

```
;  149          0146  1 %SBTTL  'CHECK_S_DUP'
;  150          0147  1 GLOBAL ROUTINE  CONV$$CHECK_S_DUP : CL$JSB_REG_9 =
;  151          0148  1 !++
;  152          0149  1 !
;  153          0150  1 ! Functional Description:
;  154          0151  1 !
;  155          0152  1 !       Checks if the current secondary key is a duplicate of the last key
;  156          0153  1 !
;  157          0154  1 ! Calling Sequence:
;  158          0155  1 !
;  159          0156  1 !       CONV$$CHECK_S_DUP()
;  160          0157  1 !
;  161          0158  1 ! Input Parameters:
;  162          0159  1 !       none
;  163          0160  1 !
;  164          0161  1 ! Implicit Inputs:
;  165          0162  1 !       none
;  166          0163  1 !
;  167          0164  1 ! Output Parameters:
;  168          0165  1 !       none
;  169          0166  1 !
;  170          0167  1 ! Implicit Outputs:
;  171          0168  1 !       none
;  172          0169  1 !
;  173          0170  1 ! Routine Value:
;  174          0171  1 !
;  175          0172  1 !       1 - Key was a duplicate
;  176          0173  1 !       0 - Key was NOT a duplicate
;  177          0174  1 !
;  178          0175  1 ! Routines Called:
;  179          0176  1 !
;  180          0177  1 !       CONV$$CVT_TO_ASC
;  181          0178  1 !
;  182          0179  1 ! Side Effects:
;  183          0180  1 !       none
;  184          0181  1 !
;  185          0182  1 !--
;  186          0183  1
;  187          0184  2     BEGIN
;  188          0185  2
;  189          0186  2     DEFINE_CTX;
;  190          0187  2     DEFINE_KEY_DESC;
;  191          0188  2
;  192          0189  2     ! If this is the very first record then just return 0
;  193          0190  2     ! else compaire the record with the last one
;  194          0191  2     !
;  195          0192  2     IF .CTX [ CTX$V_FST ]
;  196          0193  2     THEN
;  197          0194  2         RETURN 0
;  198          0195  2     ELSE
;  199          0196  2         RETURN CH$EQL ( .KEY_DESC [ KEY$B_KEYSZ ],              ! Key size
;  200          0197  2                         .CONV$GL_DUP_BUF,                       ! Old key
;  201          0198  2                         .KEY_DESC [ KEY$B_KEYSZ ],
;  202          0199  2                         .CONV$GL_RFA_BUFFER + 6 )               ! New key
;  203          0200  2
;  204          0201  1     END;
```

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

                                                .EXTRN  CONVERT$_FACILITY
                                                .EXTRN  CONV$_FAB_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_PROC_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

```
;  206          0202  1 %SBTTL  'CHECK_NULL'
;  207          0203  1 GLOBAL ROUTINE  CONV$$CHECK_NULL : CL$JSB_REG_9 =
;  208          0204  1 !++
;  209          0205  1 !
;  210          0206  1 ! Functional Description:
;  211          0207  1 !
;  212          0208  1 !       Checks is the current secondary key is a null key value
;  213          0209  1 !
;  214          0210  1 ! Calling Sequence:
;  215          0211  1 !
;  216          0212  1 !       CONV$$CHECK_NULL()
;  217          0213  1 !
;  218          0214  1 ! Input Parameters:
;  219          0215  1 !       none
;  220          0216  1 !
;  221          0217  1 ! Implicit Inputs:
;  222          0218  1 !       none
;  223          0219  1 !
;  224          0220  1 ! Output Parameters:
;  225          0221  1 !       none
;  226          0222  1 !
;  227          0223  1 ! Implicit Outputs:
;  228          0224  1 !       none
;  229          0225  1 !
;  230          0226  1 ! Routine Value:
;  231          0227  1 !
;  232          0228  1 !       1 - Key was a null value
;  233          0229  1 !       0 - Key was NOT a null value
;  234          0230  1 !
;  235          0231  1 ! Routines Called:
;  236          0232  1 !       none
;  237          0233  1 !
;  238          0234  1 ! Side Effects:
;  239          0235  1 !       none
;  240          0236  1 !
;  241          0237  1 !--
;  242          0238  1
;  243          0239  2     BEGIN
;  244          0240  2
;  245          0241  2     DEFINE_CTX;
;  246          0242  2     DEFINE_BUCKET;
;  247          0243  2     DEFINE_KEY_DESC;
;  248          0244  2
;  249          0245  2     BUILTIN
;  250          0246  2         CMPP;
;  251          0247  2
;  252          0248  2     BIND
;  253          0249  2         SEG_SIZE = .KEY_DESC + 44 : VECTOR [ ,BYTE ];
;  254          0250  2
;  255          0251  2     LOCAL
;  256          0252  2         KEYPTR  : REF BLOCK [ ,BYTE ],
;  257          0253  2         NUL;
;  258          0254  2
;  259          0255  2     ! The new key is 6 bytes into the input record
;  260          0256  2     !
;  261          0257  2     KEYPTR = .CONV$GL_RFA_BUFFER + 6;
;  262          0258  2
```

```
;  263          0259  2     ! Check the record one segment at a time
;  264          0260  2     !
;  265          0261  2     INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
;  266          0262  2     DO
;  267          0263  2
;  268          0264  2         ! If we dont have a null then dont bother looking any further
;  269          0265  2         !
;  270          0266  4         IF ( NUL = ( CASE .KEY_DESC [ KEY$B_DATATYPE ] FROM KEY$C_STRING TO KEY$C_MAX_DATA OF
;  271          0267  4                 SET
;  272          0268  4                 [ KEY$C_STRING ]  : CH$EQL( 1,KEY_DESC [ KEY$B_NULLCHAR ],
;  273          0269  4                                            .SEG_SIZE [ .I ],
;  274          0270  4                                            .KEYPTR,
;  275          0271  4                                            .KEY_DESC [ KEY$B_NULLCHAR ] );
;  276          0272  4
;  277          0273  4                 [ KEY$C_SGNWORD,KEY$C_UNSGNWORD ] : .KEYPTR [ 0,WORD_U ] EQLU 0;
;  278          0274  4
;  279          0275  4                 [ KEY$C_SGNLONG,KEY$C_UNSGNLONG ] : .KEYPTR [ 0,LONG_U ] EQLU 0;
;  280          0276  4
;  281          0277  4                 [ KEY$C_PACKED ] :
;  282          0278  4
;  283          0279  5                                 BEGIN
;  284          0280  5
;  285          0281  5                                 STACKLOCAL NULL;
;  286          0282  5
;  287          0283  5                                 NULL = 12;
;  288          0284  5
;  289          0285  5                                 ! The value returned by CMPP is backwards
;  290          0286  5                                 !
;  291          0287  5                                 IF CMPP( %REF(0), NULL,
;  292          0288  5                                          %REF (( .SEG_SIZE [ .I ] * 2 ) - 1),
;  293          0289  5                                          .KEYPTR ) EQLU 0
;  294          0290  5                                 THEN
;  295          0291  5                                     _SET
;  296          0292  5                                 ELSE
;  297          0293  5                                     _CLEAR
;  298          0294  5                                 END;
;  299          0295  4
;  300          0296  4
;  301          0297  4                 [ KEY$C_SGNQUAD,KEY$C_UNSGNQUAD ] :
;  302          0298  4                                     ( .KEYPTR [ 0,LONG_U ] EQLU 0 ) AND
;  303          0299  4                                     ( .KEYPTR [ 4,LONG_U ] EQLU 0 );
;  304          0300  4
;  305          0301  3                 TES ) )
;  306          0302  2             THEN
;  307          0303  2                 KEYPTR = .KEYPTR + .SEG_SIZE [ .I ]
;  308          0304  2             ELSE
;  309          0305  2                 EXITLOOP;
;  310          0306  2
;  311          0307  2     RETURN .NUL
;  312          0308  2
;  313          0309  2
;  314          0310  1     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, KEY_PTR                             ;
                    02  54            58  F2 000A2 15$:     AOBLSS  R8, I, 16$                              ; 0266
                                      C3  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
```

```
;  316          0311  1 %SBTTL  'SPLIT_DATA'
;  317          0312  1 GLOBAL ROUTINE  CONV$$SPLIT_DATA : CL$JSB_REG_9 =
;  318          0313  1 !++
;  319          0314  1 !
;  320          0315  1 ! Functional Description:
;  321          0316  1 !
;  322          0317  1 !       Copies the data record returned by GET_RECORD into the primary
;  323          0318  1 !       key data buffer, for prologue 3 files it also extracts the key
;  324          0319  1 !       from the data and places it into the record control buffer
;  325          0320  1 !       Also checks to see if the present record is a duplicate of the
;  326          0321  1 !       last record
;  327          0322  1 !
;  328          0323  1 ! Calling Sequence:
;  329          0324  1 !
;  330          0325  1 !       CONV$$SPLIT_DATA()
;  331          0326  1 !
;  332          0327  1 ! Input Parameters:
;  333          0328  1 !       none
;  334          0329  1 !
;  335          0330  1 ! Implicit Inputs:
;  336          0331  1 !       none
;  337          0332  1 !
;  338          0333  1 ! Output Parameters:
;  339          0334  1 !       none
;  340          0335  1 !
;  341          0336  1 ! Implicit Outputs:
;  342          0337  1 !       none
;  343          0338  1 !
;  344          0339  1 ! Routine Value:
;  345          0340  1 !
;  346          0341  1 !        1 - Key was a duplicate
;  347          0342  1 !        0 - Key was NOT a duplicate
;  348          0343  1 !       -1 - Key was out of order
;  349          0344  1 !
;  350          0345  1 ! Routines Called:
;  351          0346  1 !
;  352          0347  1 !       CONV$$CVT_TO_ASC
;  353          0348  1 !       IN_SEGMENT
;  354          0349  1 !       CONV$$COMPRESS_DATA
;  355          0350  1 !
;  356          0351  1 ! Side Effects:
;  357          0352  1 !       none
;  358          0353  1 !
;  359          0354  1 !--
;  360          0355  1
;  361          0356  2     BEGIN
;  362          0357  2
;  363          0358  2     DEFINE_CTX;
;  364          0359  2     DEFINE_BUCKET;
;  365          0360  2     DEFINE_KEY_DESC;
;  366          0361  2
;  367          0362  2     BIND
;  368          0363  2         SEG_SIZE        = .KEY_DESC + 44 : VECTOR [ ,BYTE ],
;  369          0364  2         SEG_POSI        = .KEY_DESC + 28 : VECTOR [ ,WORD ];
;  370          0365  2
;  371          0366  2     LOCAL
;  372          0367  2         DUP     : SIGNED;
```

```
;  373     0368  2
;  374     0369  2     ! For prologue 3 files we must extract the key and
;  375     0370  2     ! data from the input record
;  376     0371  2     !
;  377     0372  2     IF .CONV$GB_PROL_V3
;  378     0373  2     THEN
;  379     0374  3         BEGIN           ! Prologue 3 processing
;  380     0375  3
;  381     0376  4         BEGIN           ! Extract the key
;  382     0377  4
;  383     0378  4         LOCAL
;  384     0379  4             OFFSET,
;  385     0380  4             REC_PTR;            ! Pointer TO the data buffer
;  386     0381  4
;  387     0382  4         ! Find out where the extracted key will go...
;  388     0383  4         ! If we are doing key compression the start the control buffer + 13
;  389     0384  4         ! else it's only 11
;  390     0385  4         !
;  391     0386  4         IF .KEY_DESC [ KEY$V_KEY_COMPR ]
;  392     0387  4         THEN
;  393     0388  4             OFFSET = 13
;  394     0389  4         ELSE
;  395     0390  4
;  396     0391  4             ! If there is no record compression and the file is fixed then
;  397     0392  4             ! there is no size field
;  398     0393  4             !
;  399     0394  4             IF .KEY_DESC [ KEY$V_REC_COMPR ] OR
;  400     0395  5                ( .CONV$AB_OUT_FAB [ FAB$B_RFM ] EQL FAB$C_VAR )
;  401     0396  4             THEN
;  402     0397  4                 OFFSET = 11
;  403     0398  4             ELSE
;  404     0399  4                 OFFSET = 9;
;  405     0400  4
;  406     0401  4         ! Key goes here
;  407     0402  4         !
;  408     0403  4         REC_PTR = .CTX [ CTX$L_RCP ] + .OFFSET;
;  409     0404  4
;  410     0405  4         ! Move the key to the record buffer where it is suppose to be
;  411     0406  4         !
;  412     0407  4         INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
;  413     0408  4         DO
;  414     0409  5             BEGIN
;  415     0410  5
;  416     0411  5             ! Move it
;  417     0412  5             !
;  418     0413  5             CH$MOVE( .SEG_SIZE [ .I ],                          ! Segment size
;  419     0414  5                      .CONV$GL_RECORD_PTR + .SEG_POSI [ .I ],    ! Source
;  420     0415  5                      .REC_PTR );                                ! Destination
;  421     0416  5
;  422     0417  5             ! Increment destination
;  423     0418  5             !
;  424     0419  5             REC_PTR = .REC_PTR + .SEG_SIZE [ .I ]
;  425     0420  5
;  426     0421  4             END;
;  427     0422  4
;  428     0423  4         ! The size of the control record is the key size plus the overhead
;  429     0424  4         !
```

```
;  430          0425  4          CTX [ CTX$W_RCS ] = .KEY_DESC [ KEY$B_KEYSZ ] + .OFFSET;
;  431          0426  4
;  432          0427  4          ! Check for a duplicate with the last key
;  433          0428  4          !
;  434          0429  4          DUP = COMPARE_KEYS( .KEY_DESC [ KEY$B_KEYSZ ],          ! Key size
;  435          0430  4                              .CTX [ CTX$L_RCP ] + .OFFSET,       ! New key
;  436          0431  4                              .CTX [ CTX$L_LKP ] )                ! Last key
;  437          0432  4
;  438          0433  3          END;               ! Extract the key
;  439          0434  3
;  440          0435  4          BEGIN              ! Process the data
;  441          0436  4
;  442          0437  4          LOCAL
;  443          0438  4              BUFFER_PTR;
;  444          0439  4
;  445          0440  5          BEGIN              ! Extract the data record
;  446          0441  5
;  447          0442  5          GLOBAL REGISTER
;  448          0443  5              BYTE_OFS = 6,      ! Offset into the data record
;  449          0444  5              NEXT_SEG = 7,      ! No. of char. to the next segment
;  450          0445  5              NEXT_LEN = 8;      ! Length of next segment
;  451          0446  5
;  452          0447  5          ! Extract the data record
;  453          0448  5          !
;  454          0449  5          BYTE_OFS = 0;
;  455          0450  5
;  456          0451  5          ! Find out where to put the data record
;  457          0452  5          !
;  458          0453  5          IF .KEY_DESC [ KEY$V_REC_COMPR ]
;  459          0454  5          THEN
;  460          0455  5              BUFFER_PTR = .CTX [ CTX$L_RDP ] + 2
;  461          0456  5          ELSE
;  462          0457  5              BUFFER_PTR = .CTX [ CTX$L_RDP ];
;  463          0458  5
;  464          0459  5          ! Loop for the entire length of the record
;  465          0460  5          !
;  466          0461  5          WHILE .BYTE_OFS LSS .CONV$GW_OUT_REC_SIZ
;  467          0462  5          DO
;  468          0463  5
;  469          0464  5              ! If the this byte is in a key segment then ignore it and move
;  470          0465  5              ! past this segment
;  471          0466  5              !
;  472          0467  5              IF IN_SEGMENT()
;  473          0468  5              THEN
;  474          0469  5                  BYTE_OFS = .NEXT_SEG + .NEXT_LEN
;  475          0470  5              ELSE
;  476          0471  5
;  477          0472  6                  BEGIN
;  478          0473  6
;  479          0474  6                  ! If the byte is not in a segment then move it into the buffer
;  480          0475  6                  !
;  481          0476  6                  CH$MOVE( .NEXT_SEG - .BYTE_OFS,
;  482          0477  6                           .CONV$GL_RECORD_PTR + .BYTE_OFS,
;  483          0478  6                           .BUFFER_PTR );
;  484          0479  6
;  485          0480  6                  BUFFER_PTR = .BUFFER_PTR + ( .NEXT_SEG - .BYTE_OFS );
;  486          0481  6                  BYTE_OFS = .NEXT_SEG + .NEXT_LEN
```

```
;  487          0482  6
;  488          0483  6                    END
;  489          0484  6
;  490          0485  4            END;            ! Extract the data record
;  491          0486  4
;  492          0487  4            ! If there is record compression then do it
;  493          0488  4            !
;  494          0489  4            IF .KEY_DESC [ KEY$V_REC_COMPR ]
;  495          0490  4            THEN
;  496          0491  5                BEGIN
;  497          0492  5
;  498          0493  5                REGISTER
;  499          0494  5                    NXT_FIELD_PTR   = 6,
;  500          0495  5                    CNT_FIELD_PTR   = 7,
;  501          0496  5                    TRU_FIELD_PTR   = 3;
;  502          0497  5
;  503          0498  5                NXT_FIELD_PTR = .CTX [ CTX$L_RDP ];
;  504          0499  5                CNT_FIELD_PTR = .BUFFER_PTR;
;  505          0500  5
;  506          0501  5                CONV$$COMPRESS_DATA();
;  507          0502  5
;  508          0503  5                ! We can now say what the data size might be
;  509          0504  5                !
;  510          0505  5                CTX [ CTX$W_RDS ] = .TRU_FIELD_PTR - .CTX [ CTX$L_RDP ]
;  511          0506  5
;  512          0507  5                END
;  513          0508  4            ELSE
;  514          0509  4
;  515          0510  4                ! If there is no compression then the data size is simular to above
;  516          0511  4                !
;  517          0512  4                CTX [ CTX$W_RDS ] = .BUFFER_PTR - .CTX [ CTX$L_RDP ]
;  518          0513  4
;  519          0514  4            END             ! Process the data
;  520          0515  4
;  521          0516  3            END             ! Prologue 3 processing
;  522          0517  2        ELSE
;  523          0518  3            BEGIN           ! Prologue 2 processing
;  524          0519  3
;  525          0520  3            LOCAL       LKP_PTR;
;  526          0521  3
;  527          0522  3            ! For non prologue 3 files the size is simply what was returned by GET_RECORD
;  528          0523  3            !
;  529          0524  3            CTX [ CTX$W_RDS ] = .CONV$GW_OUT_REC_SIZ;
;  530          0525  3
;  531          0526  3            ! The control is a bit more complicated...
;  532          0527  3            !
;  533          0528  3            ! Other files depend on var or fixed format
;  534          0529  3            !
;  535          0530  3            IF .CONV$AB_OUT_FAB [ FAB$B_RFM ] EQL FAB$C_FIX
;  536          0531  3            THEN
;  537          0532  3                CTX [ CTX$W_RCS ] = IRC$C_FIXOVHDSZ
;  538          0533  3            ELSE
;  539          0534  3                CTX [ CTX$W_RCS ] = IRC$C_VAROVHDSZ;
;  540          0535  3
;  541          0536  3            ! Check for duplicate key
;  542          0537  3            !
;  543          0538  3            ! 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   10 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 00...         MOVZBL  20(KEY_DESC), -(SP)                   : 0429
                               0000V  30 000/5         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
                                  D3  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
                               00005  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$:    PUSHL   LKP_PTR                               : 0553
                    50       1C AB42  3C 00107         MOVZWL  28(KEY_DESC)[I], R0                   : 0552
                          0000G[F40]  9F 0010C         PUSHAB  @CONV$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
```

```
;  576          0570  1 %SBTTL  'COMPARE_KEYS'
;  577          0571  1 ROUTINE COMPARE_KEYS ( SIZE,NEW : REF BLOCK [ ,BYTE ],OLD : REF BLOCK [,BYTE ] )
;  578          0572  1                                                          : CL$COMPARE_KEYS =
;  579          0573  1 !++
;  580          0574  1 !
;  581          0575  1 ! Functional Description:
;  582          0576  1 !
;  583          0577  1 !
;  584          0578  1 ! Calling Sequence:
;  585          0579  1 !
;  586          0580  1 !       COMPARE_KEYS
;  587          0581  1 !
;  588          0582  1 ! Input Parameters:
;  589          0583  1 !       none
;  590          0584  1 !
;  591          0585  1 ! Implicit Inputs:
;  592          0586  1 !       none
;  593          0587  1 !
;  594          0588  1 ! Output Parameters:
;  595          0589  1 !       none
;  596          0590  1 !
;  597          0591  1 ! Implicit Outputs:
;  598          0592  1 !       none
;  599          0593  1 !
;  600          0594  1 ! Routine Value:
;  601          0595  1 !
;  602          0596  1 !        1 - Key/segment is duplicate
;  603          0597  1 !        0 - Key/segment is not duplicate
;  604          0598  1 !       -1 - Key/segment is out of order
;  605          0599  1 !
;  606          0600  1 ! Routines Called:
;  607          0601  1 !       none
;  608          0602  1 !
;  609          0603  1 ! Side Effects:
;  610          0604  1 !       none
;  611          0605  1 !
;  612          0606  1 !--
;  613          0607  1
;  614          0608  2     BEGIN
;  615          0609  2
;  616          0610  2     DEFINE_CTX;
;  617          0611  2     DEFINE_KEY_DESC;
;  618          0612  2
;  619          0613  2     BUILTIN
;  620          0614  2         CMPP;
;  621          0615  2
;  622          0616  2     LOCAL
;  623          0617  2         DUP     : SIGNED;
;  624          0618  2
;  625          0619  2     ! If this is the first record it can't be anything
;  626          0620  2     !
;  627          0621  2     IF .CTX [ CTX$V_FST ]
;  628          0622  2     THEN
;  629          0623  3         BEGIN
;  630          0624  3
;  631          0625  3         CTX [ CTX$V_FST ] = _CLEAR;
;  632          0626  3
```

```
;  633          0627  3          RETURN 0
;  634          0628  3
;  635          0629  2          END;
;  636          0630  2
;  637          0631  3      DUP = ( CASE .KEY_DESC [ KEY$B_DATATYPE ] FROM KEY$C_STRING TO KEY$C_MAX_DATA OF
;  638          0632  3      SET
;  639          0633  3          [ KEY$C_STRING ]       : CH$COMPARE( .SIZE,.NEW,.SIZE,.OLD );
;  640          0634  3
;  641          0635  3          [ KEY$C_SGNWORD ]      : IF .NEW [ 0,WORD_S ] EQL .OLD [ 0,WORD_S ]
;  642          0636  3                                   THEN
;  643          0637  3                                       0
;  644          0638  3                                   ELSE IF .NEW [ 0,WORD_S ] LSS .OLD [ 0,WORD_S ]
;  645          0639  3                                   THEN
;  646          0640  3                                       -1
;  647          0641  3                                   ELSE
;  648          0642  3                                       ';
;  649          0643  3
;  650          0644  3          [ KEY$C_UNSGNWORD ]    : IF .NEW [ 0,WORD_U ] EQLU .OLD [ 0,WORD_U ]
;  651          0645  3                                   THEN
;  652          0646  3                                       0
;  653          0647  3                                   ELSE IF .NEW [ 0,WORD_U ] LSSU .OLD [ 0,WORD_U ]
;  654          0648  3                                   THEN
;  655          0649  3                                       -1
;  656          0650  3                                   ELSE
;  657          0651  3                                       1;
;  658          0652  3
;  659          0653  3          [ KEY$C_SGNLONG ]      : IF .NEW [ 0,LONG_S ] EQL .OLD [ 0,LONG_S ]
;  660          0654  3                                   THEN
;  661          0655  3                                       0
;  662          0656  3                                   ELSE IF .NEW [ 0,LONG_S ] LSS .OLD [ 0,LONG_S ]
;  663          0657  3                                   THEN
;  664          0658  3                                       -1
;  665          0659  3                                   ELSE
;  666          0660  3                                       1;
;  667          0661  3
;  668          0662  3          [ KEY$C_UNSGNLONG ]    : IF .NEW [ 0,LONG_U ] EQLU .OLD [ 0,LONG_U ]
;  669          0663  3                                   THEN
;  670          0664  3                                       0
;  671          0665  3                                   ELSE IF .NEW [ 0,LONG_U ] LSSU .OLD [ 0,LONG_U ]
;  672          0666  3                                   THEN
;  673          0667  3                                       -1
;  674          0668  3                                   ELSE
;  675          0669  3                                       1;
;  676          0670  3
;  677          0671  4          [ KEY$C_PACKED ]       : BEGIN
;  678          0672  4
;  679          0673  4                                   LOCAL NIBBLES;
;  680          0674  4
;  681          0675  4                                   NIBBLES = ( .SIZE * 2 ) - 1;
;  682          0676  4
;  683          0677  4                                   CMPP( NIBBLES,.NEW,NIBBLES,.OLD )
;  684          0678  4
;  685          0679  3                                   END;
;  686          0680  3
;  687          0681  3          [ KEY$C_SGNQUAD ]      :
;  688          0682  3                                   ! Check the high longword first
;  689          0683  3                                   !
```

```
;  690          0684  3                                IF .NEW [ 4,LONG_S ] EQL .OLD [ 4,LONG_S ]
;  691          0685  3                                THEN
;  692          0686  3
;  693          0687  3                                    ! If the high long word is eq then check the lower
;  694          0688  3                                    !
;  695          0689  3                                    IF .NEW [ 0,LONG_U ] EQL .OLD [ 0,LONG_U ]
;  696          0690  3                                    THEN
;  697          0691  3                                        0
;  698          0692  3                                    ELSE IF .NEW [ 0,LONG_U ] LSS .OLD [ 0,LONG_U ]
;  699          0693  3                                    THEN
;  700          0694  3                                        -1
;  701          0695  3                                    ELSE
;  702          0696  3                                        1
;  703          0697  3                                ELSE IF .NEW [ 4,LONG_S ] LSS .OLD [ 4,LONG_S ]
;  704          0698  3                                THEN
;  705          0699  3                                    -1
;  706          0700  3                                ELSE
;  707          0701  3                                    1;
;  708          0702  3
;  709          0703  3          [ KEY$C_UNSGNQUAD ]   : IF .NEW [ 4,LONG_U ] EQLU .OLD [ 4,LONG_U ]
;  710          0704  3                                THEN
;  711          0705  3                                    IF .NEW [ 0,LONG_U ] EQLU .OLD [ 0,LONG_U ]
;  712          0706  3                                    THEN
;  713          0707  3                                        0
;  714          0708  3                                    ELSE IF .NEW [ 0,LONG_U ] LSSU .OLD [ 0,LONG_U ]
;  715          0709  3                                    THEN
;  716          0710  3                                        -1
;  717          0711  3                                    ELSE
;  718          0712  3                                        1
;  719          0713  3                                ELSE IF .NEW [ 4,LONG_U ] LSSU .OLD [ 4,LONG_U ]
;  720          0714  3                                THEN
;  721          0715  3                                    -1
;  722          0716  3                                ELSE
;  723          0717  3                                    1;
;  724          0718  3
;  725          0719  2              TES );
;  726          0720  2
;  727          0721  2          ! Fix the meaning of the flag
;  728          0722  2          !
;  729          0723  2          IF .DUP EQL 0
;  730          0724  2          THEN
;  731          0725  2              DUP = 1
;  732          0726  2          ELSE
;  733          0727  2              IF .DUP EQL 1
;  734          0728  2              THEN
;  735          0729  2                  DUP = 0;
;  736          0730  2
;  737          0731  2          ! Return the value
;  738          0732  2          !
;  739          0733  2          RETURN .DUP
;  740          0734  2
;  741          0735  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
```

```
;  743          0736  1 %SBTTL  'IN_SEGMENT'
;  744          0737  1 ROUTINE IN_SEGMENT : CL$IN_SEGMENT =
;  745          0738  1 !++
;  746          0739  1 !
;  747          0740  1 ! Functional Description:
;  748          0741  1 !
;  749          0742  1 !       Determines if the current position in the data record is in a key
;  750          0743  1 !       segment or not
;  751          0744  1 !
;  752          0745  1 ! Calling Sequence:
;  753          0746  1 !
;  754          0747  1 !       IN_SEGMENT()
;  755          0748  1 !
;  756          0749  1 ! Input Parameters:
;  757          0750  1 !       none
;  758          0751  1 !
;  759          0752  1 ! Implicit Inputs:
;  760          0753  1 !
;  761          0754  1 !       BYTE_OFS - R6   Offset into the data record
;  762          0755  1 !       NEXT_SEG - R7   Start of the next segment
;  763          0756  1 !       NEXT_LEN - R8   Length of next segment
;  764          0757  1 !
;  765          0758  1 ! Output Parameters:
;  766          0759  1 !       none
;  767          0760  1 !
;  768          0761  1 ! Implicit Outputs:
;  769          0762  1 !       none
;  770          0763  1 !
;  771          0764  1 ! Routine Value:
;  772          0765  1 !
;  773          0766  1 !       1 - Pointer is in a key segment
;  774          0767  1 !       0 - Pointer is NOT in a key segment
;  775          0768  1 !
;  776          0769  1 ! Routines Called:
;  777          0770  1 !       none
;  778          0771  1 !
;  779          0772  1 ! Side Effects:
;  780          0773  1 !       none
;  781          0774  1 !
;  782          0775  1 !--
;  783          0776  1
;  784          0777  2     BEGIN
;  785          0778  2
;  786          0779  2     DEFINE_KEY_DESC;
;  787          0780  2
;  788          0781  2     EXTERNAL REGISTER
;  789          0782  2              BYTE_OFS = 6,          ! Offset into the data record
;  790          0783  2              NEXT_SEG = 7,          ! Start of the next segment
;  791          0784  2              NEXT_LEN = 8;          ! Length of next segment
;  792          0785  2
;  793          0786  2     LOCAL        SEGMENT : SIGNED;
;  794          0787  2
;  795          0788  2     NEXT_SEG = .CONV$GW_OUT_REC_SIZ;
;  796          0789  2     NEXT_LEN = 0;
;  797          0790  2
;  798          0791  2     SEGMENT = .KEY_DESC [ KEY$B_SEGMENTS ] - 1;
;  799          0792  2
```

```
;  800          0793  2     ! Check all of the segments
;  801          0794  2     !
;  802          0795  2     WHILE .SEGMENT GEQ 0
;  803          0796  2     DO
;  804          0797  3         BEGIN
;  805          0798  3
;  806          0799  3         ! See if we  are past a segment if so see if we are in it else
;  807          0800  3         ! check the next segment
;  808          0801  3         !
;  809          0802  3         IF .BYTE_OFS GEQ .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ]
;  810          0803  3         THEN
;  811          0804  4             BEGIN
;  812          0805  4
;  813          0806  4             ! See if we are in the segment.  If so return with the pointers
;  814          0807  4             !
;  815          0808  5             IF .BYTE_OFS LSS ( .KEY_DESC [ ( 44 + .SEGMENT ),BYTE_U ] +
;  816          0809  5                                .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ] )
;  817          0810  4             THEN
;  818          0811  5                 BEGIN
;  819          0812  5                 NEXT_SEG = .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ];
;  820          0813  5                 NEXT_LEN = .KEY_DESC [ ( 44 + .SEGMENT ),BYTE_U ];
;  821          0814  5                 RETURN 1
;  822          0815  5                 END
;  823          0816  4             END
;  824          0817  3         ELSE
;  825          0818  3
;  826          0819  3             ! If this segment is closer than the last one then change pointers
;  827          0820  3             !
;  828          0821  3             IF .NEXT_SEG GTR .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ]
;  829          0822  3             THEN
;  830          0823  4                 BEGIN
;  831          0824  4                 NEXT_SEG = .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ];
;  832          0825  4                 NEXT_LEN = .KEY_DESC [ ( 44 + .SEGMENT ),BYTE_U ]
;  833          0826  3                 END;
;  834          0827  3
;  835          0828  3
;  836          0829  3         ! Check the next segment
;  837          0830  3         !
;  838          0831  3         SEGMENT = .SEGMENT - 1
;  839          0832  2         END;
;  840          0833  2
;  841          0834  2     RETURN 0
;  842          0835  2
;  843          0836  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
```

```
;   845          0837  1 %SBTTL  'COMPRESS_KEY'
;   846          0838  1 GLOBAL ROUTINE  CONV$$COMPRESS_KEY : CL$JSB_REG_9 NOVALUE =
;   847          0839  1 !++
;   848          0840  1 !
;   849          0841  1 ! Functional Description:
;   850          0842  1 !
;   851          0843  1 !       Does primary data level key compression for prologue 3 files
;   852          0844  1 !
;   853          0845  1 ! Calling Sequence:
;   854          0846  1 !
;   855          0847  1 !       CONV$$COMPRESS_KEY()
;   856          0848  1 !
;   857          0849  1 ! Input Parameters:
;   858          0850  1 !       none
;   859          0851  1 !
;   860          0852  1 ! Implicit Inputs:
;   861          0853  1 !       none
;   862          0854  1 !
;   863          0855  1 ! Output Parameters:
;   864          0856  1 !       none
;   865          0857  1 !
;   866          0858  1 ! Implicit Outputs:
;   867          0859  1 !       none
;   868          0860  1 !
;   869          0861  1 ! Routine Value:
;   870          0862  1 !       none
;   871          0863  1 !
;   872          0864  1 ! Side Effects:
;   873          0865  1 !       none
;   874          0866  1 !
;   875          0867  1 !--
;   876          0868  1
;   877          0869  2     BEGIN
;   878          0870  2
;   879          0871  2     DEFINE_CTX;
;   880          0872  2     DEFINE_KEY_DESC;
;   881          0873  2
;   882          0874  2     LOCAL
;   883          0875  2         KEY             : REF VECTOR [ ,BYTE ],
;   884          0876  2         LENGTH,
;   885          0877  2         COUNT,
;   886          0878  2         RECORD_CTRL     : REF BLOCK [ ,BYTE ];
;   887          0879  2
;   888          0880  2     ! The key is in different place depending on ref
;   889          0881  2     !
;   890          0882  2     IF .KEY_DESC [ KEY$B_KEYREF ] EQL 0
;   891          0883  2     THEN
;   892          0884  2         KEY = .CTX [ CTX$L_RCP ] + 13
;   893          0885  2     ELSE
;   894          0886  2         KEY = .CTX [ CTX$L_RCP ] + 4;
;   895          0887  2
;   896          0888  2     ! Start with the full length of the key
;   897          0889  2     !
;   898          0890  2     LENGTH = .KEY_DESC [ KEY$B_KEYSZ ];
;   899          0891  2     COUNT = 0;
;   900          0892  2
;   901          0893  2     ! If this is not the first record in the bucket then compress off the front
```

```
;  902     0894  2      !
;  903     0895  2      IF .CTX [ CTX$W_USE ] NEQU 0
;  904     0896  2      THEN
;  905     0897  2          BEGIN
;  906     0898  2
;  907     0899  2          BIND
;  908     0900  2              LASTKEY     = .CTX [ CTX$L_LKP ] : VECTOR [ ,BYTE ];
;  909     0901  2
;  910     0902  2          ! Compaire with the last key
;  911     0903  2          !
;  912     0904  2          INCR I FROM 0 TO ( .LENGTH - 1 ) BY 1
;  913     0905  2          DO
;  914     0906  2
;  915     0907  2              ! If this character is the same as the character in the last key
;  916     0908  2              ! then it can be compressed off.
;  917     0909  2              !
;  918     0910  2              IF .LASTKEY [ .I ] EQLU .KEY [ .I ]
;  919     0911  2              THEN
;  920     0912  2                  COUNT = .COUNT + 1
;  921     0913  2              ELSE
;  922     0914  2                  EXITLOOP;
;  923     0915  2
;  924     0916  2          ! The key could be a little bit shoter
;  925     0917  2          !
;  926     0918  2          LENGTH = .LENGTH - .COUNT;
;  927     0919  2
;  928     0920  2          ! If we truncated off bytes we must move what is left of the key in a little
;  929     0921  2          !
;  930     0922  2          CH$MOVE ( .LENGTH,KEY [ .COUNT ],KEY [ 0 ] )
;  931     0923  2
;  932     0924  2          END;
;  933     0925  2
;  934     0926  2      ! If there are some characters left do rear end truncation
;  935     0927  2      !
;  936     0928  2      IF .LENGTH GTR 1
;  937     0929  2      THEN
;  938     0930  2
;  939     0931  2          ! Start looking from the back to compress off trailing dups
;  940     0932  2          !
;  941     0933  2          DECR I FROM ( .LENGTH - 1 ) TO 1 BY 1
;  942     0934  2          DO
;  943     0935  2
;  944     0936  2              ! If this character is a dup cut it off by shorting the length
;  945     0937  2              !
;  946     0938  2              IF .KEY [ .I - 1 ] EQLU .KEY [ .I ]
;  947     0939  2              THEN
;  948     0940  2                  LENGTH = .LENGTH - 1
;  949     0941  2              ELSE
;  950     0942  2                  EXITLOOP;
;  951     0943  2
;  952     0944  2      ! Point to the control fields in front of the key
;  953     0945  2      !
;  954     0946  2      RECORD_CTRL = .KEY - 4;
;  955     0947  2
;  956     0948  2      ! Set the length field in the record
;  957     0949  2      !
;  958     0950  2      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                   ADDL3   #13, 48(CTX), KEY                         ; 0884
                                        05  11 0000E        BRB     2$                                        ;
                     56         30    AA                   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
```

```
;  978          0969  1 %SBTTL  'COMPRESS_INDEX'
;  979          0970  1 GLOBAL ROUTINE  CONV$$COMPRESS_INDEX : CL$JSB_REG_9 NOVALUE =
;  980          0971  1 !++
;  981          0972  1 !
;  982          0973  1 ! Functional Description:
;  983          0974  1 !
;  984          0975  1 !       Does prologue 3 index compression on the index level record
;  985          0976  1 !
;  986          0977  1 ! Calling Sequence:
;  987          0978  1 !
;  988          0979  1 !       CONV$$COMPRESS_INDEX()
;  989          0980  1 !
;  990          0981  1 ! Input Parameters:
;  991          0982  1 !       none
;  992          0983  1 !
;  993          0984  1 ! Implicit Inputs:
;  994          0985  1 !       none
;  995          0986  1 !
;  996          0987  1 ! Output Parameters:
;  997          0988  1 !       none
;  998          0989  1 !
;  999          0990  1 ! Implicit Outputs:
; 1000          0991  1 !       none
; 1001          0992  1 !
; 1002          0993  1 ! Routine Value:
; 1003          0994  1 !       none
; 1004          0995  1 !
; 1005          0996  1 ! Side Effects:
; 1006          0997  1 !       none
; 1007          0998  1 !
; 1008          0999  1 !--
; 1009          1000  1
; 1010          1001  2     BEGIN
; 1011          1002  2
; 1012          1003  2     DEFINE_CTX;
; 1013          1004  2     DEFINE_BUCKET;
; 1014          1005  2     DEFINE_KEY_DESC;
; 1015          1006  2
; 1016          1007  2     LOCAL
; 1017          1008  2         LENGTH,
; 1018          1009  2         COUNT,
; 1019          1010  2         RECORD_CTRL      : REF BLOCK [ ,BYTE ];
; 1020          1011  2
; 1021          1012  2     BIND
; 1022          1013  2         KEY      = .CTX [ CTX$L_RDP ] + 2 : VECTOR [ ,BYTE ];
; 1023          1014  2
; 1024          1015  2     ! Init. counts
; 1025          1016  2     !
; 1026          1017  2     LENGTH = .KEY_DESC [ KEY$B_KEYSZ ];
; 1027          1018  2     COUNT = 0;
; 1028          1019  2
; 1029          1020  2     ! If this is not the first record in the bucket then do front compression
; 1030          1021  2     !
; 1031          1022  2     IF .CTX [ CTX$W_USE ] NEQU 0
; 1032          1023  2     THEN
; 1033          1024  3         BEGIN
; 1034          1025  3
```

```
;  1035     1026  3          BIND
;  1036     1027  3              LASTKEY     = .CTX [ CTX$L_LKP ] : VECTOR [ ,BYTE ];
;  1037     1028  3
;  1038     1029  3          ! Compare this key with the last one
;  1039     1030  3          !
;  1040     1031  3          INCR I FROM 0 TO ( .LENGTH - 1 ) BY 1
;  1041     1032  3          DO
;  1042     1033  3
;  1043     1034  3              ! Count the characters that are the same
;  1044     1035  3              !
;  1045     1036  3              IF .LASTKEY [ .I ] EQLU .KEY [ .I ]
;  1046     1037  3              THEN
;  1047     1038  3                  COUNT = .COUNT + 1
;  1048     1039  3              ELSE
;  1049     1040  3                  EXITLOOP;
;  1050     1041  3
;  1051     1042  3              ! The length of the key is a bit shorter now
;  1052     1043  3              !
;  1053     1044  3              LENGTH = .LENGTH - .COUNT
;  1054     1045  3
;  1055     1046  2              END;
;  1056     1047  2
;  1057     1048  2      ! Copy this key so that we have a last key to look at
;  1058     1049  2      !
;  1059     1050  2      CH$MOVE ( .KEY_DESC [ KEY$B_KEYSZ ],
;  1060     1051  2                .CTX [ CTX$L_RDP ] + 2,
;  1061     1052  2                .CTX [ CTX$L_LKP ] );
;  1062     1053  2
;  1063     1054  2      ! If we shortened the key from the front move it
;  1064     1055  2      !
;  1065     1056  2      CH$MOVE ( .LENGTH,KEY [ .COUNT ],KEY [ 0 ] );
;  1066     1057  2
;  1067     1058  2      ! If there are some characters left do rear end truncation
;  1068     1059  2      !
;  1069     1060  2      IF .LENGTH GTR 1
;  1070     1061  2      THEN
;  1071     1062  2
;  1072     1063  2          ! Start looking from the back to compress off trailing dups
;  1073     1064  2          !
;  1074     1065  2          DECR I FROM ( .LENGTH - 1 ) TO 1 BY 1
;  1075     1066  2          DO
;  1076     1067  2
;  1077     1068  2              ! If this character is a dup cut it off by shorting the length
;  1078     1069  2              !
;  1079     1070  2              IF .KEY [ .I - 1 ] EQLU .KEY [ .I ]
;  1080     1071  2              THEN
;  1081     1072  2                  LENGTH = .LENGTH - 1
;  1082     1073  2              ELSE
;  1083     1074  2                  EXITLOOP;
;  1084     1075  2
;  1085     1076  2      ! Set up the fields in the record
;  1086     1077  2      !
;  1087     1078  2      RECORD_CTRL = .CTX [ CTX$L_RDP ];
;  1088     1079  2
;  1089     1080  2      ! Set the record size field in the record
;  1090     1081  2      !
;  1091     1082  2      RECORD_CTRL [ 0,BYTE_U ] = .LENGTH;
```

```
;  1092          1083  2
;  1093          1084  2      ! Set the compression count
;  1094          1085  2      !
;  1095          1086  2      RECORD_CTRL [ 1,BYTE_U ] = .COUNT;
;  1096          1087  2
;  1097          1088  2      ! Set the new size of the data part of the record
;  1098          1089  2      !
;  1099          1090  2      CTX [ CTX$W_RDS ] = .LENGTH + 2;
;  1100          1091  2
;  1101          1092  2      RETURN
;  1102          1093  2
;  1103          1094  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
```

```
;  1105          1095  1 %SBTTL  'MAKE_INDEX'
;  1106          1096  1 GLOBAL ROUTINE  CONV$$MAKE_INDEX : CL$JSB_REG_9 NOVALUE =
;  1107          1097  1 !++
;  1108          1098  1 !
;  1109          1099  1 ! Functional Description:
;  1110          1100  1 !
;  1111          1101  1 !       Makes an index record for the current record being processed
;  1112          1102  1 !       It also copys the key into the last key buffer
;  1113          1103  1 !
;  1114          1104  1 ! Calling Sequence:
;  1115          1105  1 !
;  1116          1106  1 !       CONV$$MAKE_INDEX
;  1117          1107  1 !
;  1118          1108  1 ! Input Parameters:
;  1119          1109  1 !       none
;  1120          1110  1 !
;  1121          1111  1 ! Implicit Inputs:
;  1122          1112  1 !       none
;  1123          1113  1 !
;  1124          1114  1 ! Output Parameters:
;  1125          1115  1 !       none
;  1126          1116  1 !
;  1127          1117  1 ! Implicit Outputs:
;  1128          1118  1 !       none
;  1129          1119  1 !
;  1130          1120  1 ! Routine Value:
;  1131          1121  1 !       none
;  1132          1122  1 !
;  1133          1123  1 ! Side Effects:
;  1134          1124  1 !       none
;  1135          1125  1 !
;  1136          1126  1 !--
;  1137          1127  1
;  1138          1128  2     BEGIN
;  1139          1129  2
;  1140          1130  2     DEFINE_CTX;
;  1141          1131  2     DEFINE_KEY_DESC;
;  1142          1132  2
;  1143          1133  2     LOCAL
;  1144          1134  2         OFFSET,
;  1145          1135  2         IDX_PTR,
;  1146          1136  2         CTX_P1  : REF BLOCK [ ,BYTE ];
;  1147          1137  2
;  1148          1138  2     ! We need the contex block from the level above
;  1149          1139  2     !
;  1150          1140  2     CTX_P1 = .CTX + CTX$K_BLN;
;  1151          1141  2
;  1152          1142  2     ! Account for control bytes if any
;  1153          1143  2     !
;  1154          1144  2     IF .KEY_DESC [ KEY$V_IDX_COMPR ]
;  1155          1145  2     THEN
;  1156          1146  2         OFFSET = 2
;  1157          1147  2     ELSE
;  1158          1148  2         OFFSET = 0;
;  1159          1149  2
;  1160          1150  2     ! Point to the destination
;  1161          1151  2     !
```

```
;  1162     1152  2      IDX_PTR = .CTX_P1 [ CTX$L_RDP ] + .OFFSET;
;  1163     1153  2
;  1164     1154  2      ! Copy the current key into the index buffer.
;  1165     1155  2      !
;  1166     1156  2      IF .KEY_DESC [ KEY$B_KEYREF ] EQLU 0
;  1167     1157  2      THEN
;  1168     1158  3          BEGIN
;  1169     1159  3
;  1170     1160  3          BIND
;  1171     1161  3              SEG_SIZE    = .KEY_DESC + 44 : VECTOR [ ,BYTE ],
;  1172     1162  3              SEG_POSI    = .KEY_DESC + 28 : VECTOR [ ,WORD ];
;  1173     1163  3
;  1174     1164  3          LOCAL       REC_PTR;
;  1175     1165  3
;  1176     1166  3          ! Copy the key into the index key buffer
;  1177     1167  3          !
;  1178     1168  3          INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
;  1179     1169  3          DO
;  1180     1170  4              BEGIN
;  1181     1171  4
;  1182     1172  4              ! Point to the correct spot in the source string
;  1183     1173  4              !
;  1184     1174  4              REC_PTR = .CONV$GL_RECORD_PTR + .SEG_POSI [ .I ];
;  1185     1175  4
;  1186     1176  4              CH$MOVE( .SEG_SIZE [ .I ],.REC_PTR,.IDX_PTR );
;  1187     1177  4
;  1188     1178  4              ! Update the destination
;  1189     1179  4              !
;  1190     1180  4              IDX_PTR = .IDX_PTR + .SEG_SIZE [ .I ]
;  1191     1181  4
;  1192     1182  4              END
;  1193     1183  3          END
;  1194     1184  2      ELSE
;  1195     1185  2          CH$MOVE( .KEY_DESC [ KEY$B_KEYSZ ],
;  1196     1186  2                   .CONV$GL_DUP_BUF,
;  1197     1187  2                   .IDX_PTR );
;  1198     1188  2
;  1199     1189  2      ! Copy the key from the index record into last key buffer
;  1200     1190  2      !
;  1201     1191  2      CH$MOVE( .KEY_DESC [ KEY$B_KEYSZ ],
;  1202     1192  2               .CTX_P1 [ CTX$L_RDP ] + .OFFSET,
;  1203     1193  2               .CTX [ CTX$L_LKP ] );
;  1204     1194  2
;  1205     1195  2      ! Set the index record size
;  1206     1196  2      !
;  1207     1197  2      CTX_P1 [ CTX$W_RDS ] = .KEY_DESC [ KEY$B_KEYSZ ] + .OFFSET;
;  1208     1198  2
;  1209     1199  2      ! Set the index control size
;  1210     1200  2      !
;  1211     1201  2      IF .CONV$GB_PROL_V3
;  1212     1202  2      THEN
;  1213     1203  2
;  1214     1204  2          ! The record control size is simply the size of the
;  1215     1205  2          ! VBN pointer to the bucket
;  1216     1206  2          !
;  1217     1207  2          CTX_P1 [ CTX$W_RCS ] = .CTX [ CTX$V_VBN ] + 2
;  1218     1208  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$:  CLRL    OFFSET                                  ;  1148
                             04   AE       34 Bo49   9E 00017 2$:  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
```

```
;  1230          1219  1 %SBTTL  'WRITE_VBN'
;  1231          1220  1 GLOBAL ROUTINE  CONV$$WRITE_VBN : CL$JSB_REG_9 NOVALUE =
;  1232          1221  1 !++
;  1233          1222  1 !
;  1234          1223  1 ! Functional Description:
;  1235          1224  1 !
;  1236          1225  1 !       Writes the vbn pointer into the index record, also sets the control
;  1237          1226  1 !       bytes describing the vbn pointer size
;  1238          1227  1 !
;  1239          1228  1 ! Calling Sequence:
;  1240          1229  1 !
;  1241          1230  1 !       CONV$$WRITE_VBN
;  1242          1231  1 !
;  1243          1232  1 ! Input Parameters:
;  1244          1233  1 !       none
;  1245          1234  1 !
;  1246          1235  1 ! Implicit Inputs:
;  1247          1236  1 !       none
;  1248          1237  1 !
;  1249          1238  1 ! Output Parameters:
;  1250          1239  1 !       none
;  1251          1240  1 !
;  1252          1241  1 ! Implicit Outputs:
;  1253          1242  1 !       none
;  1254          1243  1 !
;  1255          1244  1 ! Routine Value:
;  1256          1245  1 !       none
;  1257          1246  1 !
;  1258          1247  1 ! Side Effects:
;  1259          1248  1 !       none
;  1260          1249  1 !
;  1261          1250  1 !--
;  1262          1251  1
;  1263          1252  2     BEGIN
;  1264          1253  2
;  1265          1254  2     DEFINE_CTX;
;  1266          1255  2     DEFINE_KEY_DESC;
;  1267          1256  2
;  1268          1257  2     LOCAL
;  1269          1258  2         VBN_SIZE,
;  1270          1259  2         CTX_P1          : REF BLOCK [ ,BYTE ],
;  1271          1260  2         RECORD_CTRL     : REF BLOCK [ ,BYTE ];
;  1272          1261  2
;  1273          1262  2     CTX_P1 = .CTX + CTX$K_BLN;
;  1274          1263  2
;  1275          1264  2     RECORD_CTRL = .CTX_P1 [ CTX$L_RCP ];
;  1276          1265  2
;  1277          1266  2     ! Get the size in bits of the vbn
;  1278          1267  2     !
;  1279          1268  2     VBN_SIZE = ( .CTX [ CTX$V_VBN ] + 2 ) * 8;
;  1280          1269  2
;  1281          1270  2     ! The control bytes are different for prologue 3 files
;  1282          1271  2     !
;  1283          1272  2     IF .CONV$GB_PROL_V3
;  1284          1273  2     THEN
;  1285          1274  2         RECORD_CTRL [ 0,0,.VBN_SIZE,0 ] = .CTX [ CTX$L_CURRENT_VBN ]
;  1286          1275  2     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$:      MOVB    R2, (RECORD_CTRL)                 ; 1277
                 A1               50      00      08   AA  F0 00026          INSV    8(CTX), #0, VBN_SIZE, 1(RECORD_CTRL) ; 1278
                                                      04  BA 0002D 2$:      POPR    #^M<R2>                           ; 1283
                                                          05 0002F          RSB                                       ;

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

```
;  1296          1284  1 %SBTTL  'COPY_KEY'
;  1297          1285  1 GLOBAL ROUTINE  CONV$$COPY_KEY ( OFFSET ) : CL$COPY_KEY NOVALUE =
;  1298          1286  1 !++
;  1299          1287  1 !
;  1300          1288  1 ! Functional Description:
;  1301          1289  1 !
;  1302          1290  1 !       Copies the data record for secondary index into the proper
;  1303          1291  1 !       data record buffer and sets RCS
;  1304          1292  1 !
;  1305          1293  1 ! Calling Sequence:
;  1306          1294  1 !
;  1307          1295  1 !       CONV$$COPY_KEY()
;  1308          1296  1 !
;  1309          1297  1 ! Input Parameters:
;  1310          1298  1 !       none
;  1311          1299  1 !
;  1312          1300  1 ! Implicit Inputs:
;  1313          1301  1 !       none
;  1314          1302  1 !
;  1315          1303  1 ! Output Parameters:
;  1316          1304  1 !       none
;  1317          1305  1 !
;  1318          1306  1 ! Implicit Outputs:
;  1319          1307  1 !       none
;  1320          1308  1 !
;  1321          1309  1 ! Routine Value:
;  1322          1310  1 !       none
;  1323          1311  1 !
;  1324          1312  1 ! Side Effects:
;  1325          1313  1 !       none
;  1326          1314  1 !
;  1327          1315  1 !--
;  1328          1316  1
;  1329          1317  2     BEGIN
;  1330          1318  2
;  1331          1319  2     DEFINE_CTX;
;  1332          1320  2     DEFINE_KEY_DESC;
;  1333          1321  2     DEFINE_RECORD_CTRL;
;  1334          1322  2
;  1335          1323  2     ! If this is a prologue 3 file then the offset is either 2 or 4
;  1336          1324  2     ! depending on compression
;  1337          1325  2     !
;  1338          1326  2     IF .CONV$GB_PROL_V3
;  1339          1327  2     THEN
;  1340          1328  3         BEGIN
;  1341          1329  3
;  1342          1330  3         IF .KEY_DESC [ KEY$V_KEY_COMPR ]
;  1343          1331  3         THEN
;  1344          1332  3             OFFSET = 4
;  1345          1333  3         ELSE
;  1346          1334  3             OFFSET = 2;
;  1347          1335  3
;  1348          1336  3         ! Set the size of the record here sice it is known, for prologue 1
;  1349          1337  3         ! it is set in load_secondary
;  1350          1338  3         !
;  1351          1339  3         RECORD_CTRL [ 0,WORD_U ] = .KEY_DESC [ KEY$B_KEYSZ ] + .OFFSET - 2
;  1352          1340  3
```

```
;  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
```

```
;  1369          1356  1 %SBTTL  'CREATE_HIGH_KEY'
;  1370          1357  1 GLOBAL ROUTINE  CONV$$CREATE_HIGH_KEY    : CL$JSB_REG_9 NOVALUE =
;  1371          1358  1 !++
;  1372          1359  1 !
;  1373          1360  1 ! Functional Description:
;  1374          1361  1 !
;  1375          1362  1 !       Creates a high key vaule index for the current key of reference
;  1376          1363  1 !       and data type
;  1377          1364  1 !
;  1378          1365  1 ! Calling Sequence:
;  1379          1366  1 !
;  1380          1367  1 !       CONV$$CREATE_HIGH_KEY()
;  1381          1368  1 !
;  1382          1369  1 ! Input Parameters:
;  1383          1370  1 !       none
;  1384          1371  1 !
;  1385          1372  1 ! Implicit Inputs:
;  1386          1373  1 !       none
;  1387          1374  1 !
;  1388          1375  1 ! Output Parameters:
;  1389          1376  1 !       none
;  1390          1377  1 !
;  1391          1378  1 ! Implicit Outputs:
;  1392          1379  1 !       none
;  1393          1380  1 !
;  1394          1381  1 ! Routine Value:
;  1395          1382  1 !       none
;  1396          1383  1 !
;  1397          1384  1 ! Side Effects:
;  1398          1385  1 !       none
;  1399          1386  1 !
;  1400          1387  1 !--
;  1401          1388  1
;  1402          1389  2     BEGIN
;  1403          1390  2
;  1404          1391  2     DEFINE_CTX;
;  1405          1392  2     DEFINE_BUCKET;
;  1406          1393  2     DEFINE_KEY_DESC;
;  1407          1394  2
;  1408          1395  2     BUILTIN
;  1409          1396  2         CVTLP;
;  1410          1397  2
;  1411          1398  2     LOCAL
;  1412          1399  2         CTX_P1          : REF BLOCK [ ,BYTE ],
;  1413          1400  2         RECORD_DATA     : REF BLOCK [ ,BYTE ];
;  1414          1401  2
;  1415          1402  2     ! Set up the control info for the index record
;  1416          1403  2     !
;  1417          1404  2     CONV$$WRITE_VBN();
;  1418          1405  2
;  1419          1406  2     ! The record we are making is actually for the level above
;  1420          1407  2     !
;  1421          1408  2     CTX_P1 = .CTX + CTX$K_BLN;
;  1422          1409  2
;  1423          1410  2     ! Get the start where the key should go
;  1424          1411  2     !
;  1425          1412  2     IF .KEY_DESC [ KEY$V_IDX_COMPR ]
```

```
;  1426          1413  2     THEN
;  1427          1414  2         RECORD_DATA = 2
;  1428          1415  2     ELSE
;  1429          1416  2         RECORD_DATA = 0;
;  1430          1417  2
;  1431          1418  2     ! We know what the size of the finished record is now
;  1432          1419  2     !
;  1433          1420  2     CTX_P1 [ CTX$W_RDS ] = .KEY_DESC [ KEY$B_KEYSZ ] + .RECORD_DATA;
;  1434          1421  2
;  1435          1422  2     ! Actually point to the record
;  1436          1423  2     !
;  1437          1424  2     RECORD_DATA = .RECORD_DATA + .CTX_P1 [ CTX$L_RDP ];
;  1438          1425  2
;  1439          1426  2     ! Loop for all of the segments
;  1440          1427  2     !
;  1441          1428  2     INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
;  1442          1429  2     DO
;  1443          1430  3         BEGIN
;  1444          1431  3
;  1445          1432  3         ! Select the data type
;  1446          1433  3         !
;  1447          1434  3         CASE .KEY_DESC [ KEY$B_DATATYPE ] FROM KEY$C_STRING TO KEY$C_MAX_DATA OF
;  1448          1435  3         SET
;  1449          1436  4             [ KEY$C_STRING ]  : BEGIN
;  1450          1437  4
;  1451          1438  4                                 LOCAL HIGH_VALUE : BYTE;
;  1452          1439  4
;  1453          1440  4                                 HIGH_VALUE = %X'FF';
;  1454          1441  4                                 CH$FILL ( .HIGH_VALUE,
;  1455          1442  4                                           .KEY_DESC [ ( 44 + .I ),BYTE_U ],
;  1456          1443  4                                           .RECORD_DATA )
;  1457          1444  3                                 END;
;  1458          1445  3
;  1459          1446  3             [ KEY$C_SGNWORD ]   : RECORD_DATA [ 0,WORD_S ] = %X'7FFF';
;  1460          1447  3
;  1461          1448  3             [ KEY$C_UNSGNWORD ] : RECORD_DATA [ 0,WORD_U ] = %X'FFFF';
;  1462          1449  3
;  1463          1450  3             [ KEY$C_SGNLONG ]   : RECORD_DATA [ 0,LONG_S ] = %X'7FFFFFFF';
;  1464          1451  3
;  1465          1452  3             [ KEY$C_UNSGNLONG ] : RECORD_DATA [ 0,LONG_U ] = %X'FFFFFFFF';
;  1466          1453  3
;  1467          1454  4             [ KEY$C_PACKED ]  : BEGIN
;  1468          1455  4
;  1469          1456  4                                 LOCAL   SIZE,
;  1470          1457  4                                         HIGH_VALUE : BYTE;
;  1471          1458  4
;  1472          1459  4                                 SIZE = .KEY_DESC [ ( 44 + .I ),BYTE_U ] - 1;
;  1473          1460  4                                 HIGH_VALUE = %X'99';
;  1474          1461  4
;  1475          1462  4                                 CH$FILL ( .HIGH_VALUE,
;  1476          1463  4                                           .SIZE,
;  1477          1464  4                                           .RECORD_DATA );
;  1478          1465  4
;  1479          1466  4                                 RECORD_DATA [ .SIZE,BYTE_U ] = %X'9C'
;  1480          1467  4
;  1481          1468  3                                 END;
;  1482          1469  3
```

```
; 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$:    MNECW   #1, (RECORD_DATA)                          ; 1448
                                                32  11 0005E        BRB     13$                                       ;
                                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
```

CONVDATA
LIS

CONVFSTLD
LIS

CONVFSTIO
LIS

CONVFSTRC
LIS

CONVFILES
LIS

CONVERROR
LIS

CONVFASTM
LIS

CONVDCL
LIS

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RECLDCL
LIS

RECLREC
LIS

RECLCTRL
LIS

RECLRMSIO
LIS

CONVMSG
LIS

CONVMAIN
LIS

CONVSORT
LIS

CONVVEC
LIS