```
   SSSSSSSSSSS     000000000     RRRRRRRRRRR     TTTTTTTTTTTTTTT     333333333       222222222
   SSSSSSSSSSS     000000000     RRRRRRRRRRRR    TTTTTTTTTTTTTTT     333333333       222222222
   SSSSSSSSSSS     000000000     RRRRRRRRRRRR    TTTTTTTTTTTTTTT     333333333       222222222
SSS              000          000  RRR         RRR        TTT        333          333  222          222
SSS              000          000  RRR         RRR        TTT        333          333  222          222
SSS              000          000  RRR         RRR        TTT        333          333  222          222
SSS              000          000  RRR         RRR        TTT                     333                222
SSS              000          000  RRR         RRR        TTT                     333                222
SSS              000          000  RRR         RRR        TTT                     333                222
   SSSSSSSSS     000          000  RRRRRRRRRRR            TTT                333                   222
   SSSSSSSSS     000          000  RRRRRRRRRRRR           TTT                333                   222
   SSSSSSSSS     000          000  RRRRRRRRRRRR           TTT                333                   222
            SSS  000          000  RRR   RRR              TTT                   333             222
            SSS  000          000  RRR   RRR              TTT                   333             222
            SSS  000          000  RRR   RRR              TTT                   333             222
            SSS  000          000  RRR      RRR           TTT        333        333          222
            SSS  000          000  RRR      RRR           TTT        333        333          222
            SSS  000          000  RRR      RRR           TTT        333        333          222
            SSS  000          000  RRR      RRR           TTT        333        333          222
SSSSSSSSSSS          000000000     RRR         RRR        TTT           333333333       222222222222222
SSSSSSSSSSS          000000000     RRR         RRR        TTT           333333333       222222222222222
SSSSSSSSSSS          000000000     RRR         RRR        TTT           333333333       222222222222222
```

FILEID**SORKEYSUB

```
  SSSSSSSS    000000    RRRRRRRR    KK      KK  EEEEEEEEEE  YY      YY    SSSSSSSS  UU      UU  BBBBBBBB
  SSSSSSSS    000000    RRRRRRRR    KK      KK  EEEEEEEEEE  YY      YY    SSSSSSSS  UU      UU  BBBBBBBB
SS          00      00  RR      RR  KK      KK  EE          YY      YY  SS          UU      UU  BB      BB
SS          00      00  RR      RR  KK      KK  EE          YY      YY  SS          UU      UU  BB      BB
SS          00      00  RR      RR  KK    KK    EE            YY  YY    SS          UU      UU  BB      BB
SS          00      00  RR      RR  KK    KK    EE            YY  YY    SS          UU      UU  BB      BB
  SSSSSS    00      00  RRRRRRRR    KKKKKK      EEEEEEEE        YY        SSSSSS    UU      UU  BBBBBBBB
  SSSSSS    00      00  RRRRRRRR    KKKKKK      EEEEEEEE        YY        SSSSSS    UU      UU  BBBBBBBB
        SS  00      00  RR  RR      KK    KK    EE              YY              SS  UU      UU  BB      BB
        SS  00      00  RR  RR      KK    KK    EE              YY              SS  UU      UU  BB      BB
        SS  00      00  RR    RR    KK      KK  EE              YY              SS  UU      UU  BB      BB      ....
        SS  00      00  RR    RR    KK      KK  EE              YY              SS  UU      UU  BB      BB      ....
SSSSSSSS      000000    RR      RR  KK      KK  EEEEEEEEEE      YY      SSSSSSSS    UUUUUUUUUU  BBBBBBBB      ....
SSSSSSSS      000000    RR      RR  KK      KK  EEEEEEEEEE      YY      SSSSSSSS    UUUUUUUUUU  BBBBBBBB      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE SOR$KEY_SUB (
;     2          0002  0                   IDENT = 'V04-000'       ! File: SORKEYSUB.B32 Edit: PDG3033
;     3          0003  0                   ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE          *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER     *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY      *
;    17          0017  1 !*  TRANSFERRED.                                                             *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
;    20          0020  1 !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
;    21          0021  1 !*  CORPORATION.                                                             *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS     *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 !
;    32          0032  1 ! FACILITY:     VAX-11 SORT/MERGE
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !
;    36          0036  1 !       This module contains the routines that build the comparison routine.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:  VAX/VMS user mode
;    39          0039  1 !
;    40          0040  1 ! AUTHOR: P. Gilbert, CREATION DATE: 14-Dec-1981
;    41          0041  1 !
;    42          0042  1 ! MODIFIED BY:
;    43          0043  1 !
;    44          0044  1 !       T03-015         Original
;    45          0045  1 !       T03-016 Add run-time check for presence of VFC area in LENADR routine.
;    46          0046  1 !               PDG 20-Dec-1982
;    47          0047  1 !       T03-017 Check for DISP[COM_ORD_MAX] (not CTX[COM_LRL_INT]) exceeding
;    48          0048  1 !               MAX_REFSIZE.  PDG 28-Dec-1982
;    49          0049  1 !       T03-018 Added clean-up routines.  PDG 6-Jan-1983
;    50          0050  1 !       T03-019 New interface for collating sequence stuff.  PDG 26-Jan-1983
;    51          0051  1 !       T03-020 Don't output the stable field for index sorts.  Change the
;    52          0052  1 !               severity of SOR$_KEY_LEN.  Save the stream number for stable
;    53          0053  1 !               merges.  PDG 27-Jan-1983
;    54          0054  1 !       T03-021 Changes for hostile environment.  PDG 3-Feb-1983
;    55          0055  1 !       T03-022 Change MOVC5s to use a pad character.  PDG 8-Feb-1983
;    56          0056  1 !       T03-023 Pass the context address to callback routines.  PDG 11-Feb-1983
;    57          0057  1 !       T03-024 Some changes with linkages.  PDG 10-Mar-1983
```

```
;    58          0058  1 !        T03-025 Fix bug in GEN_CONVERT_FLT.  Check validity of KBF_ORDER.
;    59          0059  1 !                PDG 22-Mar-1983
;    60          0060  1 !        T03-026 Redefine 2-byte opcodes to conform with STARLET's definition.
;    61          0061  1 !                PDG 4-Apr-1983
;    62          0062  1 !        T03-027 Various changes for KANJI.  PDG 2-May-1983
;    63          0063  1 !        T03-028 Test for F-floating, D-floating and decimal hardware support.
;    64          0064  1 !                PDG 10-May-1983
;    65          0065  1 !        T03-029 Allocate an extra byte in generated code to avoid a 11/750
;    66          0066  1 !                problem when the next byte is not readable.  PDG 9-Aug-1983
;    67          0067  1 !        T03-030 Set COM_EQUAL equal to 0 if it's not needed.  PDG 26-Aug-1983
;    68          0068  1 !        T03-031 Add COM_ARCHFLAG to store SYI$_ARCHFLAG.  PDG 31-Jan-1984
;    69          0069  1 !        T03-032 Change COM_RHB to COM_RHB_INP and COM_RHB_OUT.
;    70          0070  1 !                This is to avoid problems with merge, where an incoming
;    71          0071  1 !                record overwrites the VFC area for the outgoing record.
;    72          0072  1 !                PDG 24-Jul-1984
;    73          0073  1 !        T03-033 Correct diagnoses of entire key disappearing (SOR$_KEY_LEN).
;    74          0074  1 !                PDG 9-Aug-1984
;    75          0075  1 !--
```

```
;    77          0076  1 LIBRARY 'SYS$LIBRARY:LIB';
;    78          0077  1 LIBRARY 'SRC$:OPCODES';
;    79          0078  1 REQUIRE 'SRC$:COM';
;    80          0148  1
;    81          0149  1 LITERAL                         ! Global registers used between routines
;    82          0150  1     R_CUR_PC =  10,
;    83          0151  1     R_BRANCH =  9;
;    84          0152  1 LITERAL
;    85          0153  1     FUN_K_STAB= FALSE;          ! True to pass records in a stable order
;    86          0154  1 LINKAGE
;    87          0155  1     LINK_OPOPNEQ =
;    88          0156  1         JSB(REGISTER=2,REGISTER=4):
;    89          0157  1         GLOBAL(CUR_PC=R_CUR_PC, BRANCH=R_BRANCH, CTX=COM_REG_CTX)
;    90          0158  1         NOTUSED(1,5,6,7,8),
;    91          0159  1
;    92          0160  1     LINK_BNEQ =
;    93          0161  1         JSB(REGISTER=4):
;    94          0162  1         GLOBAL(CUR_PC=R_CUR_PC, BRANCH=R_BRANCH, CTX=COM_REG_CTX)
;    95          0163  1         NOTUSED(1,2,3,5,6,7,8),
;    96          0164  1
;    97          0165  1     LINK_SAVE =
;    98          0166  1         JSB(REGISTER=4):
;    99          0167  1         GLOBAL(CUR_PC=R_CUR_PC, BRANCH=R_BRANCH, CTX=COM_REG_CTX)
;   100          0168  1         NOTUSED(2,3,5,6,7,8),
;   101          0169  1
;   102          0170  1     LINK_DISP =
;   103          0171  1         JSB(REGISTER=2,REGISTER=3):
;   104          0172  1         GLOBAL(CUR_PC=R_CUR_PC)
;   105          0173  1         NOTUSED(1,4,5,6,7,8,9,11),
;   106          0174  1
;   107          0175  1     LINK_LITE =
;   108          0176  1         JSB(REGISTER=2,REGISTER=3):
;   109          0177  1         GLOBAL(CUR_PC=R_CUR_PC)
;   110          0178  1         NOTUSED(1,5,6,7,8,9,11),
;   111          0179  1
;   112          0180  1     LINK_ROOM =
;   113          0181  1         JSB(REGISTER=0):
;   114          0182  1         GLOBAL(CUR_PC=R_CUR_PC,
;   115          0183  1                 CTX=COM_REG_CTX)
;   116          0184  1         NOTUSED(6,7,8,9),
;   117          0185  1
;   118          0186  1     LINK_COMPARE =
;   119          0187  1         CALL:
;   120          0188  1         GLOBAL(CUR_PC=R_CUR_PC, BRANCH=R_BRANCH,
;   121          0189  1                 CTX=COM_REG_CTX),
;   122          0190  1
;   123          0191  1     LINK_MOVE =
;   124          0192  1         CALL:
;   125          0193  1         GLOBAL(CUR_PC=R_CUR_PC,
;   126          0194  1                 CTX=COM_REG_CTX);
;   127          0195  1
;   128          0196  1 FORWARD ROUTINE
;   129          0197  1     TKS_HACK:           NOVALUE CAL_CTXREG,
;   130          0198  1     KEY_COMPRESS:       NOVALUE CAL_CTXREG,
;   131          0199  1     COND_HAND,
;   132          0200  1 !   CHAR_HARDWARE:      CAL_CTXREG,             ! Test for Char String support
;   133          0201  1 !   EDPC_HARDWARE:      CAL_CTXREG,             ! Test for EDITPC support
```

```
;  134          0202  1 !    CRC_HARDWARE:       CAL_CTXREG,             ! Test for CRC support
;  135          0203  1 !    DCML_HARDWARE:      CAL_CTXREG,             ! Test for Decimal support
;  136          0204  1      FFLT_HARDWARE:      CAL_CTXREG,             ! Test for F-floating support
;  137          0205  1      DFLT_HARDWARE:      CAL_CTXREG,             ! Test for D-floating support
;  138          0206  1      GFLT_HARDWARE:      CAL_CTXREG,             ! Test for G-floating support
;  139          0207  1      HFLT_HARDWARE:      CAL_CTXREG,             ! Test for H-floating support
;  140          0208  1      DO_REI:             NOVALUE,
;  141          0209  1      EMIT_DISP:          NOVALUE LINK_DISP,
;  142          0210  1      SAVE_REGS:          NOVALUE LINK_SAVE,
;  143          0211  1      EMIT_BNEQ:          NOVALUE LINK_BNEQ,
;  144          0212  1      EMIT_LITE:          NOVALUE LINK_LITE,
;  145          0213  1      ROOM:                       LINK_ROOM,
;  146          0214  1      OPOPNEQ:            NOVALUE LINK_OPOPNEQ,
;  147          0215  1      GEN_CONVERT_DEC:            LINK_COMPARE,
;  148          0216  1      GEN_CONVERT_FLT:            LINK_COMPARE,
;  149          0217  1      GEN_MOVE:           NOVALUE LINK_MOVE,
;  150          0218  1      GEN_COMPARE:        NOVALUE LINK_COMPARE,
;  151          0219  1      MOVE_KEYS:                  LINK_COMPARE,
;  152          0220  1      EXPAND:             NOVALUE,
;  153          0221  1      SOR$$KEY_SUB:               CAL_CTXREG,
;  154          0222  1      CLEAN_UP:           CAL_CTXREG NOVALUE;
;  155          0223  1
;  156          0224  1  SOR$$END_ROUTINE_(CLEAN_UP);
;  157          0225  1
;  158          0226  1  EXTERNAL ROUTINE
;  159          0227  1      SOR$$ERROR,                                    ! Issue diagnostic
;  160          0228  1      %IF NOT HOSTILE %THEN
;  161          0229  1      SOR$$RDT:           CAL_CTXREG,                ! Use record defn table
;  162          0230  1      SOR$$RFA_ACCESS:    NOVALUE CAL_ACCESS,        ! Access record by RFA
;  163          0231  1      %FI
;  164          0232  1      SOR$$ALLOCATE:      CAL_CTXREG,                ! Allocate storage
;  165          0233  1      SOR$$DEALLOCATE:    CAL_CTXREG NOVALUE;        ! Deallocate storage
;  166          0234  1
;  167          0235  1  %IF NOT HOSTILE %THEN
;  168          0236  1  EXTERNAL
;  169          0237  1      LIB$AB_CVTTP_O:     ADDRESSING_MODE(GENERAL),
;  170          0238  1      LIB$AB_CVTTP_U:     ADDRESSING_MODE(GENERAL),
;  171          0239  1      LIB$AB_CVTTP_Z:     ADDRESSING_MODE(GENERAL);
;  172          0240  1  %FI
;  173          0241  1
;  174          0242  1  %IF NOT HOSTILE %THEN
;  175          0243  1  EXTERNAL LITERAL
;  176          0244  1      FUN_K_KANJI: WEAK UNSIGNED(1);
;  177          0245  1  %FI
;  178          0246  1
;  179        U 0247  1  %IF HOSTILE %THEN
;  180        U 0248  1      MACRO
;  181        U 0249  1          SYS$GETSYIW = SOR$SYS$GETSYIW %,
;  182        U 0250  1          SYS$UNWIND  = SOR$SYS$UNWIND  %;
;  183          0251  1  %FI
;  184          0252  1
;  185          0253  1  ! This bit in the key description buffer indicates a converted key
;  186          0254  1  !
;  187          0255  1  MACRO
;  188        M 0256  1      KBF_CVT =   %FIELDEXPAND(KBF_ORDER,0),
;  189          0257  1                 %FIELDEXPAND(KBF_ORDER,1) + 1, 1, 0 %;
;  190          0258  1
```

```
;  191          0259  1
;  192          0260  1 ! PLEN_ gives the number of bytes required for a packed number of X digits.
;  193          0261  1 ! LEN_ gives the number of bytes for a key.
;  194          0262  1 !
;  195          0263  1 MACRO
;  196          0264  1     PLEN_(X) = ((X)/2+1) %,
;  197        M 0265  1     LEN_(B) =
;  198        M 0266  1         BEGIN
;  199        M 0267  1         SWITCHES STRUCTURE(BLOCK[,BYTE]);       ! STRUCTURE(KBF_BLOCK)
;  200        M 0268  1         IF .B[KBF_TYPE] EQL DSC$K_DTYPE_P
;  201        M 0269  1         THEN PLEN_(.B[KBF_LENGTH])
;  202        M 0270  1         ELSE .B[KBF_LENGTH]
;  203          0271  1         END %;
;  204          0272  1
;  205          0273  1 ! Define the field within COM_ROUTINES that references the start of the
;  206          0274  1 ! generated code.
;  207          0275  1 !
;  208          0276  1 MACRO
;  209          0277  1     S_START =   %EXPAND %FIELDEXPAND(COM_ROUTINES,0) + 1, 0, 32, 0 %;
```

```
;  211          0278  1 !
;  212          0279  1 ! The following tables contain the largest allowed sizes for the various
;  213          0280  1 ! datatypes (a value of -1 indicates no upper limit); a bitvector
;  214          0281  1 ! indicating that the length, if specified, should match the maximum length;
;  215          0282  1 ! and a bit indicating that the datatype can be compared like a binary number.
;  216          0283  1 !
;  217          0284  1 MACRO
;  218        M 0285  1     DSC_SUPPORTED =
;  219        M 0286  1             DSC$K_DTYPE_ADT,    8,      TRUE,   TRUE,
;  220        M 0287  1             DSC$K_DTYPE_B,      1,      TRUE,   TRUE,
;  221        M 0288  1             DSC$K_DTYPE_BU,     1,      TRUE,   TRUE,
;  222        M 0289  1             DSC$K_DTYPE_D,      8,      TRUE,   FALSE,
;  223        M 0290  1             DSC$K_DTYPE_F,      4,      TRUE,   FALSE,
;  224        M 0291  1             DSC$K_DTYPE_G,      8,      TRUE,   FALSE,
;  225        M 0292  1             DSC$K_DTYPE_H,      16,     TRUE,   FALSE,
;  226        M 0293  1             DSC$K_DTYPE_L,      4,      TRUE,   TRUE,
;  227        M 0294  1             DSC$K_DTYPE_LU,     4,      TRUE,   TRUE,
;  228        M 0295  1             DSC$K_DTYPE_NL,     32,     FALSE,  FALSE,
;  229        M 0296  1             DSC$K_DTYPE_NLO,    31,     FALSE,  FALSE,
;  230        M 0297  1             DSC$K_DTYPE_NR,     32,     FALSE,  FALSE,
;  231        M 0298  1             DSC$K_DTYPE_NRO,    31,     FALSE,  FALSE,
;  232        M 0299  1             DSC$K_DTYPE_NU,     31,     FALSE,  FALSE,
;  233        M 0300  1             DSC$K_DTYPE_NZ,     31,     FALSE,  FALSE,
;  234        M 0301  1             DSC$K_DTYPE_O,      16,     TRUE,   TRUE,
;  235        M 0302  1             DSC$K_DTYPE_OU,     16,     TRUE,   TRUE,
;  236        M 0303  1             DSC$K_DTYPE_P,      31,     FALSE,  FALSE,
;  237        M 0304  1             DSC$K_DTYPE_Q,      8,      TRUE,   TRUE,
;  238        M 0305  1             DSC$K_DTYPE_QU,     8,      TRUE,   TRUE,
;  239        M 0306  1             DSC$K_DTYPE_T,      -1,     FALSE,  FALSE,
;  240        M 0307  1             DSC$K_DTYPE_W,      2,      TRUE,   TRUE,
;  241        M 0308  1             DSC$K_DTYPE_WU,     2,      TRUE,   TRUE,
;  242          0309  1             DSC$K_DTYPE_Z,      -1,     FALSE,  FALSE %;
;  243          0310  1 LITERAL
;  244          0311  1     MAX_SUPPORTED = DSC$K_DTYPE_ADT;    ! Value of largest supported data type
;  245          0312  1 MACRO
;  246          0313  1     DSC_L_[A,B,C,D] = [A] = B %,
;  247          0314  1     DSC_F_[A,B,C,D] = [A] = C %,
;  248          0315  1     DSC_B_[A,B,C,D] = [A] = D %;
;  249          0316  1 OWN
;  250          0317  1     DSC_LENGTH: VECTOR[MAX_SUPPORTED+1,BYTE,SIGNED]
;  251          0318  1         PSECT(SOR$RO_CODE) PRESET(DSC_L_(DSC_SUPPORTED)),
;  252          0319  1     DSC_FORCE: BITVECTOR[MAX_SUPPORTED+1]
;  253          0320  1         PSECT(SOR$RO_CODE) PRESET(DSC_F_(DSC_SUPPORTED)),
;  254          0321  1     DSC_BINARY: BITVECTOR[MAX_SUPPORTED+1]
;  255          0322  1         PSECT(SOR$RO_CODE) PRESET(DSC_B_(DSC_SUPPORTED));
```

```
;  257          0323  1 ! Macroes to emit a sequence of bytes.
;  258          0324  1 !
;  259          0325  1 MACRO
;  260        M 0326  1     EMIT_4[W,X,Y,Z] =
;  261        M 0327  1         %IF NOT %NULL(Z) %THEN EMIT_LONG((W)+(X)^8+(Y)^16+(Z)^24) %ELSE
;  262        M 0328  1         %IF NOT %NULL(X) %THEN EMIT_WORD((W)+(X)^8)
;  263        M 0329  1         %IF NOT %NULL(Y) %THEN;EMIT_BYTE((Y)) %FI %ELSE
;  264          0330  1         %IF NOT %NULL(W) %THEN EMIT_BYTE((W)) %FI %FI %FI %,
;  265        M 0331  1     EMIT_BYTES[] =
;  266        M 0332  1         BEGIN
;  267        M 0333  1         EMIT_4(%REMAINING)
;  268          0334  1         END %,
;  269          0335  1     EMIT_BYTE(X) = CH$WCHAR_A(X,CUR_PC) %,
;  270          0336  1     EMIT_WORD(X) = (CUR_PC[0,0,16,0] = X; CUR_PC = .CUR_PC+2) %,
;  271          0337  1     EMIT_LONG(X) = (CUR_PC[0,0,32,0] = X; CUR_PC = .CUR_PC+4) %,
;  272          0338  1     !
;  273          0339  1     ! Emit an absolute address
;  274          0340  1     !
;  275          0341  1     EMIT_ABSA(X) = (EMIT_BYTE(M_AID+R_PC); EMIT_LONG(X)) %;
;  276          0342  1 LITERAL
;  277          0343  1     K_ABSA = 5;
;  278          0344  1
;  279          0345  1 LITERAL
;  280          0346  1     K_BYTE =    1,
;  281          0347  1     K_WORD =    2,
;  282          0348  1     K_LONG =    4;
```

```
;  284          0349  1 ! Define the value of the largest short literal
;  285          0350  1 !
;  286          0351  1 LITERAL
;  287          0352  1     SHORT_LIT = 63;     ! Largest short literal
;  288          0353  1
;  289          0354  1 ! None of the code generated by these routines should reference registers
;  290          0355  1 ! COM_REG_SRC1, COM_REG_SRC2 or COM_REG_CTX except by these names.  Therefore,
;  291          0356  1 ! undeclare the R_x names for these registers.
;  292          0357  1 !
;  293          0358  1 UNDECLARE
;  294          0359  1     %NAME('R_',%NUMBER(COM_REG_SRC1)),
;  295          0360  1     %NAME('R_',%NUMBER(COM_REG_SRC2)),
;  296          0361  1     %NAME('R_',%NUMBER(COM_REG_CTX));
```

```
;  298          0362  1 !
;  299          0363  1 ! Saving and restoring registers in the key comparison routines.
;  300          0364  1 !
;  301          0365  1 ! The linkage to the key comparison routine allows only registers R0..R5 to be
;  302          0366  1 ! used, register R0 is the returned value, and register R1 need not be saved.
;  303          0367  1 ! When any of R2..R5 must be saved, SAVE_REGS is called with a mask of the
;  304          0368  1 ! registers to save.  This may generate code, and affects the code generated
;  305          0369  1 ! by EMIT_BNEQ to restore saved registers.
;  306          0370  1 !
;  307          0371  1 ! When registers are saved (with a PUSHR), the mask of saved registers is
;  308          0372  1 ! updated.  EMIT_BNEQ will generate appropriate code to branch to (or around)
;  309          0373  1 ! code to restore the saved registers and return (in R0) plus or minus one.
;  310          0374  1 !
;  311          0375  1 ! EMIT_BNEQ identifies the appropriate action based on its parameter, which is
;  312          0376  1 ! used as an index into the BRANCH vector.  This parameter is one of:
;  313          0377  1 !       K_U     Unsigned ascending      K_U+1   Unsigned descending
;  314          0378  1 !       K_S     Signed ascending        K_S+1   Si  ed descending
;  315          0379  1 !
;  316          0380  1 ! The following code may be generated.  Choices are listed by preference.
;  317          0381  1 !
;  318          0382  1 !       {
;  319          0383  1 !         { BNEQ {UA:UD SA:SD} : BEQL 1$/BRW {UA:UD:SA:SD}/1$: }
;  320          0384  1 !       :
;  321          0385  1 !         { UA:BLEQU 1$ : UD:BGEQU 1$ : SA:BLEQ 1$ : SD:BGEQ 1$ }
;  322          0386  1 !         POPR #^M<mask>/MOVL #1,R0/RSB
;  323          0387  1 !         1$: { BNEQ M : BEQL 2$/M:POPR #^M<mask>/MNEGL #1,R0/RSB/2$: }
;  324          0388  1 !       }
;  325          0389  1 !
;  326          0390  1 ! Thus, from 2 to 16 bytes of code are generated per EMIT_BNEQ.  The branches
;  327          0391  1 ! to UA, UD, SA, SD, or M are taken only if that label has been defined, is
;  328          0392  1 ! withing range, and restores the appropriate registers.  POPRs are generated
;  329          0393  1 ! only if registers must be restored.
;  330          0394  1 !
;  331          0395  1 ! A zero is returned at the end of the key comparison routine by the following.
;  332          0396  1 !
;  333          0397  1 !       POPR #^M<mask>/CLRL R0/RSB
;  334          0398  1 !
;  335          0399  1 ! For each label, the following information is stored (offsets are from the
;  336          0400  1 ! beginning of generated code).
;  337          0401  1 !
;  338          0402  1 !       The offset to the label (-1 indicates the label hasn't been generated).
;  339          0403  1 !       The mask of registers that are restored at that label.
;  340          0404  1 !
;  341          0405  1 ! Note that the registers must be saved in order.  That is, if Ri is saved,
;  342          0406  1 ! then Rj (with j < i) cannot later be saved.  This should be no problem,
;  343          0407  1 ! since all register saves are from R0..Rk.
;  344          0408  1 !
;  345          0409  1 LITERAL
;  346          0410  1     K_U  =      0,      ! Unsigned ascending    (descending is one greater)
;  347          0411  1     K_S  =      2;      ! Signed ascending      (descending is one greater)
;  348          0412  1
;  349          0413  1 OWN
;  350          0414  1     OPC_BRANCHES: VECTOR[4,BYTE,UNSIGNED]
;  351          0415  1         PSECT(SOR$RO_CODE) PRESET(
;  352          0416  1             [K_U] = OPC_BLEQU,  [K_U+1] = OPC_BGEQU,
;  353          0417  1             [K_S] = OPC_BLEQ,   [K_S+1] = OPC_BGEQ);
;  354          0418  1
```

```
;  355          0419  1 MACRO
;  356          0420  1     SAVED_REGS = BRANCH[0] %,
;  357          0421  1     BR_D_(X) = BRANCH[1+(X)] %, ! Address for a direct branch
;  358          0422  1     BR_M_(X) = BRANCH[5+(X)] %, ! Mask of restored registers
;  359          0423  1     BR_I_(X) = BRANCH[9+(X)] %; ! Address for an indirect branch
;  360          0424  1
;  361          0425  1 LITERAL
;  362          0426  1     BR_SIZE = 1+3*4;           ! Size in longwords of branches array
```

```
;  364     0427  1 ROUTINE TKS_HACK
;  365     0428  1         (
;  366     0429  1         KEY_BUFF:       REF KEY_BLOCK
;  367     0430  1         ):      CAL_CTXREG NOVALUE =
;  368     0431  1 !++
;  369     0432  1 ! Functional Description:
;  370     0433  1 !
;  371     0434  1 !       This routine modifies the key description buffer to account for the
;  372     0435  1 !       fact that keys are being stripped.
;  373     0436  1 !
;  374     0437  1 !       The number of bytes to strip is computed from the key descriptions,
;  375     0438  1 !       or specified by the user as the "total key size" parameter,
;  376     0439  1 !       depending.
;  377     0440  1 !
;  378     0441  1 !       Key stripping and key prefixing are known as the infamous TKS hack.
;  379     0442  1 !
;  380     0443  1 ! Formal Parameters:
;  381     0444  1 !
;  382     0445  1 !       KEY_BUFF        Address of DSC format key descriptions.
;  383     0446  1 !                       The descriptions may be modified by this routiine.
;  384     0447  1 !
;  385     0448  1 ! Implicit Inputs:
;  386     0449  1 !
;  387     0450  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  388     0451  1 !
;  389     0452  1 ! Implicit Outputs:
;  390     0453  1 !
;  391     0454  1 !       CTX[COM_TKS]    Number of bytes to strip before calling user-written
;  392     0455  1 !                       routines, and before returning the record.
;  393     0456  1 !       CTX[COM_LRL]    Advanced by the number of bytes we are stripping.
;  394     0457  1 !
;  395     0458  1 ! Routine Value:
;  396     0459  1 !
;  397     0460  1 !       None (may signal errors).
;  398     0461  1 !
;  399     0462  1 ! Side Effects:
;  400     0463  1 !
;  401     0464  1 !       None.
;  402     0465  1 !
;  403     0466  1 !--
;  404     0467  2     BEGIN
;  405     0468  2     EXTERNAL REGISTER
;  406     0469  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK;
;  407     0470  2
;  408     0471  2     IF NOT .CTX[COM_HACK_STRIP]
;  409     0472  2     THEN
;  410     0473  3         BEGIN
;  411     0474  3         !
;  412     0475  3         ! We were requested to not do this hack.
;  413     0476  3         !
;  414     0477  3         CTX[COM_TKS] = 0;
;  415     0478  3         RETURN;
;  416     0479  2         END;
;  417     0480  2
;  418     0481  2
;  419     0482  2     ! If the user-comparison routine is being used, strip as many bytes as
;  420     0483  2     ! were specified by the user (TOT_KEY_SIZE parameter).
```

```
;  421          0484  2     !
;  422          0485  2     ! If we are generating our own key comparison routine,
;  423          0486  2     !   and the record interface is being used (on input),
;  424          0487  2     ! then use KEY_BUFF to calculate the number of bytes to strip,
;  425          0488  2     ! otherwise, don't strip any keys (set COM_TKS to zero).
;  426          0489  2     !
;  427          0490  2     IF .CTX[COM_COMPARE] NEQ 0  ! His own comparison routine?
;  428          0491  2     THEN
;  429          0492  2         0                       ! Don't change COM_TKS
;  430          0493  2     ELIF
;  431          0494  2         .CTX[COM_NUM_FILES] NEQ 0
;  432          0495  2     THEN
;  433          0496  2         CTX[COM_TKS] = 0        ! File interface, don't strip keys
;  434          0497  2     ELSE
;  435          0498  3         BEGIN
;  436          0499  3         CTX[COM_TKS] = 0;
;  437          0500  3         INCR I FROM 0 TO .KEY_BUFF[KEY_NUMBER]-1 DO
;  438          0501  4             BEGIN
;  439          0502  4             LOCAL
;  440          0503  4                 KBF:    REF KBF_BLOCK;  ! Pointer to the key description
;  441          0504  4
;  442          0505  4             ! Grab a local pointer to the key description buffer
;  443          0506  4             !
;  444          0507  4             KBF = KEY_BUFF[KEY_KBF(.I)];
;  445          0508  4
;  446          0509  4             ! Store the offset to this key
;  447          0510  4             !
;  448          0511  4             KBF[KBF_POSITION] = .CTX[COM_TKS];
;  449          0512  4
;  450          0513  4             ! Note: The old sort didn't allow unconverted decimal keys, we do.
;  451          0514  4             !
;  452          0515  4             CTX[COM_TKS] = .CTX[COM_TKS] + LEN_(KBF[BASE_]);
;  453          0516  4
;  454          0517  3             END;
;  455          0518  2         END;
;  456          0519  2
;  457          0520  2     CTX[COM_LRL] = .CTX[COM_LRL] + .CTX[COM_TKS];
;  458          0521  2
;  459          0522  1     END;


                                                        .TITLE  SOR$KEY_SUB
                                                        .IDENT  \V04-000\

                                                        .PSECT  SOR$RO_CODE_____________________2,NOWRT,  SHR,  PIC,

                                00000000V 00000 _CLEAN_UP:
                                                        .LONG   <CLEAN_UP-_CLEAN_UP>                            ;

                                                        .PSECT  SOR$RO_CODE,NOWRT,  SHR,  PIC,2

                                      FF  00000 DSC_LENGTH:
                                                        .BYTE   -1                                              ;
                                      00  00001         .BYTE   0                                               ;
                  08  04  08  04  02  01  08  04  02  01  00002         .BYTE   1, 2, 4, 8, 1, 2, 4, 8, 4, 8    ;
                                      00# 0000C         .BYTE   0[2]                                            ;
                      1F  1F  1F  20  1F  20  1F  FF  0000E         .BYTE   -1, 31, 32, 31, 32, 31, 31, 31  ;
```

```
                                00# 00016          .BYTE   0[3]
                            10  10  00019          .BYTE   16, 16, 8, 16                   ;
                            00# 0001D          .BYTE   0[6]                            ;
                            08  00023          .BYTE   8                               ;
                        08  1E  00  0F  FC  00024 DSC_FORCE:
                                            .BYTE   -4, 15, 0, 30, 8                ;
                                00029          .BLKB   3
                        08  06  00  03  FC  0002C DSC_BINARY:
                                            .BYTE   -4, 3, 0, 6, 8                  ;
                                00031          .BLKB   3
                            18  15  1E  1B  00034 OPC_BRANCHES:
                                            .BYTE   27, 30, 21, 24                  ;

                                            .EXTRN  SOR$$ERROR, SOR$$RDT
                                            .EXTRN  SOR$$RFA_ACCESS
                                            .EXTRN  SOR$$ALLOCATE, SOR$$DEALLOCATE
                                            .EXTRN  LIB$AB_CVTTP_O, LIB$AB_CVTTP_U
                                            .EXTRN  LIB$AB_CVTTP_Z
                                            .WEAK   FUN_K_RANJI

                                    000C 00000 TKS_HACK:
                                            .WORD   Save R2,R3                      ; 0427
04      5C  AB                      06  E0 00002          BBS     #6, 92(CTX), 1$         ; 0471
                            78      AB  94 00007          CLRB    120(CTX)                ; 0477
                                        04 0000A          RET                             ; 0473
                                    6B  D5 0000B 1$:      TSTL    (CTX)                   ; 0490
                                    3F  12 0000D          BNEQ    7$                      ;
                            52      78  AB 9E 0000F      MOVAB   120(CTX), R2            ; 0496
                                    59  AB 95 00013      TSTB    89(CTX)                 ; 0494
                                        04 13 00016       BEQL    2$                      ;
                                        62 94 00018       CLRB    (R2)                    ; 0496
                                        32 11 0001A       BRB     7$                      ;
                                        62 94 0001C 2$:   CLRB    (R2)                    ; 0499
                                53      04  BC 3C 0001E      MOVZWL  @KEY_BUFF, R3           ; 0500
                                51          01 CE 00022       MNEGL   #1, I                   ;
                                            23 11 00025       BRB     6$                      ;
                                50      04 BC41 7E 00027 3$:  MOVAQ   @KEY_BUFF[I], KBF       ; 0507
                                50          02 C0 0002C       ADDL2   #2, KBF                 ;
                            04  A0          62 9B 0002F       MOVZBW  (R2), 4(KBF)            ; 0511
                                15          60 B1 00033       CMPW    (KBF), #21              ; 0515
                                            0B 12 00036       BNEQ    4$                      ;
                                50      06  A0 3C 00038       MOVZWL  6(KBF), R0              ;
                                50          02 C6 0003C       DIVL2   #2, R0                  ;
                                            50 D6 0003F       INCL    R0                      ;
                                            04 11 00041       BRB     5$                      ;
                                50      06  A0 3C 00043 4$:   MOVZWL  6(KBF), R0              ;
                                62          50 80 00047 5$:   ADDB2   R0, (R2)                ;
                        D9                  51 F2 0004A 6$:   AOBLSS  R3, I, 3$               ; 0500
                                50      78  AB 9A 0004E 7$:   MOVZBL  120(CTX), R0            ; 0520
                        0084    CB          50 A0 00052       ADDW2   R0, 132(CTX)            ;
                                            04 00057          RET                             ; 0522

; Routine Size:  88 bytes,    Routine Base:  SOR$RO_CODE + 0038
```

```
;  461          0523  1 ROUTINE KEY_COMPRESS
;  462          0524  1         (
;  463          0525  1         KEY_BUFF:       REF KEY_BLOCK
;  464          0526  1         ):      CAL_CTXREG NOVALUE =
;  465          0527  1 !++
;  466          0528  1 ! Functional Description:
;  467          0529  1 !
;  468          0530  1 !       This routine attempts to combine adjacent keys.
;  469          0531  1 !       Additionally, it converts keys to a normalized form.
;  470          0532  1 !
;  471          0533  1 ! Formal Parameters:
;  472          0534  1 !
;  473          0535  1 !       KEY_BUFF        Address of DSC format key descriptions.
;  474          0536  1 !                       The descriptions may be modified by this routiine.
;  475          0537  1 !
;  476          0538  1 ! Implicit Inputs:
;  477          0539  1 !
;  478          0540  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  479          0541  1 !
;  480          0542  1 ! Implicit Outputs:
;  481          0543  1 !
;  482          0544  1 !       None.
;  483          0545  1 !
;  484          0546  1 ! Routine Value:
;  485          0547  1 !
;  486          0548  1 !       None (may signal errors).
;  487          0549  1 !
;  488          0550  1 ! Side Effects:
;  489          0551  1 !
;  490          0552  1 !       None.
;  491          0553  1 !
;  492          0554  1 ! Notes:
;  493          0555  1 !
;  494          0556  1 !       The following datatypes compare bytes in the following order:
;  495          0557  1 !
;  496          0558  1 !       C       u0,u1,u2,...
;  497          0559  1 !       xB      x0
;  498          0560  1 !       xW      x1,u0
;  499          0561  1 !       xL      x3,u2,u1,u0
;  500          0562  1 !       xQ      x7,u6,...,u1,u0
;  501          0563  1 !       xO      x15,u14,...,u1,u0
;  502          0564  1 !
;  503          0565  1 !       The following pairs of adjacent keys can be combined:
;  504          0566  1 !
;  505          0567  1 !       Keys    Conditions              Result
;  506          0568  1 !       C       x.l=1                   Ub(x.a,x.l)
;  507          0569  1 !       C,C     x.a+x.l=y.a             C(x.a,x.l+y.l)
;  508          0570  1 !       C,uB    x.a+x.l=y.a,y.l=1       C(x.a,x.l+y.l)
;  509          0571  1 !       uB,C    x.a+x.l=y.a,x.l=1       C(x.a,x.l+y.l)
;  510          0572  1 !       xb,Ub   x.a=y.a+y.l             xb(y.a,x.l+y.l)
;  511          0573  1 !
;  512          0574  1 !--
;  513          0575  2     BEGIN
;  514          0576  2     EXTERNAL REGISTER
;  515          0577  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK;
;  516          0578  2     0;
;  517          0579  2
```

```
;  518          C 0580  2 %(
;  519          C 0581  2         KEY_NUMBER =    0, 0, 16, 0 %,                  ! Number of keys
;  520          C 0582  2         KEY_KBF(N) =    2 + KBF_K_SIZE * (N), 0, 0, 0 %,
;  521          C 0583  2         KEY_BLOCK =     BLOCK[2 + KBF_K_SIZE * MAX_KEYS, BYTE] %;
;  522          C 0584  2
;  523          C 0585  2     ! For each key, attempt to combine it with following keys
;  524          C 0586  2     !
;  525          C 0587  2     INCR I FROM 0 TO .KEY_BUFF[KEY_NUMBER]-1 DO
;  526          C 0588  2         BEGIN
;  527          C 0589  2         LOCAL
;  528          C 0590  2             KBF1:       REF KBF_BLOCK,  ! Pointer to the key description
;  529          C 0591  2             KBF2:       REF KBF_BLOCK;  ! Pointer to the key description
;  530          C 0592  2
;  531          C 0593  2         ! Grab a local pointer to the key description buffer
;  532          C 0594  2         !
;  533          C 0595  2         KBF1 = KEY_BUFF[KEY_KBF(.I)];
;  534          C 0596  2
;  535          C 0597  2         ?????
;  536          C 0598  2
;  537          C 0599  2         END;
;  538            0600  2 )%
;  539            0601  1     END;


                     0000 00000 KEY_COMPRESS:
                                        .WORD   Save nothing                    ; 0523
                       04 00002         RET                                     ; 0601

; Routine Size:  3 bytes,    Routine Base:  SOR$RO_CODE + 0090
```

```
;  541          0602  1 ROUTINE COND_HAND (
;  542          0603  1         SIGVEC: REF BLOCK[,BYTE],       ! Signal vector
;  543          0604  1         MCHVEC: REF BLOCK[,BYTE]) =     ! Mechanism vector
;  544          0605  1 !++
;  545          0606  1 ! Functional Description:
;  546          0607  1 !
;  547          0608  1 !       This routine is a condition handler for the x_HARDWARE routines.
;  548          0609  1 !       The x_HARDWARE routines determine whether the x-type instructions
;  549          0610  1 !       are implemented in hardware.
;  550          0611  1 !
;  551          0612  1 ! Formal Parameters:
;  552          0613  1 !
;  553          0614  1 !       SIGVEC  Signal vector
;  554          0615  1 !       MCHVEC  Mechanism vector
;  555          0616  1 !
;  556          0617  1 ! Implicit Inputs:
;  557          0618  1 !
;  558          0619  1 !       None.
;  559          0620  1 !
;  560          0621  1 ! Implicit Outputs:
;  561          0622  1 !
;  562          0623  1 !       None.
;  563          0624  1 !
;  564          0625  1 ! Routine Value:
;  565          0626  1 !
;  566          0627  1 !       If SS$_OPCDEC was signalled, unwind the stack so that the x_HARDWARE
;  567          0628  1 !       routine returns FALSE (not implemented in hardware).
;  568          0629  1 !       If any other error was signalled, return SS$_RESIGNAL.
;  569          0630  1 !
;  570          0631  1 ! Notes:
;  571          0632  1 !
;  572          0633  1 !       The returned value of the x_HARDWARE routines should not affect the
;  573          0634  1 !       correctness of the sort.
;  574          0635  1 !       If TRUE is incorrectly returned, LIB$EMULATE fakes the instructions.
;  575          0636  1 !       If FALSE is incorrectly returned, the key will be converted, so that
;  576          0637  1 !       binary compares will be used.
;  577          0638  1 !
;  578          0639  1 !       The x_HARDWARE routines may be short-circuited by defining the symbol
;  579          0640  1 !       x_FORCE to be the value to return.
;  580          0641  1 !
;  581          0642  1 !       If the instruction succeeds, we then look at SYI$_ARCHFLAG to determine
;  582          0643  1 !       whether the instruction emulator is in use (since this would cause us
;  583          0644  1 !       to not catch the signal).
;  584          0645  1 !
;  585          0646  1 ! Side Effects:
;  586          0647  1 !
;  587          0648  1 !       None.
;  588          0649  1 !
;  589          0650  1 !--
;  590          0651  2     BEGIN
;  591          0652  2     IF .SIGVEC[CHF$L_SIG_NAME] EQL SS$_OPCDEC
;  592          0653  2     THEN
;  593          0654  3         BEGIN
;  594          0655  3         MCHVEC[CHF$L_MCH_SAVR0] = FALSE;
;  595          0656  3         RETURN $UNWIND();
;  596          0657  3         END
;  597          0658  2     ELSE
```

```
;  598          0659  2          RETURN SS$_RESIGNAL;
;  599          0660  1      END;


                                                        .EXTRN  SYS$UNWIND

                                            0000 00000 COND_HAND:
                                                        .WORD   Save nothing                    ; 0602
                          50        04    AC  D0 00002          MOVL    SIGVEC, R0              ; 0652
                0000043C  8F        04    A0  D1 00006          CMPL    4(R0), #1084
                                          11  12 0000E          BNEQ    1$
                          50        08    AC  D0 00010          MOVL    MCHVEC, R0              ; 0655
                                    0C    A0  D4 00014          CLRL    12(R0)
                                          7E  7C 00017          CLRQ    -(SP)                   ; 0656
                00000000G 00              02  FB 00019          CALLS   #2, SYS$UNWIND
                                          04  00020          RET                             ; 0659
                          50        0918  8F  3C 00021 1$:     MOVZWL  #2328, R0
                                          04  00026          RET                             ; 0660

; Routine Size:  39 bytes,    Routine Base:  SOR$RO_CODE + 0093


;  600          0661  1
;  601          0662  1 LINKAGE
;  602          0663  1     JSB0 = JSB: NOPRESERVE(0,1,2,3,4,5) NOTUSED(6,7,8,9,10,11),
;  603          0664  1     JSB1 = JSB(REGISTER=2): GLOBAL(CTX=COM_REG_CTX)
;  604          0665  1                 NOPRESERVE(0,1,2,3,4,5) NOTUSED(6,7,8,9,10);
;  605          0666  1 LITERAL
;  606          0667  2     ARC_NOTUSED = NOT(          ! Mask of bits with no meaning (to us)
;  607          0668  2         ARC$M_CHAR_EMUL OR ARC$M_DCML_EMUL OR ARC$M_EDPC_EMUL OR
;  608          0669  2         ARC$M_CRC_EMUL  OR ARC$M_DFLT_EMUL OR ARC$M_FFLT_EMUL OR
;  609          0670  1         ARC$M_GFLT_EMUL OR ARC$M_HFLT_EMUL);
;  610          0671  1     ASSERT_(ARC_NOTUSED NEQ 0)  ! Assert there are some unused bits
;  611          0672  1
;  612          0673  1
;  613          0674  1 ROUTINE ARCHFLAG(P): JSB1 =
;  614          0675  2     BEGIN
;  615          0676  2     EXTERNAL REGISTER
;  616          0677  2         CTX = COM_REG_CTX:      REF CTX_BLOCK;
;  617          0678  2
;  618          0679  2     ! Have we gotten SYI$_ARCHFLAG before?
;  619          0680  2     !
;  620          0681  2     IF .CTX[COM_ARCHFLAG] EQL 0
;  621          0682  2     THEN
;  622          0683  3         BEGIN
;  623          0684  3         !
;  624          0685  3         ! Call $GETSYI, and then indicate that we've gotten SYI$_ARCHFLAG
;  625          0686  3         !
;  626          0687  3         ASSERT_(%FIELDEXPAND(COM_ARCHFLAG,2) GEQ ARC$S_ARCDEF * %BPUNIT)
;  627          0688  3         LOCAL
;  628          0689  3             ITMLST: VECTOR[4] INITIAL
;  629          0690  3                 (SYI$_ARCHFLAG ^ 16 + ARC$S_ARCDEF, CTX[COM_ARCHFLAG], 0, 0);
;  630          0691  3         $GETSYIW(ITMLST=ITMLST[0]);     ! On errors COM_ARCHFLAG is still zero
;  631          0692  3         CTX[COM_ARCHFLAG] = .CTX[COM_ARCHFLAG] OR ARC_NOTUSED;
;  632          0693  2         END;
;  633          0694  2
```

```
;  634          0695  2      ! Return the value of the ARC$V_xxx_EMUL flag
;  635          0696  2      !
;  636          0697  2 !    IF      .BITVECTOR[UPLIT(ARC_NOTUSED),.P] THEN SOR$$ERROR(SOR$_SHR_BADLOGIC);
;  637          0698  2      RETURN .BITVECTOR[CTX[COM_ARCHFLAG], .P];
;  638          0699  1      END;


                                                        000BA          .BLKB   2
                                                10DA0004  000BC P.AAA:  .LONG   282722308
                                00000000  00000000  00000000  000C0          .LONG   0, 0, 0                                 ;

                                                                       .EXTRN  SYS$GETSYIW

                                        5E              10  C2 00000 ARCHFLAG:
                                                                       SUBL2   #16, SP                                 ; 0674
                                                        52  DD 00003          PUSHL   R2
                                                  014C  CB  9F 00005          PUSHAB  332(CTX)                        ; 0681
                                                    00  BE  D5 00009          TSTL    @0(SP)
                                                        24  12 0000C          BNEQ    1$
                        08  AE          DE  AF             10  28 0000E          MOVC3   #16, P.AAA, ITMLST              ; 0690
                                         0C  AE             6E  D0 00014          MOVL    (SP), ITMLST+4
                                                            7E  7C 00018          CLRQ    -(SP)                           ; 0691
                                                            7E  D4 0001A          CLRL    -(SP)
                                                      14    AE  9F 0001C          PUSHAB  ITMLST
                                                            7E  7C 0001F          CLRQ    -(SP)
                                                            7E  D4 00021          CLRL    -(SP)
                                00000000G   00              07  FB 00023          CALLS   #7, SYS$GETSYIW
                                        00  BE  FFFFF00F     8F  C8 0002A          BISL2   #-4081, @0(SP)                  ; 0692
                50              9E              01      04  AE  EF 00032 1$:      EXTZV   P, #1, @(SP)+, R0               ; 0698
                                        5E              14  C0 00038          ADDL2   #20, SP                         ; 0699
                                                        05  0003B          RSB                                             ;

; Routine Size:  60 bytes,    Routine Base:  SOR$RO_CODE + 00CC

;  639          0700  1
;  640          0701  1 MACRO
;  641          0702  1     P_(O,P,S,E) = P %;
;  642          0703  1 MACRO
;  643        M 0704  1     X_HARDWARE(A) =
;  644        M 0705  1         BEGIN
;  645        M 0706  1         !
;  646        M 0707  1         ! Return true if hardware support exists
;  647        M 0708  1         ! Return false otherwise
;  648        M 0709  1         !
;  649        M 0710  1         EXTERNAL REGISTER
;  650        M 0711  1             CTX = COM_REG_CTX:  REF CTX_BLOCK;
;  651        M 0712  1         %IF %DECLARED(%NAME(A,'_FORCE'))
;  652        M 0713  1         %THEN
;  653        M 0714  1             RETURN %NAME(A,'_FORCE')
;  654        M 0715  1         %ELSE
;  655        M 0716  1             BEGIN
;  656        M 0717  1             !
;  657        M 0718  1             ! Establish a condition handler so that if we catch an "opcode
;  658        M 0719  1             ! reserved to Digital" signal, we will assume no hardware support.
;  659        M 0720  1             !
```

```
;  660        M 0721  1              ESTABLISH_(COND_HAND);
;  661        M 0722  1              !
;  662        M 0723  1              ! Try the instruction.
;  663        M 0724  1              !
;  664        M 0725  1              JSB0(UPLIT BYTE(%REMAINING, OPC_RSB));
;  665        M 0726  1              !
;  666        M 0727  1              ! We got here.  The instruction is either implemented in hardware
;  667        M 0728  1              ! or emulated (so that we couldn't catch a signal).
;  668        M 0729  1              ! If the system claims it is emulating, assume no hardware support.
;  669        M 0730  1              ! If it claims no emulation, assume the hardware got us here.
;  670        M 0731  1              !
;  671        M 0732  1              ASSERT_((ARC_NOTUSED AND %NAME('ARC$M_',A,'_EMUL')) EQL 0)
;  672        M 0733  1              RETURN NOT ARCHFLAG( P_( %NAME('ARC$V_',A,'_EMUL') ) );
;  673        M 0734  1              END
;  674        M 0735  1          %FI
;  675          0736  1          END %;
;  676          0737  1
;  677          0738  1 !        CHAR_HARDWARE:CAL_CTXREG = X_HARDWARE('CHAR', OPC_CMPC3, 0, %X'0C', %X'0C');
;  678          0739  1 !        EDPC_HARDWARE:CAL_CTXREG = X_HARDWARE('EDPC', OPC_BPT);
;  679          0740  1 !        CRC_HARDWARE :CAL_CTXREG = X_HARDWARE('CRC',  OPC_BPT);
;  680          0741  1 !        DCML_HARDWARE:CAL_CTXREG = X_HARDWARE('DCML', OPC_CMPP3, 0, %X'0C', %X'0C');
;  681          0742  1 ROUTINE FFLT_HARDWARE:CAL_CTXREG = X_HARDWARE('FFLT', OPC_CMPF, 0, 0);


                                05  00  00  51  00108 P.AAB:  .BYTE   81, 0, 0, 5                                ;


                                                    003C 00000 FFLT_HARDWARE:
                                                               .WORD   Save R2,R3,R4,R5                          ; 0742
                                6D        82    AF  9E 00002         MOVAB   COND_HAND, (FP)                     ;
                                                F4  10 00006         BSBB    P.AAB                               ;
                                52              09  D0 00008         MOVL    #9, R2                              ;
                                                B3  10 0000B         BSBB    ARCHFLAG                            ;
                                50              50  D2 0000D         MCOML   R0, R0                              ;
                                                    04 00010         RET                                         ;

; Routine Size:  17 bytes,    Routine Base:  SOR$RO_CODE + 010C

;  682          0743  1 ROUTINE DFLT_HARDWARE:CAL_CTXREG = X_HARDWARE('DFLT', OPC_CMPD, 0, 0);


                                     05  00  00  71  0011D P.AAC:  .BYTE   113, 0, 0, 5                          ;


                                                    003C 00000 DFLT_HARDWARE:
                                                               .WORD   Save R2,R3,R4,R5                          ; 0743
                                6D      FF6C    CF  9E 00002         MOVAB   COND_HAND, (FP)                     ;
                                                F3  10 00007         BSBB    P.AAC                               ;
                                52              08  D0 00009         MOVL    #8, R2                              ;
                                                9D  10 0000C         BSBB    ARCHFLAG                            ;
                                50              50  D2 0000E         MCOML   R0, R0                              ;
                                                    04 00011         RET                                         ;
```

```
; Routine Size:  18 bytes,    Routine Base:  SOR$RO_CODE + 0121

;  683          0744  1 ROUTINE GFLT_HARDWARE:CAL_CTXREG = X_HARDWARE('GFLT', WORD(OPC_CMPG), 0, 0);

                                        51FD  00133 P.AAD:  .WORD   20989                                    ;
                                  05 00   00  00135         .BYTE   0, 0, 5                                  ;


                                            003C 00000 GFLT_HARDWARE:
                                                       .WORD   Save R2,R3,R4,R5                              ; 0744
                              6D    FF55    CF  9E 00002         MOVAB   COND_HAND, (FP)                      ;
                                            F2  10 00007         BSBB    P.AAD                                ;
                              52            0A  D0 00009         MOVL    #10, R2                              ;
                                            86  10 0000C         BSBB    ARCHFLAG                             ;
                              50            50  D2 0000E         MCOML   R0, R0                               ;
                                                04 00011         RET                                          ;

; Routine Size:  18 bytes,    Routine Base:  SOR$RO_CODE + 0138

;  684          0745  1 ROUTINE HFLT_HARDWARE:CAL_CTXREG = X_HARDWARE('HFLT', WORD(OPC_CMPH), 0, 0);


                                        71FD  0014A P.AAE:  .WORD   29181                                    ;
                                  05 00   00  0014C         .BYTE   0, 0, 5                                  ;


                                            003C 00000 HFLT_HARDWARE:
                                                       .WORD   Save R2,R3,R4,R5                              ; 0745
                              6D    FF3E    CF  9E 00002         MOVAB   COND_HAND, (FP)                      ;
                                            F2  10 00007         BSBB    P.AAE                                ;
                              52            0B  D0 00009         MOVL    #11, R2                              ;
                                          FF6E  30 0000C         BSBW    ARCHFLAG                             ;
                              50            50  D2 0000F         MCOML   R0, R0                               ;
                                                04 00012         RET                                          ;

; Routine Size:  19 bytes,    Routine Base:  SOR$RO_CODE + 014F

;  685          0746  1
;  686          0747  1 ASSERT_(DSC$K_DTYPE_F MOD 5 EQL 0)
;  687          0748  1 ASSERT_(DSC$K_DTYPE_D MOD 5 EQL 1)
;  688          0749  1 ASSERT_(DSC$K_DTYPE_G MOD 5 EQL 2)
;  689          0750  1 ASSERT_(DSC$K_DTYPE_H MOD 5 EQL 3)
;  690          0751  1 MACRO
;  691        M 0752  1     FDGH_HARDWARE_(DTY) = (.VECTOR[UPLIT_BYTE(
;  692        M 0753  1         FFLT_HARDWARE - FFLT_HARDWARE, DFLT_HARDWARE - FFLT_HARDWARE,
;  693        M 0754  1         GFLT_HARDWARE - FFLT_HARDWARE, HFLT_HARDWARE - FFLT_HARDWARE),
;  694          0755  1         (DTY) MOD 5;,BYTE] + FFLT_HARDWARE)() %;
```

```
;  696          0756  1 ROUTINE DO_REI: NOVALUE =
;  697          0757  2     BEGIN
;  698          0758  2     !
;  699          0759  2     ! This little routine executes an REI instruction.  This is the only
;  700          0760  2     ! architecturally defined way to ensure that code which was written by
;  701          0761  2     ! a program is actually available before the instruction prefetch.
;  702          0762  2     !
;  703          0763  2     LINKAGE LINK_REI = INTERRUPT: NOTUSED(2,3,4,5,6,7,8,9,10,11);
;  704          0764  2     ROUTINE REI( RETPC, RETPSL ): LINK_REI = 0;



                                                02 00000 REI:    REI                                    ; 0764

; Routine Size:  1 bytes,    Routine Base:  SOR$RO_CODE + 0162

;  705          0765  2     LOCAL NEWPSL;
;  706          0766  2     BUILTIN MOVPSL;
;  707          0767  2     MOVPSL(NEWPSL);
;  708          0768  2     REI(.NEWPSL);
;  709          0769  1     END;



                                          0000 00000 DO_REI: .WORD   Save nothing                       ; 0756
                                    50    DC 00002           MOVPSL  NEWPSL                           ; 0767
                                    50    DD 00004           PUSHL   NEWPSL                           ; 0768
                                    F7    10 00006           BSBB    REI                              ;
                                       04 00008           RET                                      ; 0769

; Routine Size:  9 bytes,    Routine Base:  SOR$RO_CODE + 0163
```

```
1 LITERAL K_SAVE_REGS = 3;                                      ! Max bytes from this routine
1 ROUTINE SAVE_REGS(S): LINK_SAVE NOVALUE =
2     BEGIN
2     EXTERNAL REGISTER
2         CUR_PC = R_CUR_PC:          REF BLOCK,
2         BRANCH = R_BRANCH:          REF VECTOR;
2     LOCAL
2         M;                                              ! Registers not currently saved
2
2     ! Don't bother saving registers that don't need to be preserved for the
2     ! linkage to the comparison routine.  Don't bother saving registers that
2     ! have already been saved.  We must, however, save registers in increasing
2     ! order (because that's the way PUSHR and POPR work, and the only info we
2     ! keep is the mask of registers saved, not the order we saved them).
2     !
2     M = .S AND NOT .SAVED_REGS AND NOT %NOPRESERVE(JSB_COMPARE);
2     IF .M EQL 0 THEN RETURN;
2
2     ! We need to save some more registers.  If (for example), we are saving R8,
2     ! we must also save R0 through R7 (unless they've have already been saved,
2     ! or don't need to be saved for this linkage).
2     !
2     M = .M OR .M^-1;
2     M = .M OR .M^-2;
2     M = .M OR .M^-4;
2     M = .M OR .M^-8;
2     M = .M AND NOT .SAVED_REGS AND NOT %NOPRESERVE(JSB_COMPARE);
2
2     ! Check that the save mask fits in a short literal.
2     !
2     IF .M GTRU SHORT_LIT THEN SOR$$ERROR(SOR$_SHR_BADLOGIC);
2
2     ! Generate code to store the registers on the stack, and update the
2     ! register save mask.
2     !
2     EMIT_BYTES(OPC_PUSHR, .M);
2     SAVED_REGS = .SAVED_REGS OR .M;
2
1     END;
```

```
                                  54 DD 00000 SAVE_REGS:
                                                PUSHL   R4                          ; 0771
                 54               69 CA 00002   BICL2   (BRANCH), R4                ; 0785
                 54               3F 8A 00005   BICB2   #63, M                      ;
                                  54 D5 00008   TSTL    M                           ; 0786
                                  47 13 0000A   BEQL    2$                          ;
       50        54      FF       8F 78 0000C   ASHL    #-1, M, R0                  ; 0792
                 54               50 C8 00011   BISL2   R0, M                       ;
       50        54      FE       8F 78 00014   ASHL    #-2, M, R0                  ; 0793
                 54               50 C8 00019   BISL2   R0, M                       ;
       50        54      FC       8F 78 0001C   ASHL    #-4, M, R0                  ; 0794
                 54               50 C8 00021   BISL2   R0, M                       ;
       50        54      F8       8F 78 00024   ASHL    #-8, M, R0                  ; 0795
```

```
                        54              50  C8 00029          BISL2   R0, M                                   ;
                50      54              69  CB 0002C          BICL3   (BRANCH), M, R0                         ; 0796
                54      50              3F  CB 00030          BICL3   #63, R0, M                              ;
                        3F              54  D1 00034          CMPL    M, #63                                  ; 0800
                                        0D  1B 00037          BLEQU   1$                                      ;
                                001C1124    8F  DD 00039          PUSHL   #1839396                                ;
            00000000G   00              01  FB 0003F          CALLS   #1, SOR$$ERROR                          ;
                50      54              08  78 00046 1$:      ASHL    #8, M, R0                               ; 0805
                8A      50      00BB    8F  A1 0004A          ADDW3   #187, R0, (CUR_PC)+                     ;
                        69              54  C8 00050          BISL2   M, (BRANCH)                             ; 0806
                                        10  BA 00053 2$:      POPR    #^M<R4>                                 ; 0808
                                            05 00055          RSB                                             ;

; Routine Size:  86 bytes,    Routine Base:  SOR$RO_CODE + 016C
```

```
;  751          0809  1 LITERAL K_BNEQ = 18;                                  ! Max bytes from this routine
;  752          0810  1 ROUTINE EMIT_BNEQ(DST): LINK_BNEQ NOVALUE =
;  753          0811  1 !+
;  754          0812  1 !   Emit a BNEQ instruction
;  755          0813  1 !-
;  756          0814  2     BEGIN
;  757          0815  2     EXTERNAL REGISTER
;  758          0816  2         CUR_PC = R_CUR_PC:      REF BLOCK,
;  759          0817  2         BRANCH = R_BRANCH:      REF VECTOR,
;  760          0818  2         CTX = COM_REG_CTX:      REF CTX_BLOCK;
;  761          0819  2
;  762          0820  2     ! See whether we have a definition for the desired label.
;  763          0821  2     !
;  764          0822  2     IF  .BR_D_(.DST) LSS 0 OR
;  765          0823  2         .BR_M_(.DST) NEQ .SAVED_REGS
;  766          0824  2     THEN
;  767          0825  3         BEGIN
;  768          0826  3         !
;  769          0827  3         ! No label defined yet.  Generate code.
;  770          0828  3         !
;  771          0829  3         LOCAL
;  772          0830  3             TMP: REF VECTOR[,BYTE];
;  773          0831  3         BR_D_(.DST) = BR_I_(.DST) = .CUR_PC - .CTX[S_START];
;  774          0832  3         BR_M_(.DST) = .SAVED_REGS;
;  775          0833  3         ASSERT_(K_BNEQ GEQ 2+2+4+2+2+4)
;  776          0834  3         EMIT_BYTES(.OPC_BRANCHES[.DST],0);
;  777          0835  3         TMP = .CUR_PC;
;  778          0836  3         IF .SAVED_REGS NEQ 0 THEN EMIT_BYTES(OPC_POPR, .SAVED_REGS);
;  779          0837  3         EMIT_BYTES(OPC_MOVL, 1, M_R+R_0, OPC_RSB);
;  780          0838  3         TMP[-1] = .CUR_PC - .TMP;
;  781          0839  3         EMIT_BYTES(OPC_BEQL, 0);
;  782          0840  3         TMP = .CUR_PC;
;  783          0841  3         IF .SAVED_REGS NEQ 0 THEN EMIT_BYTES(OPC_POPR, .SAVED_REGS);
;  784          0842  3         EMIT_BYTES(OPC_MNEGL, 1, M_R+R_0, OPC_RSB);
;  785          0843  3         TMP[-1] = .CUR_PC - .TMP;
;  786          0844  3         END
;  787          0845  2     ELSE
;  788          0846  3         BEGIN
;  789          0847  3         ASSERT_(K_BNEQ GEQ 5)
;  790          0848  3         !
;  791          0849  3         ! The code exists, we just have to get there.
;  792          0850  3         !
;  793          0851  3         LOCAL
;  794          0852  3             Z;
;  795          0853  3         Z = .BR_D_(.DST) + .CTX[S_START] - .CUR_PC - 2; ! Branch displacement
;  796          0854  3         IF
;  797          0855  4             BEGIN
;  798          0856  4             IF .Z<0,8,1> EQL .Z                 ! Will branch byte suffice?
;  799          0857  4             THEN
;  800          0858  4                 TRUE
;  801          0859  4             ELSE
;  802          0860  5                 BEGIN
;  803          0861  5                 LOCAL
;  804          0862  5                     T: REF VECTOR[,BYTE,SIGNED];
;  805          0863  5                 T = .BR_I_(.DST) + .CTX[S_START];
;  806          0864  5                 Z = .T - .CUR_PC - 2;
;  807          0865  5                 IF .Z<0,8,1> EQL .Z             ! Can we branch to a branch?
```

```
;  808          0866  5                THEN
;  809          0867  6                    BEGIN
;  810          0868  6                    !
;  811          0869  6                    ! Try a little branch chaining
;  812          0870  6                    !
;  813          0871  6                    IF .T[0] EQL OPC_BNEQ
;  814          0872  6                    THEN
;  815          0873  7                        BEGIN
;  816          0874  7                        T = .T + .T[1] - .CUR_PC;
;  817          0875  7                        IF .T<0,8,1> EQL .T THEN Z = .T;
;  818          0876  6                        END;
;  819          0877  6                    TRUE
;  820          0878  6                    END
;  821          0879  5                ELSE
;  822          0880  5                    FALSE
;  823          0881  5                END
;  824          0882  4            END
;  825          0883  3        THEN
;  826          0884  4            BEGIN
;  827          0885  4            BR_I_(.DST) = .CUR_PC - .CTX[S_START];      ! Save indirect branch address
;  828          0886  4            EMIT_BYTE(OPC_BNEQ);
;  829          0887  4            EMIT_BYTE(.Z);
;  830          0888  4            END
;  831          0889  3        ELSE
;  832          0890  4            BEGIN
;  833          0891  4            EMIT_BYTES(OPC_BEQL, 3);
;  834          0892  4            BR_I_(.DST) = .CUR_PC - .CTX[S_START];      ! Save indirect branch address
;  835          0893  4            EMIT_BYTE(OPC_BRW);
;  836          0894  4            EMIT_WORD(.Z);                              ! Branch to final destination
;  837          0895  3            END;
;  838          0896  2        END;
;  839          0897  1    END;


                                    54  DD 00000 EMIT_BNEQ:
                                                PUSHL   R4                                          ; 0810
                          5E            04  C2 00002          SUBL2   #4, SP                       ;
                                    24 A944 DF 00005          PUSHAL  36(BRANCH)[DST]              ; 0831
                          50        04 A944 D0 00009          MOVL    4(BRANCH)[DST], R0           ; 0822
                                    07  19 0000E              BLSS    1$                           ;
                          69        14 A944 D1 00010          CMPL    20(BRANCH)[DST], (BRANCH)    ; 0823
                                    5E  13 00015              BEQL    4$                           ;
                    50              5A  1C  AB C3 00017 1$:   SUBL3   28(CTX), CUR_PC, R0          ; 0831
                          00        BE      50 D0 0001C       MOVL    R0, @0(SP)                   ;
                          04 A944           50 D0 00020       MOVL    R0, 4(BRANCH)[DST]           ;
                          14 A944           69 D0 00025       MOVL    (BRANCH), 20(BRANCH)[DST]    ; 0832
                             8A     FE43 CF44 9B 0002A        MOVZBW  OPC_BRANCHES[DST], (CUR_PC)+ ; 0834
                             50        5A  D0 00030           MOVL    CUR_PC, TMP                  ; 0835
                                    04  AE  D4 00033          CLRL    4(SP)                        ; 0836
                                        69  D5 00036          TSTL    (BRANCH)                     ;
                                        0D  13 00038          BEQL    2$                           ;
                                    04  AE  D6 0003A          INCL    4(SP)                        ;
                    54                  69      08  78 0003D ASHL    #8, (BRANCH), R4             ;
                    8A                  54  00BA  8F  A1 00041 ADDW3   #186, R4, (CUR_PC)+         ;
```

```
                                8A 055001D0    8F   D0 00047 2$:     MOVL    #89129424, (CUR_PC)+                    ; 0837
            FF  A0              5A             50   83 0004E         SUBB3   TMP, CUR_PC, -1(TMP)                    ; 0838
                                8A             13   B0 00053         MOVW    #19, (CUR_PC)+                          ; 0839
                                50             5A   D0 00056         MOVL    CUR_PC, TMP                             ; 0840
                                0A       04    AE   E9 00059         BLBC    4(SP), 3$                               ; 0841
                54              69             08   78 0005D         ASHL    #8, (BRANCH), R4                        ;
                8A              54       00BA  8F   A1 00061         ADDW3   #186, R4, (CUR_PC)+                     ;
                                8A 055001CE    8F   D0 00067 3$:     MOVL    #89129422, (CUR_PC)+                    ; 0842
            FF  A0              5A             50   83 0006E         SUBB3   TMP, CUR_PC, -1(TMP)                    ; 0843
                                               67   11 00073         BRB     7$                                      ; 0822
                                50       1C    AB   C0 00075 4$:     ADDL2   28(CTX), R0                             ; 0853
                                50             5A   C2 00079         SUBL2   CUR_PC, R0                              ;
                                50             02   C2 0007C         SUBL2   #2, Z                                   ;
        50                      50             08   00   EC 0007F    CMPV    #0, #8, Z, Z                            ; 0856
                                               37   13 00084         BEQL    5$                                      ;
                54      24 A944         1C     AB   C1 00086         ADDL3   28(CTX), 36(BRANCH)[DST], T              ; 0863
            04  AE              54             5A   C3 0008D         SUBL3   CUR_PC, T, 4(SP)                        ; 0864
                50      04  AE                 02   C3 00092         SUBL3   #2, 4(SP), Z                            ;
        50              50              08     00   EC 00097         CMPV    #0, #8, Z, Z                            ; 0865
                                               2D   12 0009C         BNEQ    6$                                      ;
                                12             64   91 0009E         CMPB    (T), #18                                ; 0871
                                               1A   12 000A1         BNEQ    5$                                      ;
                04  AE              01         A4   98 000A3         CVTBL   1(T), 4(SP)                             ; 0874
                04  AE              04 BE44    9E   000A8         MOVAB   @4(SP)[T], 4(SP)                        ;
                54      04  AE                 5A   C3 000AE         SUBL3   CUR_PC, 4(SP), T                        ;
        54              54              08     00   EC 000B3         CMPV    #0, #8, T, T                            ; 0875
                                               03   12 000B8         BNEQ    5$                                      ;
                                50             54   D0 000BA         MOVL    T, Z                                    ;
            00  BE              5A         1C  AB   C3 000BD 5$:     SUBL3   28(CTX), CUR_PC, @0(SP)                 ; 0885
                                8A             12   90 000C3         MOVB    #18, (CUR_PC)+                          ; 0886
                                8A             50   90 000C6         MOVB    Z, (CUR_PC)+                            ; 0887
                                               11   11 000C9         BRB     7$                                      ; 0854
                                8A       0313  8F   B0 000CB 6$:     MOVW    #787, (CUR_PC)+                         ; 0891
                00  BE          5A         1C  AB   C3 000D0         SUBL3   28(CTX), CUR_PC, @0(SP)                 ; 0892
                                8A             31   90 000D6         MOVB    #49, (CUR_PC)+                          ; 0893
                                8A             50   B0 000D9         MOVW    Z, (CUR_PC)+                            ; 0894
                                5E             08   C0 000DC 7$:     ADDL2   #8, SP                                  ; 0897
                                               10   BA 000DF         POPR    #^M<R4>                                 ;
                                                    05 000E1         RSB                                             ;

; Routine Size:  226 bytes,    Routine Base:  SOR$RO_CODE + 01C2
```

```
;  841     0898  1 LITERAL K_DISP = 5;                                  ! Max bytes from this routine
;  842     0899  1 ROUTINE EMIT_DISP(DISP, REG): LINK_DISP NOVALUE =
;  843     0900  1 !+
;  844     0901  1 !   Displacement mode addressing (does not handle PC-relative addressing).
;  845     0902  1 !-
;  846     0903  2     BEGIN
;  847     0904  2     EXTERNAL REGISTER
;  848     0905  2         CUR_PC = R_CUR_PC:      REF BLOCK;
;  849     0906  2     IF
;  850     0907  2         .DISP<0,08,1> EQL .DISP
;  851     0908  2     THEN
;  852     0909  3         BEGIN
;  853     0910  3         IF .DISP EQL 0
;  854     0911  3         THEN
;  855     0912  3             EMIT_BYTE(M_RD+.REG)                    ! (reg)
;  856     0913  3         ELSE
;  857     0914  4             BEGIN
;  858     0915  4             EMIT_BYTE(M_BD+.REG);                   ! ^Bxx(reg)
;  859     0916  4             EMIT_BYTE(.DISP);
;  860     0917  3             END;
;  861     0918  3         END
;  862     0919  2     ELIF
;  863     0920  2         .DISP<0,16,1> EQL .DISP
;  864     0921  2     THEN
;  865     0922  3         BEGIN
;  866     0923  3         EMIT_BYTE(M_WD+.REG);                       ! ^Wxxxx(reg)
;  867     0924  3         EMIT_WORD(.DISP);
;  868     0925  3         END
;  869     0926  2     ELSE
;  870     0927  3         BEGIN
;  871     0928  3         EMIT_BYTE(M_LD+.REG);                       ! ^Lxxxxxxxx(reg)
;  872     0929  3         EMIT_LONG(.DISP);
;  873     0930  2         END;
;  874     0931  1     END;


                                00  EC 00000 EMIT_DISP:
                                              CMPV    #0, #8, DISP, DISP          ; 0907
                                16  12 00005         BNEQ    3$
                                52  D5 00007         TSTL    DISP                ; 0910
                                07  12 00009         BNEQ    1$
                    6A      53  60  8F  81 0000B         ADDB3   #96, REG, (CUR_PC)  ; 0912
                                08  11 00010         BRB     2$
                    8A      53  A0  8F  81 00012 1$:     ADDB3   #160, REG, (CUR_PC)+ ; 0915
                            6A      52  90 00017         MOVB    DISP, (CUR_PC)      ; 0916
                                5A  D6 0001A 2$:     INCL    CUR_PC              ; 0912
                                    05 0001C         RSB                         ; 0906
                52  52      10      00  EC 0001D 3$:     CMPV    #0, #16, DISP, DISP ; 0920
                                09  12 00022         BNEQ    4$
                    8A      53  C0  8F  81 00024         ADDB3   #192, REG, (CUR_PC)+ ; 0923
                            8A      52  B0 00029         MOVW    DISP, (CUR_PC)+     ; 0924
                                    05 0002C         RSB                         ; 0918
                    8A      53  E0  8F  81 0002D 4$:     ADDB3   #224, REG, (CUR_PC)+ ; 0928
                            8A      52  D0 00032         MOVL    DISP, (CUR_PC)+     ; 0929
```

```
                                05 00035        RSB                                     ; 0931

; Routine Size:  54 bytes,    Routine Base:  SOR$RO_CODE + 02A4
```

```
;  876          0932  1 LITERAL K_OPOPNEQ = K_DISP+K_DISP+K_BNEQ;       ! Max bytes from this routine
;  877          0933  1 ROUTINE OPOPNEQ(OFF, DST): LINK_OPOPNEQ NOVALUE =
;  878          0934  2     BEGIN
;  879          0935  2     EXTERNAL REGISTER
;  880          0936  2         CUR_PC = R_CUR_PC:      REF BLOCK,
;  881          0937  2         BRANCH = R_BRANCH:      REF VECTOR,
;  882          0938  2         CTX = COM_REG_CTX:      REF CTX_BLOCK;
;  883          0939  2     BUILTIN
;  884          0940  2         TESTBITCC;
;  885          0941  2
;  886          0942  2     IF TESTBITCC(DST<0,1,0>)            ! Check ascending/descending flag
;  887          0943  2     THEN
;  888          0944  3         BEGIN
;  889          0945  3         EMIT_DISP(.OFF, COM_REG_SRC1);  ! xx(Rsrc1)
;  890          0946  3         EMIT_DISP(.OFF, COM_REG_SRC2);  ! yy(Rsrc2)
;  891          0947  3         END
;  892          0948  2     ELSE
;  893          0949  3         BEGIN
;  894          0950  3         EMIT_DISP(.OFF, COM_REG_SRC2);  ! yy(Rsrc2)
;  895          0951  3         EMIT_DISP(.OFF, COM_REG_SRC1);  ! xx(Rsrc1)
;  896          0952  2         END;
;  897          0953  2     EMIT_BNEQ(.DST);                    ! BNEQ dst
;  898          0954  1     END;


                              18  BB 00000 OPOPNEQ:PUSHR   #^M<R3,R4>                          ; 0933
                    0A        54  00  E4 00002        BBSC    #0, DST, 1$                     ; 0942
                              53  09  D0 00006        MOVL    #9, R3                          ; 0945
                                  BF  10 00009        BSBB    EMIT_DISP                       ;
                              53  0A  D0 0000B        MOVL    #10, R3                         ; 0946
                                  08  11 0000E        BRB     2$                              ;
                              53  0A  D0 00010 1$:    MOVL    #10, R3                         ; 0950
                                  B5  10 00013        BSBB    EMIT_DISP                       ;
                              53  09  D0 00015        MOVL    #9, R3                          ; 0951
                                  B0  10 00018 2$:    BSBB    EMIT_DISP                       ;
                                FECB  30 0001A        BSBW    EMIT_BNEQ                       ; 0953
                                  18  BA 0001D        POPR    #^M<R3,R4>                      ; 0954
                                      05 0001F        RSB                                     ;

; Routine Size:  32 bytes,    Routine Base:  SOR$RO_CODE + 02DA
```

```
;  900          0955  1 LITERAL K_LITE = 5;                                ! Max bytes from this routine
;  901          0956  1 ROUTINE EMIT_LITE(BWL, LIT): LINK_LITE NOVALUE =
;  902          0957  1 !+
;  903          0958  1 !   Literal mode addressing
;  904          0959  1 !-
;  905          0960  2     BEGIN
;  906          0961  2     EXTERNAL REGISTER
;  907          0962  2         CUR_PC = R_CUR_PC:      REF BLOCK;
;  908          0963  2     IF
;  909          0964  2         .LIT LEQU SHORT_LIT
;  910          0965  2     THEN
;  911          0966  2         EMIT_BYTE(.LIT)
;  912          0967  2     ELSE
;  913          0968  3         BEGIN
;  914          0969  3         EMIT_BYTE(M_AI+R_PC);                       ! (PC)+
;  915          0970  3         CUR_PC[0,0,8*.BWL,0] = .LIT;
;  916          0971  3         CUR_PC = .CUR_PC + .BWL;
;  917          0972  2         END;
;  918          0973  1     END;


                                    52  DD 00000 EMIT_LITE:
                                                 PUSHL   R2                                      ; 0956
                          50        52  D0 00002          MOVL    R2, R0                         ;
                          3F        53  D1 00005          CMPL    LIT, #63                       ; 0964
                                    05  1A 00008          BGTRU   1$                             ;
                          8A        53  90 0000A          MOVB    LIT, (CUR_PC)+                 ; 0966
                                    10  11 0000D          BRB     2$                             ;
                          8A   8F   8F  90 0000F 1$:      MOVB    #-113, (CUR_PC)+               ; 0969
                     52   50        03  78 00013          ASHL    #3, BWL, R2                    ; 0970
                6A   52   00        53  F0 00017          INSV    LIT, #0, R2, (CUR_PC)          ;
                          5A        50  C0 0001C          ADDL2   BWL, CUR_PC                    ; 0971
                                    04  BA 0001F 2$:      POPR    #^M<R2>                        ; 0973
                                        05 00021          RSB                                    ;

; Routine Size:  34 bytes,    Routine Base:  SOR$RO_CODE + 02FA
```

```
;  920          0974  1 LITERAL K_CALL4 = 37              ! Max bytes from this routine
;  921          0975  1         %IF FUN_K_STAB %THEN +30 %FI;
;  922          0976  1 ROUTINE EMIT_CALL4
;  923          0977  1         (
;  924          0978  1         U_RTN,                     ! Address of the user routine
;  925          0979  1         DISP:   REF VECTOR         ! Address of field displacements table
;  926          0980  1         ): NOVALUE LINK_COMPARE =
;  927          0981  1 !+
;  928          0982  1 !   Generate a call to a user routine.
;  929          0983  1 !
;  930          0984  1 !   The arguments passed to the user routine are:
;  931          0985  1 !       Address of source1, by reference
;  932          0986  1 !       Address of source2, by reference
;  933          0987  1 !       Length of source1, by reference
;  934          0988  1 !       Length of source2, by reference
;  935          0989  1 !       Address of the user's context longword
;  936          0990  1 !-
;  937          0991  2     BEGIN
;  938          0992  2     EXTERNAL REGISTER
;  939          0993  2         CTX=    COM_REG_CTX:    REF CTX_BLOCK,
;  940          0994  2         CUR_PC = R_CUR_PC:      REF BLOCK;
;  941          0995  2
;  942        P 0996  2     ASSERT (K_CALL4 GEQ
;  943        P 0997  2         1+K_DISP+                          ! Push address of the context longword
;  944        P 0998  2         MAX(1+K_DISP+1+K_DISP,2+2+2)+      ! Push the lengths
;  945        P 0999  2         MAX(4,1+K_DISP+1+K_DISP)+          ! Push the addresses
;  946        U 1000  2         %IF FUN_K_STAB %THEN
;  947        U 1001  2         1+2*K_DISP+2+7+12+                 ! Swap for stable sorts
;  948        P 1002  2         %FI
;  949          1003  2         2+K_ABSA)                          ! The CALL itself
;  950          1004  2
;  951          1005  2
;  952          1006  2     IF NOT .CTX[COM_HACK_2ARGS]
;  953          1007  2     THEN
;  954          1008  3         BEGIN
;  955          1009  3         !
;  956          1010  3         ! Push the address of the user's context longword
;  957          1011  3         !
;  958          1012  3         EMIT_BYTE(OPC_PUSHAB);
;  959          1013  3         EMIT_DISP(%FIELDEXPAND(COM_CTXADR,0)*%UPVAL, COM_REG_CTX);
;  960          1014  3         !
;  961          1015  3         ! Push the addresses of the word lengths
;  962          1016  3         !
;  963          1017  3         IF .DISP[COM_ORD_VAR] GEQ 0
;  964          1018  3         THEN
;  965          1019  4             BEGIN
;  966          1020  4             EMIT_BYTE(OPC_PUSHAB);
;  967          1021  4             EMIT_DISP(.DISP[COM_ORD_VAR], COM_REG_SRC2);
;  968          1022  4             EMIT_BYTE(OPC_PUSHAB);
;  969          1023  4             EMIT_DISP(.DISP[COM_ORD_VAR], COM_REG_SRC1);
;  970          1024  4             END
;  971          1025  3         ELSE
;  972          1026  4             BEGIN
;  973          1027  4             EMIT_BYTES(OPC_PUSHAW, M_AI+R_PC);
;  974          1028  4             EMIT_WORD(.CTX[COM_LRL]-.CTX[COM_TKS]);
;  975          1029  4             EMIT_BYTES(OPC_PUSAL, M_RD+R_SP);
;  976          1030  3             END;
```

```
;  977          1031  2          END;
;  978          1032  2
;  979          1033  2      ! Now push the addresses of the records
;  980          1034  2      !
;  981        L 1035  2      %IF COM_REG_SRC1+1 EQL COM_REG_SRC2
;  982          1036  2      %THEN
;  983          1037  2      IF .DISP[COM_ORD_DATA] EQL 0 AND .CTX[COM_TKS] EQL 0
;  984          1038  2      THEN
;  985        P 1039  2          EMIT_BYTES(OPC_MOVQ, M_R+COM_REG_SRC1,          ! MOVQ  Rsrc1
;  986          1040  3              M_AD+R_SP)                                  !       -(SP)
;  987          1041  2      ELSE
;  988          1042  2      %FI
;  989          1043  3          BEGIN
;  990          1044  3          EMIT_BYTE(OPC_PUSHAB);                          ! PUSHAB
;  991          1045  3          EMIT_DISP(.DISP[COM_ORD_DATA]+.CTX[COM_TKS],    !       n(Rsrc2)
;  992          1046  3                  COM_REG_SRC2);
;  993          1047  3          EMIT_BYTE(OPC_PUSHAB);                          ! PUSHAB
;  994          1048  3          EMIT_DISP(.DISP[COM_ORD_DATA]+.CTX[COM_TKS],    !       n(Rsrc1)
;  995          1049  3                  COM_REG_SRC1);
;  996          1050  2          END;
;  997          1051  2
;  998          1052  2      ! If stable sorts were requested, be sure that we pass the arguments in
;  999          1053  2      ! a stable order!
; 1000          1054  2      !
; 1001        U 1055  2      %IF FUN_K_STAB %THEN
; 1002        U 1056  2      IF .DISP[COM_ORD_STAB] GEQ 0
; 1003        U 1057  2      THEN
; 1004        U 1058  2          BEGIN
; 1005        U 1059  2          LOCAL
; 1006        U 1060  2              TMP: REF VECTOR[,BYTE];
; 1007        U 1061  2          EMIT_BYTE(OPC_CMPL);
; 1008        U 1062  2          EMIT_DISP(.DISP[COM_ORD_STAB], COM_REG_SRC1);
; 1009        U 1063  2          EMIT_DISP(.DISP[COM_ORD_STAB], COM_REG_SRC2);
; 1010        U 1064  2          EMIT_BYTES(OPC_BLEQ, 0);
; 1011        U 1065  2          TMP = .CUR_PC;
; 1012        U 1066  2          EMIT_BYTES(OPC_MOVQ, M_AI+R_SP, M_R+R_0,
; 1013        U 1067  2              OPC_PUSHL, M_R+R_0,
; 1014        U 1068  2              OPC_PUSHL, M_R+R_1);
; 1015        U 1069  2          IF NOT .CTX[COM_HACK_2ARGS] AND .DISP[COM_ORD_VAR] GEQ 0
; 1016        U 1070  2          THEN
; 1017        U 1071  2              EMIT_BYTES(OPC_MOVQ, M_BD+R_SP, 8, M_R+R_0,
; 1018        U 1072  2                  OPC_MOVL, M_R+R_1, M_BD+R_SP, 8,
; 1019        U 1073  2                  OPC_MOVL, M_R+R_0, M_BD+R_SP, 12);
; 1020        U 1074  2          TMP[-1] = .CUR_PC - .TMP;
; 1021        U 1075  2          END;
; 1022          1076  2      %FI
; 1023          1077  2
; 1024          1078  2
; 1025          1079  2      ! Now emit the CALL
; 1026          1080  2      !
; 1027          1081  2      IF NOT .CTX[COM_HACK_2ARGS]
; 1028          1082  2      THEN
; 1029          1083  3          EMIT_BYTES(OPC_CALLS, 5)                        ! CALLS #5
; 1030          1084  2      ELSE
; 1031          1085  2          EMIT_BYTES(OPC_CALLS, 2);                       ! CALLS #2
; 1032          1086  2      EMIT_ABSA(.U_RTN);                                  !       rtn
; 1033          1087  2
```

```
; 1034          1088  1      END;



                                              001C 00000 EMIT_CALL4:
                                                                   .WORD   Save R2,R3,R4                        ; 0976
                        54        83    AF   9E 00002              MOVAB   EMIT_DISP, R4                        ;
          3F       5C   AB              05   E0 00006              BBS     #5, 92(CTX), 2$                      ; 1006
                        8A        9F    8F   90 0000B              MOVB    #-97, (CUR_PC)+                      ; 1012
                        53              0B   D0 0000F              MOVL    #11, R3                              ; 1013
                        52        54    8F   9A 00012              MOVZBL  #84, R2                              ;
                                        64   16 00016              JSB     EMIT_DISP                            ;
                        50        08    AC   D0 00018              MOVL    DISP, R0                             ; 1017
                        52        18    A0   D0 0001C              MOVL    24(R0), R2                           ;
                                        14   19 00020              BLSS    1$                                   ;
                        8A        9F    8F   90 00022              MOVB    #-97, (CUR_PC)+                      ; 1020
                        53              0A   D0 00026              MOVL    #10, R3                              ; 1021
                                        64   16 00029              JSB     EMIT_DISP                            ;
                        8A        9F    8F   90 0002B              MOVB    #-97, (CUR_PC)+                      ; 1022
                        53              09   D0 0002F              MOVL    #9, R3                               ; 1023
                                        64   16 00032              JSB     EMIT_DISP                            ;
                                        14   11 00034              BRB     2$                                   ; 1017
                        8A      8F3F    8F   B0 00036 1$:          MOVW    #-28865, (CUR_PC)+                   ; 1027
                        50        78    AB   9A 0003B              MOVZBL  120(CTX), R0                         ; 1028
          8A     0084   CB              50   A3 0003F              SUBW3   R0, 132(CTX), (CUR_PC)+              ;
                        8A      6EDD    8F   B0 00045              MOVW    #28381, (CUR_PC)+                    ; 1029
                        52        08    AC   D0 0004A 2$:          MOVL    DISP, R2                             ; 1037
                                  20    A2   D5 0004E              TSTL    32(R2)                               ;
                                        10   12 00051              BNEQ    3$                                   ;
                                  78    AB   95 00053              TSTB    120(CTX)                             ;
                                        0B   12 00056              BNEQ    3$                                   ;
                        8A      597D    8F   B0 00058              MOVW    #22909, (CUR_PC)+                    ; 1040
                        8A        7E    8F   90 0005D              MOVB    #126, (CUR_PC)+                      ;
                                        1B   11 00061              BRB     4$                                   ; 1037
                        8A        9F    8F   90 00063 3$:          MOVB    #-97, (CUR_PC)+                      ; 1044
                        50        78    AB   9A 00067              MOVZBL  120(CTX), R0                         ; 1045
          52            50        20    A2   C1 0006B              ADDL3   32(R2), R0, R2                       ;
                        53              0A   D0 00070              MOVL    #10, R3                              ;
                                        64   16 00073              JSB     EMIT_DISP                            ;
                        8A        9F    8F   90 00075              MOVB    #-97, (CUR_PC)+                      ; 1047
                        53              09   D0 00079              MOVL    #9, R3                               ; 1048
                                        64   16 0007C              JSB     EMIT_DISP                            ;
          07       5C   AB              05   E0 0007E 4$:          BBS     #5, 92(CTX), 5$                      ; 1083
                        6A      05FB    8F   B0 00083              MOVW    #1531, (CUR_PC)                      ;
                                        05   11 00088              BRB     6$                                   ;
                        6A      02FB    8F   B0 0008A 5$:          MOVW    #763, (CUR_PC)                       ; 1085
                        5A              02   C0 0008F 6$:          ADDL2   #2, CUR_PC                           ;
                        8A        9F    8F   90 00092              MOVB    #-97, (CUR_PC)+                      ; 1086
                        8A        04    AC   D0 00096              MOVL    U_RTN, (CUR_PC)+                     ;
                                             04 0009A              RET                                          ; 1088

; Routine Size:  155 bytes,    Routine Base:  SOR$RO_CODE + 031C
```

```
;  1036          1089  1 ROUTINE ROOM(SPACE): LINK_ROOM =
;  1037          1090  1 !+
;  1038          1091  1 !   Verify amount of space remaining in string, extending string if needed.
;  1039          1092  1 !-
;  1040          1093  2     BEGIN
;  1041          1094  2     EXTERNAL REGISTER
;  1042          1095  2         CUR_PC = R_CUR_PC:      REF BLOCK,
;  1043          1096  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK;
;  1044          1097  2     BIND
;  1045          1098  2         XCODE = CTX[COM_ROUTINES]: VECTOR[2];
;  1046          1099  2
;  1047          1100  2     LOCAL
;  1048          1101  2         DELTA:  VECTOR[2],
;  1049          1102  2         OLDSTART;
;  1050          1103  2
;  1051          1104  2     ! Determine how much memory we need
;  1052          1105  2     !
;  1053          1106  2     DELTA[0] = .CUR_PC - .XCODE[1] + .SPACE;
;  1054          1107  2
;  1055          1108  2     ! See whether we have enough space.
;  1056          1109  2     ! Return if there's already more than enough.
;  1057          1110  2     !
;  1058          1111  2     IF .XCODE[0] GEQ .DELTA[0]
;  1059          1112  2     THEN
;  1060          1113  2         RETURN TRUE;
;  1061          1114  2
;  1062          1115  2     ! Round memory request up to a multiple of 128 (i.e., get more than needed)
;  1063          1116  2     !
;  1064          1117  2     DELTA[0] = ROUND_(.DELTA[0], 128);
;  1065          1118  2
;  1066          1119  2     ! Save the old starting address
;  1067          1120  2     !
;  1068          1121  2     OLDSTART = .XCODE[1];
;  1069          1122  2
;  1070          1123  2     ! Allocate the memory
;  1071          1124  2     !
;  1072          1125  2     DELTA[1] = SOR$$ALLOCATE(.DELTA[0]);
;  1073          1126  2
;  1074          1127  2     ! Copy the old code into the new buffer
;  1075          1128  2     !
;  1076          1129  2     CH$MOVE(.XCODE[0], .XCODE[1], .DELTA[1]);
;  1077          1130  2
;  1078          1131  2     ! Deallocate the old code
;  1079          1132  2     !
;  1080          1133  2     SOR$$DEALLOCATE(.XCODE[0], XCODE[1]);
;  1081          1134  2
;  1082          1135  2     ! Copy the new length/address into COM_ROUTINES
;  1083          1136  2     !
;  1084          1137  2     XCODE[0] = .DELTA[0];
;  1085          1138  2     XCODE[1] = .DELTA[1];
;  1086          1139  2
;  1087          1140  2     ! Update the current PC
;  1088          1141  2     !
;  1089          1142  2     CUR_PC = .CUR_PC - .OLDSTART + .XCODE[1];
;  1090          1143  2
;  1091          1144  2     RETURN FALSE;
;  1092          1145  1     END;
```

```
                                                3C  BB 00000 ROOM:    PUSHR   #^M<R2,R3,R4,R5>                 ; 1089
                                5E              0C  C2 00002         SUBL2   #12, SP
                                        18      AB  9F 00005         PUSHAB  24(CTX)                         ; 1098
        52                      6E              04  C1 00008         ADDL3   #4, (SP), R2                    ; 1106
        51                      5A              62  C3 0000C         SUBL3   (R2), CUR_PC, R1
    08  AE                      51              50  C1 00010         ADDL3   SPACE, R1, DELTA
                    08  AE              00      BE  D1 00015         CMPL    @0(SP), DELTA                   ; 1111
                                                05  19 0001A         BLSS    1$
                                50              01  D0 0001C         MOVL    #1, R0                          ; 1113
                                                60  11 0001F         BRB     2$
        50          08  AE  0000007F           8F  C1 00021 1$:      ADDL3   #127, DELTA, R0                 ; 1117
    08  AE                  50  0000007F       8F  CB 0002A         BICL3   #127, R0, DELTA
        50                      6E              04  C1 00033         ADDL3   #4, (SP), R0                    ; 1121
                    04  AE                      60  D0 00037         MOVL    (R0), OLDSTART
                                        08      AE  DD 0003B         PUSHL   DELTA                           ; 1125
            00000000G   00                      01  FB 0003E         CALLS   #1, SOR$$ALLOCATE
                    0C  AE                      50  D0 00045         MOVL    R0, DELTA+4
        7E                      6E              04  C1 00049         ADDL3   #4, (SP), -(SP)                 ; 1129
                                                9E  DD 0004D         PUSHL   @(SP)+
    0C  BE                      9E          04  BE  28 0004F         MOVC3   @4(SP), @(SP)+, @DELTA+4
        50                      6E              04  C1 00055         ADDL3   #4, (SP), R0                    ; 1133
                                                50  DD 00059         PUSHL   R0
                                        04      BE  DD 0005B         PUSHL   @4(SP)
            00000000G   00                      02  FB 0005E         CALLS   #2, SOR$$DEALLOCATE
                    00  BE              08      AE  D0 00065         MOVL    DELTA, @0(SP)                   ; 1137
    50                  6E                      04  C1 0006A         ADDL3   #4, (SP), R0                    ; 1138
                        60              0C      AE  D0 0006E         MOVL    DELTA+4, (R0)
    50                  5A              04      AE  C3 00072         SUBL3   OLDSTART, CUR_PC, R0            ; 1142
    51                  6E                      04  C1 00077         ADDL3   #4, (SP), R1
    5A                  50                      61  C1 0007B         ADDL3   (R1), R0, CUR_PC
                                                50  D4 0007F         CLRL    R0                              ; 1144
                                5E              10  C0 00081 2$:      ADDL2   #16, SP                         ; 1145
                                                3C  BA 00084         POPR    #^M<R2,R3,R4,R5>
                                                    05 00086         RSB

; Routine Size:  135 bytes,    Routine Base:  SOR$RO_CODE + 03B7
```

```
;1094     1146  1 LITERAL K_MOVE = 66;                          ! Max bytes from this routine
;1095     1147  1 ROUTINE GEN_MOVE
;1096     1148  1         (
;1097     1149  1         LEN,                                    ! Bytes to move
;1098     1150  1         SRCOFF,                                 ! Source offset
;1099     1151  1         SRCREG,                                 ! Source register
;1100     1152  1         DSTOFF,                                 ! Destination offset
;1101     1153  1         DSTREG                                  ! Destination register
;1102     1154  1         ):      NOVALUE LINK_MOVE =
;1103     1155  1 !++
;1104     1156  1 ! Functional Description:
;1105     1157  1 !
;1106     1158  1 !       This routine generates code to do a simple move.
;1107     1159  1 !
;1108     1160  1 ! Formal Parameters:
;1109     1161  1 !
;1110     1162  1 !       (see above)
;1111     1163  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;1112     1164  1 !
;1113     1165  1 ! Implicit Inputs:
;1114     1166  1 !
;1115     1167  1 !       None.
;1116     1168  1 !
;1117     1169  1 ! Implicit Outputs:
;1118     1170  1 !
;1119     1171  1 !       None.
;1120     1172  1 !
;1121     1173  1 ! Routine Value:
;1122     1174  1 !
;1123     1175  1 !       None.
;1124     1176  1 !
;1125     1177  1 ! Side Effects:
;1126     1178  1 !
;1127     1179  1 !       None.
;1128     1180  1 !
;1129     1181  1 !--
;1130     1182  2     BEGIN
;1131     1183  2     EXTERNAL REGISTER
;1132     1184  2         CUR_PC = R_CUR_PC:      REF BLOCK,
;1133     1185  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK;
;1134     1186  2
;1135     1187  2     IF .SRCREG EQL COM_REG_SRC1 AND .LEN+.SRCOFF GTR .CTX[COM_SRL]
;1136     1188  2     THEN
;1137     1189  3         BEGIN
;1138     1190  3         ASSERT_(K_MOVE GEQ 1+K_LITE+8+K_DISP+K_LITE+K_LITE+K_DISP)
;1139     1191  3             !
;1140     1192  3             ! If the source is coming from the input record, and the field extends
;1141     1193  3             ! past the shortest record length, we must emit a MOVC5 instruction.
;1142     1194  3             ! It is a little tacky to assume that we are in the input conversion
;1143     1195  3             ! routine, based on SRCREG being COM_REG_SRC1, but this test suffices;
;1144     1196  3             ! Care should be taken in the rest of this module if this is not so.
;1145     1197  3             ! Code depending on this aspect of GEN_MOVE is coded as GEN_MOVE_VAR.
;1146     1198  3             !
;1147     1199  3             IF .SRCOFF NEQ 0
;1148     1200  3             THEN
;1149     1201  4                 BEGIN
;1150     1202  4                 EMIT_BYTE(OPC_SUBW3);               ! SUBW3
```

```
;  1151          1203  4              EMIT_LITE(K_WORD, .SRCOFF);          !       #srcoff,
;  1152        P 1204  4              EMIT_BYTES(M_R+R_6, M_R+R_0,         !       R6, R0
;  1153        P 1205  4                  OPC_BGEQU, 2,                    ! BGEQU 0$
;  1154        P 1206  4                  OPC_CLRW, M_R+R_0,               ! CLRW  R0
;  1155          1207  4                  OPC_MOVC5, M_R+R_0);             ! 0$: MOVC5 R0,
;  1156          1208  4              END
;  1157          1209  3          ELSE
;  1158          1210  4              BEGIN
;  1159          1211  4              EMIT_BYTES(OPC_MOVC5, M_R+R_6);      ! MOVC5 R6,
;  1160          1212  3              END;
;  1161          1213  3          EMIT_DISP(.SRCOFF, .SRCREG);             !       srcoff(srcreg),
;  1162          1214  3          EMIT_LITE(K_BYTE, .CTX[COM_PAD]);        !       #pad,
;  1163          1215  3          EMIT_LITE(K_WORD, .LEN);                 !       #len,
;  1164          1216  3          EMIT_DISP(.DSTOFF, .DSTREG);             !       dstoff(dstreg)
;  1165          1217  3          END
;  1166          1218  2      ELIF .LEN GTR TUN_K_BINMOVE
;  1167          1219  2      THEN
;  1168          1220  3          BEGIN
;  1169          1221  3          ASSERT (K_MOVE GEQ 1+K_LITE+K_DISP+K_DISP)
;  1170          1222  3          EMIT_BYTE(OPC_MOVC3);
;  1171          1223  3          EMIT_LITE(K_WORD, .LEN);
;  1172          1224  3          EMIT_DISP(.SRCOFF, .SRCREG);
;  1173          1225  3          EMIT_DISP(.DSTOFF, .DSTREG);
;  1174          1226  3          END
;  1175          1227  2      ELSE
;  1176          1228  3          BEGIN
;  1177          1229  3          BIND
;  1178          1230  3              MOVE = UPLIT BYTE(OPC_MOVB, OPC_MOVW, OPC_MOVL, OPC_MOVQ):
;  1179          1231  3                  VECTOR[,BYTE];
;  1180          1232  3          LOCAL
;  1181          1233  3              L;
;  1182          1234  3          ASSERT (K_MOVE GEQ (1+K_DISP+K_DISP)*(((TUN_K_BINMOVE-7)/8)+3))
;  1183          1235  3          L = .LEN;
;  1184          1236  3          DECR I FROM 3 TO 0 DO WHILE .L GEQ 1^.I DO
;  1185          1237  4              BEGIN
;  1186          1238  4              EMIT_BYTE(.MOVE[.I]);
;  1187          1239  4              EMIT_DISP(.SRCOFF+.LEN-.L, .SRCREG);
;  1188          1240  4              EMIT_DISP(.DSTOFF+.LEN-.L, .DSTREG);
;  1189          1241  4              L = .L - 1^.I;
;  1190          1242  3              END;
;  1191          1243  2          END;
;  1192          1244  1      END;


                        7D  D0  B0  90  0043E P.AAF:  .BYTE   -112, -80, -48, 125                    ;

                                              MOVE=           P.AAF


                                    01FC 00000 GEN_MOVE:
                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8                ; 1147
                            58      FE5C    CF  9E 00002    MOVAB   EMIT_DISP, R8                    ;
                            54        08    AC  D0 00007    MOVL    SRCOFF, R4                       ; 1199
                            51        04    AC  D0 0000B    MOVL    LEN, R1                          ; 1215
                            09        0C    AC  D1 0000F    CMPL    SRCREG, #9                       ; 1187
                                            54  12 00013    BNEQ    3$                               ;
```

```
                50      04  AC      08  AC  C1 00015          ADDL3   SRCOFF, LEN, R0
          50    0086    CB          10      00  ED 0001B      CMPZV   #0, #16, 134(CTX), R0
                                          45  18 00022      BGEQ    3$
                                          54  D5 00024      TSTL    R4                          ; 1199
                                          1D  13 00026      BEQL    1$
                                  8A      A3  8F  90 00028  1$:     MOVB    #-93, (CUR_PC)+             ; 1202
                                  53          54  D0 0002C      MOVL    R4, R3                      ; 1203
                                  52          02  D0 0002F      MOVL    #2, R2
                                          56  A8  16 00032      JSB     EMIT_LITE
                                  8A  021E5056    8F  D0 00035  2$:     MOVL    #355411078, (CUR_PC)+       ; 1207
                                  8A  502C50B4    8F  D0 0003C      MOVL    #1345081524, (CUR_PC)+
                                              05  11 00043      BRB     2$                          ; 1199
                                  8A      562C    8F  B0 00045  1$:     MOVW    #22060, (CUR_PC)+           ; 1211
                                  53      0C      AC  D0 0004A  2$:     MOVL    SRCREG, R3              ; 1213
                                  52              54  D0 0004E      MOVL    R4, R2
                                              68  16 00051      JSB     EMIT_DISP
                                  53      0101    CB  9A 00053      MOVZBL  257(CTX), R3            ; 1214
                                  52              01  D0 00058      MOVL    #1, R2
                                          56  A8  16 0005B      JSB     EMIT_LITE
                                  53              51  D0 0005E      MOVL    R1, R3                  ; 1215
                                  52              02  D0 00061      MOVL    #2, R2
                                          56  A8  16 00064      JSB     EMIT_LITE
                                              1A  11 00067      BRB     4$                          ; 1216
                                  20              51  D1 00069  3$:     CMPL    R1, #32             ; 1218
                                              1C  15 0006C      BLEQ    5$
                                  8A              28  90 0006E      MOVB    #40, (CUR_PC)+          ; 1222
                                  53              51  D0 00071      MOVL    R1, R3                  ; 1223
                                  52              02  D0 00074      MOVL    #2, R2
                                          56  A8  16 00077      JSB     EMIT_LITE
                                  53      0C      AC  D0 0007A      MOVL    SRCREG, R3              ; 1224
                                  52              54  D0 0007E      MOVL    R4, R2
                                              68  16 00081      JSB     EMIT_DISP
                                  52      10      AC  7D 00083  4$:     MOVQ    DSTOFF, R2          ; 1225
                                              68  16 00087      JSB     EMIT_DISP
                                                  04 00089      RET                                 ; 1217
                                  55              51  D0 0008A  5$:     MOVL    R1, L               ; 1235
                                  54              51  C0 0008D      ADDL2   R1, R4                  ; 1239
                      56          51      10      AC  C1 00090      ADDL3   DSTOFF, R1, R6          ; 1240
                                  51              03  D0 00095      MOVL    #3, I
                      57          01              51  78 00098  6$:     ASHL    I, #1, R7           ; 1236
                                  57              55  D1 0009C  7$:     CMPL    L, R7
                                              22  19 0009F      BLSS    8$
                                  8A  FF56 CF41   90  00A1      MOVB    MOVE[I], (CUR_PC)+          ; 1238
                      50          54              55  C3 00A7      SUBL3   L, R4, R0               ; 1239
                                  53      0C      AC  D0 00AB      MOVL    SRCREG, R3
                                  52              50  D0 00AF      MOVL    R0, R2
                                              68  16 00B2      JSB     EMIT_DISP
                      52          56              55  C3 00B4      SUBL3   L, R6, R2               ; 1240      ; F
                                  53      14      AC  D0 00B8      MOVL    DSTREG, R3
                                              68  16 00BC      JSB     EMIT_DISP
                                  55              57  C2 00BE      SUBL2   R7, L                   ; 1241
                                              D9  11 00C1      BRB     7$                          ; 1236
                                  D2              51  F4 00C3  8$:     SOBGEQ  I, 6$
                                                  04 00C6      RET                                 ; 1244

; Routine Size:  199 bytes,    Routine Base:  SOR$RO_CODE + 0442
```

```
; 1193          1245  1 BIND ROUTINE GEN_MOVE_VAR = GEN_MOVE: NOVALUE LINK_MOVE;
```

```
;  1195          1246  1 ROUTINE GEN_CONVERT_DEC
;  1196          1247  1         (
;  1197          1248  1         PKBF:           REF KBF_BLOCK,          ! Key description
;  1198          1249  1         DISP,                                   ! Displacement from SRC2
;  1199          1250  1         STACK:          REF VECTOR[1]           ! Amount of stack needed
;  1200          1251  1         ):              LINK_COMPARE =
;  1201          1252  1 !++
;  1202          1253  1 ! Functional Description:
;  1203          1254  1 !
;  1204          1255  1 !       This routine generates code to convert a single decimal key.
;  1205          1256  1 !
;  1206          1257  1 ! Formal Parameters:
;  1207          1258  1 !
;  1208          1259  1 !       PKBF            Address of the key description.
;  1209          1260  1 !                       This is modified to reflect the new key description.
;  1210          1261  1 !       DISP            Displacement from SRC2 of where to write the key.
;  1211          1262  1 !       STACK           Address of the amount of temporary stack allocation
;  1212          1263  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1213          1264  1 !
;  1214          1265  1 ! Implicit Inputs:
;  1215          1266  1 !
;  1216          1267  1 !       None.
;  1217          1268  1 !
;  1218          1269  1 ! Implicit Outputs:
;  1219          1270  1 !
;  1220          1271  1 !       None.
;  1221          1272  1 !
;  1222          1273  1 ! Routine Value:
;  1223          1274  1 !
;  1224          1275  1 !       Length in bytes of the converted key.
;  1225          1276  1 !
;  1226          1277  1 ! Side Effects:
;  1227          1278  1 !
;  1228          1279  1 !       None.
;  1229          1280  1 !
;  1230          1281  1 ! Notes:
;  1231          1282  1 !
;  1232          1283  1 !       It is worthwhile to convert packed numbers to a more convienient form,
;  1233          1284  1 !       because of the performance of the CMPP instructions, particularly on
;  1234          1285  1 !       VAX architectures that don't implement these instructions in microcode.
;  1235          1286  1 !       This is not yet implemented.
;  1236          1287  1 !
;  1237          1288  1 !       Note the following classification of how the sign nibble compares on
;  1238          1289  1 !       VAX-11/780s:
;  1239          1290  1 !               Negative: 1,3,5,9,B,D
;  1240          1291  1 !               Positive: 0,2,4,6,7,8,A,C,E,F
;  1241          1292  1 !               All forms of zero compare equal.
;  1242          1293  1 !--
;  1243          1294  2     BEGIN
;  1244          1295  2     EXTERNAL REGISTER
;  1245          1296  2         CTX=    COM_REG_CTX:    REF CTX_BLOCK,
;  1246          1297  2         CUR_PC= R_COR_PC:       REF BLOCK;
;  1247          1298  2     LITERAL
;  1248          1299  2         K_MAXDEC =      31;     ! Maximum length of decimal data
;  1249          1300  2     LOCAL
;  1250          1301  2         POFF,                   ! Offset to packed number on stack
;  1251          1302  2         CVTLEN,                 ! Length in bytes of converted data
```

```
;  1252          1303  2          LEN,                        ! Length used in CVTSP and CVTTP instructions
;  1253          1304  2          REG_SRC,                    ! Source register
;  1254          1305  2          KBF:    REF KBF_BLOCK;      ! Key description
;  1255          1306  2
;  1256          1307  2      %IF NOT HOSTILE %THEN
;  1257          1308  2
;  1258          1309  2      KBF = PKBF[BASE_];
;  1259          1310  2      LEN = .KBF[KBF_LENGTH];
;  1260          1311  3      IF ONEOF_(.KBF[KBF_TYPE], BMSK_(DSC$K_DTYPE_NL, DSC$K_DTYPE_NR))
;  1261          1312  2      THEN
;  1262          1313  2          LEN = .LEN - 1;             ! DSC length includes sign, the hardware doesn't
;  1263          1314  2
;  1264          1315  2      IF .LEN EQL 1 AND .KBF[KBF_TYPE] EQL DSC$K_DTYPE_NLO
;  1265          1316  2      THEN
;  1266          1317  2          KBF[KBF_TYPE] = DSC$K_DTYPE_NRO;
;  1267          1318  2
;  1268          1319  2
;  1269          1320  2      ! If all we will be looking at is a sign, don't do this test, since
;  1270          1321  2      ! +0 and -0 sort equally.
;  1271          1322  2      !
;  1272          1323  2      IF .LEN EQL 0 AND
;  1273          1324  3           ONEOF_(.KBF[KBF_TYPE], BMSK_(DSC$K_DTYPE_NL, DSC$K_DTYPE_NR))
;  1274          1325  2      THEN
;  1275          1326  3          BEGIN
;  1276          1327  3          KBF[KBF_TYPE] = DSC$K_DTYPE_Z;
;  1277          1328  3          KBF[KBF_LENGTH] = 0;
;  1278          1329  3          RETURN 0;
;  1279          1330  2          END;
;  1280          1331  2
;  1281          1332  2
;  1282          1333  2      ! Determine how much stack space is required for this routine
;  1283          1334  2      !
;  1284          1335  3      BEGIN
;  1285          1336  3      LOCAL
;  1286          1337  3          STAK;
;  1287          1338  3      POFF = 0;
;  1288          1339  3      STAK = 0;
;  1289          1340  4      IF ONEOF_(.KBF[KBF_TYPE], BMSK_(DSC$K_DTYPE_NLO, DSC$K_DTYPE_NR))
;  1290          1341  3      THEN
;  1291          1342  3          STAK = ROUND_(.LEN+1);
;  1292          1343  3      IF .LEN LEQ 9
;  1293          1344  3      THEN
;  1294          1345  4          BEGIN
;  1295          1346  4          POFF = .STAK;
;  1296          1347  4          STAK = .STAK + PLEN_(.LEN);
;  1297          1348  3          END;
;  1298          1349  3      STACK[0] = MAX(.STACK[0], .STAK);
;  1299          1350  2      END;
;  1300          1351  2
;  1301          1352  2      ! Normally, the source comes from the input record (COM_REG_SRC2).
;  1302          1353  2      ! However, for indexed sorts, the key may have already been moved to the
;  1303          1354  2      ! internal format record.  If so, use COM_REG_SRC2 as the input register.
;  1304          1355  2      !
;  1305          1356  2      REG_SRC = COM_REG_SRC1;
;  1306          1357  2      IF .KBF[KBF_CVT] THEN REG_SRC = COM_REG_SRC2;
;  1307          1358  2
;  1308          1359  2      ! Convert the number to packed.
```

```
;  1309          1360  2      !
;  1310          1361  2      IF .KBF[KBF_TYPE] EQL DSC$K_DTYPE_NLO
;  1311          1362  2      THEN
;  1312          1363  3          BEGIN
;  1313          1364  3          ROOM(K_MOVE+1+K_DISP+3+K_ABSA+13);
;  1314          1365  3          GEN_MOVE(MAX(.LEN-1,0),                          ! MOVE  #len-1
;  1315          1366  3              .KBF[KBF_POSITION]+1, .REG_SRC,              !       1(src)
;  1316          1367  3              2, R_SP);                                    !       2(SP)
;  1317          1368  3          EMIT_BYTE(OPC_MOVZBL);                           ! MOVZBL
;  1318          1369  3          EMIT_DISP(.KBF[KBF_POSITION], .REG_SRC);         !       (src)
;  1319        P 1370  3          EMIT_BYTES(M_R+R_0,                              !       R0
;  1320          1371  3              OPC_MOVB, M_I+R_0);                          ! MOVB
;  1321          1372  3          EMIT_ABSA(LIB$AB_CVTTP_0);                       !       CVTTP_0[R0]
;  1322        P 1373  3          EMIT_BYTES(M_AD+R_SP,                            !       -(SP)
;  1323        P 1374  3              OPC_MOVB, 1, M_AD+R_SP,                      ! MOVB  #1, -(SP)
;  1324        P 1375  3              OPC_CVTPS, 2, M_RD+R_SP,                     ! CVTPS #2, (SP)
;  1325        P 1376  3              1, M_BD+R_SP, 2,                             !       #1, 2(SP)
;  1326          1377  3              OPC_ADDL2, 2, M_R+R_SP);                     ! ADDL2 #2, SP
;  1327          1378  3          END
;  1328          1379  2      ELIF
;  1329          1380  2          .KBF[KBF_TYPE] EQL DSC$K_DTYPE_NR
;  1330          1381  2      THEN
;  1331          1382  3          BEGIN
;  1332          1383  3          ROOM(1+K_DISP+1+K_MOVE);
;  1333          1384  3          EMIT_BYTE(OPC_MOVB);                             ! MOVB  nn+len(Rsrc1)
;  1334          1385  3          EMIT_DISP(.KBF[KBF_POSITION]+.LEN, .REG_SRC);
;  1335          1386  3          EMIT_BYTE(M_RD+R_SP);                            !       (SP)
;  1336          1387  3          GEN_MOVE(.LEN,                                   ! MOVE  #len
;  1337          1388  3              .KBF[KBF_POSITION], .REG_SRC,                !       nn(Rsrc1)
;  1338          1389  3              1, R_SP);                                    !       1(SP)
;  1339          1390  2          END;
;  1340          1391  2
;  1341          1392  2      ROOM(1+K_LITE+MAX(1,K_DISP)+K_ABSA+K_LITE);
;  1342          1393  2
;  1343          1394  2      ! Emit the opcode
;  1344          1395  2      !
;  1345        P 1396  2      IF ONEOF (.KBF[KBF_TYPE], BMSK (
;  1346          1397  3          DSC$K_DTYPE_NU, DSC$K_DTYPE_NZ, DSC$K_DTYPE_NRO))
;  1347          1398  2      THEN
;  1348          1399  2          EMIT_BYTE(OPC_CVTTP)
;  1349          1400  2      ELSE
;  1350          1401  2          EMIT_BYTE(OPC_CVTSP);
;  1351          1402  2
;  1352          1403  2      ! Emit the source length
;  1353          1404  2      !
;  1354          1405  2      EMIT_LITE(K_WORD, .LEN);
;  1355          1406  2
;  1356          1407  2
;  1357          1408  2      ! Emit the source address
;  1358          1409  2      !
;  1359        P 1410  2      IF ONEOF (.KBF[KBF_TYPE], BMSK (
;  1360          1411  3          DSC$K_DTYPE_NLO, DSC$K_DTYPE_NR))
;  1361          1412  2      THEN
;  1362          1413  3          EMIT_BYTES(M_RD+R_SP)
;  1363          1414  2      ELSE
;  1364          1415  2          EMIT_DISP(.KBF[KBF_POSITION], .REG_SRC);
;  1365          1416  2
```

```
;  1366          1417  2
;  1367          1418  2      ! Emit a reference to the appropriate translation table, if needed
;  1368          1419  2      !
;  1369          1420  2      IF .KBF[KBF_TYPE] EQL DSC$K_DTYPE_NU
;  1370          1421  2      THEN
;  1371          1422  3          EMIT_ABSA(LIB$AB_CVTTP_U)
;  1372          1423  2      ELIF .KBF[KBF_TYPE] EQL DSC$K_DTYPE_NZ
;  1373          1424  2      THEN
;  1374          1425  3          EMIT_ABSA(LIB$AB_CVTTP_Z)
;  1375          1426  2      ELIF .KBF[KBF_TYPE] EQL DSC$K_DTYPE_NRO
;  1376          1427  2      THEN
;  1377          1428  2          EMIT_ABSA(LIB$AB_CVTTP_O);
;  1378          1429  2
;  1379          1430  2
;  1380          1431  2      ! The destination length
;  1381          1432  2      !
;  1382          1433  2      EMIT_LITE(K_WORD, .LEN);
;  1383          1434  2
;  1384          1435  2
;  1385          1436  2      ! Determine whether packed is the best we can do
;  1386          1437  2      !
;  1387          1438  2      ROOM(K_DISP+1+K_LITE+1+1+2+K_DISP+1+K_LITE+1);
;  1388          1439  2      IF .LEN GTR 9
;  1389          1440  2      THEN                                        ! Value won't fit in a longword
;  1390          1441  3          BEGIN
;  1391          1442  3          !
;  1392          1443  3          ! We've converted the number to packed
;  1393          1444  3          !
;  1394          1445  3          KBF[KBF_TYPE] = DSC$K_DTYPE_P;
;  1395          1446  3          CVTLEN = PLEN (.LEN);   ! Value to return
;  1396          1447  3          KBF[KBF_LENGTH] = .LEN;
;  1397          1448  3          END
;  1398          1449  2      ELSE                                        ! Value will fit in a longword
;  1399          1450  3          BEGIN
;  1400          1451  3          !
;  1401          1452  3          ! We will convert to some form of signed binary
;  1402          1453  3          !
;  1403          1454  3          KBF[KBF_TYPE] = DSC$K_DTYPE_B;
;  1404          1455  3          EMIT_DISP(.POFF, R_SP);                 !       xx(SP)
;  1405          1456  3          EMIT_BYTE(OPC_CVTP[);                   ! CVTPL
;  1406          1457  3          EMIT_LITE(K_WORD, .LEN);                !       #len
;  1407          1458  3          EMIT_BYTE(M_RD+R_3);                    !       (R3)    packed number
;  1408          1459  3          IF .LEN GTR 4
;  1409          1460  3          THEN                                    ! Value won't fit in a word
;  1410          1461  4              BEGIN
;  1411          1462  4              CVTLEN = 4;
;  1412          1463  4              END
;  1413          1464  3          ELSE                                    ! Value will fit in a word
;  1414          1465  4              BEGIN
;  1415          1466  4              EMIT_BYTE(M_R+R_0);
;  1416          1467  4              IF .LEN GTR 2
;  1417          1468  4              THEN                                ! Value won't fit in a byte
;  1418          1469  5                  BEGIN
;  1419          1470  5                  EMIT_BYTES(OPC_MOVW,M_R+R_0);
;  1420          1471  5                  CVTLEN = 2;
;  1421          1472  5                  END
;  1422          1473  4              ELSE                                ! Value will fit in a byte
```

```
;  1423          1474  5                 BEGIN
;  1424          1475  5                 EMIT_BYTES(OPC_MOVB,M_R+R_0);
;  1425          1476  5                 CVTLEN = 1;
;  1426          1477  4                 END;
;  1427          1478  3             END;
;  1428          1479  3         KBF[KBF_LENGTH] = .CVTLEN;
;  1429          1480  2         END;
;  1430          1481  2
;  1431          1482  2     EMIT_DISP(.DISP, COM_REG_SRC2);
;  1432          1483  2
;  1433          1484  2     KBF[KBF_POSITION] = .DISP;
;  1434          1485  2     KBF[KBF_CVT] = TRUE;
;  1435          1486  2
;  1436        U 1487  2     %ELSE
;  1437        U 1488  2
;  1438        U 1489  2     SOR$$ERROR(SOR$_SHR_BADLOGIC);
;  1439        U 1490  2     CVTLEN = 0;
;  1440        U 1491  2
;  1441          1492  2     %FI
;  1442          1493  2
;  1443          1494  2     RETURN .CVTLEN;
;  1444          1495  1     END;


                                                    GEN_MOVE_VAR=       GEN_MOVE

                                        03FC 00000 GEN_CONVERT_DEC:
                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                    ; 1246
                59      FD95    CF  9E 00002        MOVAB   EMIT_DISP, R9
                54        04    AC  D0 00007        MOVL    PKBF, KBF                                       ; 1309
                56        06    A4  3C 0000B        MOVZWL  6(KBF), LEN                                     ; 1310
     50 0000A000  8F            64  78 0000F        ASHL    (KBF), #40960, R0                               ; 1311
                                02  18 00017        BGEQ    1$
                                56  D7 00019        DECL    LEN                                             ; 1313
                01              56  D1 0001B 1$:    CMPL    LEN, #1                                         ; 1315
                                08  12 0001E        BNEQ    2$
                11              64  B1 00020        CMPW    (KBF), #17
                                03  12 00023        BNEQ    2$
                64              13  B0 00025        MOVW    #19, (KBF)                                      ; 1317
                                56  D5 00028 2$:    TSTL    LEN                                             ; 1323
                                12  12 0002A        BNEQ    3$
     50 0000A000  8F            64  78 0002C        ASHL    (KBF), #40960, R0                               ; 1324
                                08  18 00034        BGEQ    3$
                                64  B4 00036        CLRW    (KBF)                                           ; 1327
                        06      A4  B4 00038        CLRW    6(KBF)                                          ; 1328
                            01C5 31  31 0003B       BRW     25$                                             ; 1329
                                57  D4 0003E 3$:    CLRL    POFF                                            ; 1338
                                50  D4 00040        CLRL    STAK                                            ; 1339
     51 00006000  8F            64  78 00042        ASHL    (KBF), #24576, R1                               ; 1340
                                58  D4 0004A        CLRL    R8
                                51  D5 0004C        TSTL    R1
                                0A  18 0004E        BGEQ    4$
                                58  D6 00050        INCL    R8
                59        04    A6  9E 00052        MOVAB   4(R6), R3                                       ; 1342
     50         53            03  CB 00056        BICL3   #3, R3, STAK                                    ;
```

```
                09              56  D1 0005A 4$:    CMPL    LEN, #9                         ; 1343
                                0C  14 0005D        BGTR    5$
                57              50  D0 0005F        MOVL    STAK, POFF                      ; 1346
          53    56              02  C7 00062        DIVL3   #2, LEN, R3                     ; 1347
                50        01 A340  9E 00066        MOVAB   1(R3)[STAK], STAK
                51        0C    BC  D0 0006B 5$:    MOVL    @STACK, R1                      ; 1349
                50              51  D1 0006F        CMPL    R1, STAK
                                03  18 00072        BGEQ    6$
                51              50  D0 00074        MOVL    STAK, R1
             0C BC              51  D0 00077 6$:    MOVL    R1, @STACK
                55              09  D0 0007B        MOVL    #9, REG_SRC                     ; 1356
          03 02 A4              01  E1 0007E        BBC     #1, 2(KBF), 7$                  ; 1357
                55              0A  D0 00083        MOVL    #10, REG_SRC
                11              64  B1 00086 7$:    CMPW    (KBF), #17                      ; 1361
                                59  12 00089        BNEQ    9$
                50        5D    8F  9A 0008B        MOVZBL  #93, R0                         ; 1364
                        0113    C9  16 0008F        JSB     ROOM
                                0E  DD 00093        PUSHL   #14                             ; 1365
                                02  DD 00095        PUSHL   #2
                                55  DD 00097        PUSHL   REG_SRC                         ; 1366
                7E        04    A4  3C 00099        MOVZWL  4(KBF), -(SP)
                                6E  D6 0009D        INCL    (SP)
                53        FF    A6  9E 0009F        MOVAB   -1(R6), R3                      ; 1365
                                53  DD 000A3        PUSHL   R3
                                02  18 000A5        BGEQ    8$
                                6E  D4 000A7        CLRL    (SP)
             019E C9            05  FB 000A9 8$:    CALLS   #5, GEN_MOVE
                8A        9A    8F  90 000AE        MOVB    #-102, (CUR_PC)+                ; 1368
                53              55  D0 000B2        MOVL    REG_SRC, R3                     ; 1369
                52        04    A4  3C 000B5        MOVZWL  4(KBF), R2
                                69  16 000B9        JSB     EMIT_DISP
                8A 9F409050     8F  D0 000BB        MOVL    #-1623158704, (CUR_PC)+         ; 1371
                8A 00000000G    00  9E 000C2        MOVAB   LIB$AB_CVTTP_0, (CUR_PC)+       ; 1372
                8A 7E01907E     8F  D0 000C9        MOVL    #2114031742, (CUR_PC)+          ; 1377
                8A 016E0208     8F  D0 000D0        MOVL    #23986696, (CUR_PC)+
                8A 02C002AE     8F  D0 000D7        MOVL    #46138030, (CUR_PC)+
                8A        5E    8F  90 000DE        MOVB    #94, (CUR_PC)+
                                2F  11 000E2        BRB     10$                             ; 1361
                12              64  B1 000E4 9$:    CMPW    (KBF), #18                      ; 1380
                                2A  12 000E7        BNEQ    10$
                50        49    8F  9A 000E9        MOVZBL  #73, R0                         ; 1383
                        0113    C9  16 000ED        JSB     ROOM
                8A        90    8F  90 000F1        MOVB    #-112, (CUR_PC)+                ; 1384
                51        04    A4  3C 000F5        MOVZWL  4(KBF), R1                      ; 1385
          52    51              56  C1 000F9        ADDL3   LEN, R1, R2
                53              55  D0 000FD        MOVL    REG_SRC, R3
                                69  16 00100        JSB     EMIT_DISP
                8A        6E    8F  90 00102        MOVB    #110, (CUR_PC)+                 ; 1386
                                0E  DD 00106        PUSHL   #14                             ; 1387
                                01  DD 00108        PUSHL   #1
                                22  BB 0010A        PUSHR   #^M<R1,R5>                      ; 1388
                                56  DD 0010C        PUSHL   LEN                             ; 1387
             019E C9            05  FB 0010E        CALLS   #5, GEN_MOVE
                50              15  D0 00113 10$:   MOVL    #21, R0                         ; 1392
                        0113    C9  16 00116        JSB     ROOM
 50 00011800    8F              64  78 0011A        ASHL    (KBF), #71680, R0               ; 1397
                                05  18 00122        BGEQ    11$
```

```
                  6A             26  90 00124          MOVB    #38, (CUR_PC)                        ; 1399
                                 03  11 00127          BRB     12$                                  ; 1401
                  6A             09  90 00129 11$:     MOVB    #9, (CUR_PC)
                                 5A  D6 0012C 12$:     INCL    CUR_PC                               ; 1399
                  53             56  D0 0012E          MOVL    LEN, R3                              ; 1405
                  52             02  D0 00131          MOVL    #2, R2                               ;
                           56    A9  16 00134          JSB     EMIT_LITE                            ;
                  06             58  E9 00137          BLBC    R8, 13$                              ; 1411
                  8A       6E    8F  90 0013A          MOVB    #110, (CUR_PC)+                      ; 1413
                                 09  11 0013E          BRB     14$                                  ; 1410
                  53             55  D0 00140 13$:     MOVL    REG_SRC, R3                          ; 1415
                  52       04    A4  3C 00143          MOVZWL  4(KBF), R2                           ;
                                 69  16 00147          JSB     EMIT_DISP                            ;
                  0F             64  B1 00149 14$:     CMPW    (KBF), #15                           ; 1420
                                 0D  12 0014C          BNEQ    15$                                  ;
                  8A       9F    8F  90 0014E          MOVB    #-97, (CUR_PC)+                      ; 1422
                  6A 00000000G   00  9E 00152          MOVAB   LIB$AB_CVTTP_U, (CUR_PC)             ;
                                 22  11 00159          BRB     17$                                  ;
                  14             64  B1 0015B 15$:     CMPW    (KBF), #20                           ; 1423
                                 0D  12 0015E          BNEQ    16$                                  ;
                  8A       9F    8F  90 00160          MOVB    #-97, (CUR_PC)+                      ; 1425
                  6A 00000000G   00  9E 00164          MOVAB   LIB$AB_CVTTP_Z, (CUR_PC)             ;
                                 10  11 0016B          BRB     17$                                  ;
                  13             64  B1 0016D 16$:     CMPW    (KBF), #19                           ; 1426
                                 0E  12 00170          BNEQ    18$                                  ;
                  8A       9F    8F  90 00172          MOVB    #-97, (CUR_PC)+                      ; 1428
                  6A 00000000G   00  9E 00176          MOVAB   LIB$AB_CVTTP_O, (CUR_PC)             ;
                  5A             04  C0 0017D 17$:     ADDL2   #4, CUR_PC                           ;
                  53             56  D0 00180 18$:     MOVL    LEN, R3                              ; 1433
                  52             02  D0 00183          MOVL    #2, R2                               ;
                           56    A9  16 00186          JSB     EMIT_LITE                            ;
                  50             1B  D0 00189          MOVL    #27, R0                              ; 1438
                         0113    C9  16 0018C          JSB     ROOM                                 ;
                  09             56  D1 00190          CMPL    LEN, #9                              ; 1439
                                 11  15 00193          BLEQ    19$                                  ;
                  64             15  B0 00195          MOVW    #21, (KBF)                           ; 1445
     53           56             02  C7 00198          DIVL3   #2, LEN, R3                          ; 1446
                  51       01    A3  9E 0019C          MOVAB   1(R3), CVTLEN                        ;
             06   A4             56  B0 001A0          MOVW    LEN, 6(KBF)                          ; 1447
                                 47  11 001A4          BRB     24$                                  ; 1439
                  64             06  B0 001A6 19$:     MOVW    #6, (KBF)                            ; 1454
                  53             0E  D0 001A9          MOVL    #14, R3                              ; 1455
                  52             57  D0 001AC          MOVL    POFF, R2                             ;
                                 69  16 001AF          JSB     EMIT_DISP                            ;
                  8A             36  90 001B1          MOVB    #54, (CUR_PC)+                       ; 1456
                  53             56  D0 001B4          MOVL    LEN, R3                              ; 1457
                  52             02  D0 001B7          MOVL    #2, R2                               ;
                           56    A9  16 001BA          JSB     EMIT_LITE                            ;
                  8A       63    8F  90 001BD          MOVB    #99, (CUR_PC)+                       ; 1458
                  04             56  D1 001C1          CMPL    LEN, #4                              ; 1459
                                 05  15 001C4          BLEQ    20$                                  ;
                  51             04  D0 001C6          MOVL    #4, CVTLEN                           ; 1462
                                 1E  11 001C9          BRB     23$                                  ; 1459
                  8A       50    8F  90 001CB 20$:     MOVB    #80, (CUR_PC)+                       ; 1466
                  02             56  D1 001CF          CMPL    LEN, #2                              ; 1467
                                 0A  15 001D2          BLEQ    21$                                  ;
                  6A     50B0    8F  B0 001D4          MOVW    #20656, (CUR_PC)                     ; 1470
```

```
                        51              02  D0 001D9          MOVL    #2, CVTLEN                      ; 1471
                                        08  11 001DC          BRB     22$                             ; 1467
                        6A      5090    8F  B0 001DE 21$:     MOVW    #20624, (CUR_PC)                ; 1475
                        51              01  D0 001E3          MOVL    #1, CVTLEN                      ; 1476
                        5A              02  C0 001E6 22$:     ADDL2   #2, CUR_PC                      ; 1470
                    06  A4              51  B0 001E9 23$:     MOVW    CVTLEN, 6(KBF)                  ; 1479
                        53              0A  D0 001ED 24$:     MOVL    #10, R3                         ; 1482
                        52          08  AC  D0 001F0          MOVL    DISP, R2                        ;
                                        69  16 001F4          JSB     EMIT_DISP                       ;
                    04  A4          08  AC  B0 001F6          MOVW    DISP, 4(KBF)                    ; 1484
                    02  A4              02  88 001FB          BISB2   #2, 2(KBF)                      ; 1485
                        50              51  D0 001FF          MOVL    CVTLEN, R0                      ; 1494
                                            04 00202          RET                                     ;
                                        50  D4 00203 25$:     CLRL    R0                              ; 1495
                                            04 00205          RET                                     ;

; Routine Size:  518 bytes,    Routine Base:  SOR$RO_CODE + 0509
```

```
;  1446          1496  1 ROUTINE GEN_CONVERT_FLT
;  1447          1497  1         (
;  1448          1498  1         PKBF:           REF KBF_BLOCK,  ! Key description
;  1449          1499  1         DISP                            ! Displacement from SRC2
;  1450          1500  1         ):              LINK_COMPARE =
;  1451          1501  1 !++
;  1452          1502  1 ! Functional Description:
;  1453          1503  1 !
;  1454          1504  1 !       This routine generates code to convert a single (F,D,G,H) floating key.
;  1455          1505  1 !
;  1456          1506  1 ! Formal Parameters:
;  1457          1507  1 !
;  1458          1508  1 !       PKBF            Address of the key description.
;  1459          1509  1 !                       This is modified to reflect the new key description.
;  1460          1510  1 !       DISP            Displacement from SRC2 of where to write the key.
;  1461          1511  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1462          1512  1 !
;  1463          1513  1 ! Implicit Inputs:
;  1464          1514  1 !
;  1465          1515  1 !       None.
;  1466          1516  1 !
;  1467          1517  1 ! Implicit Outputs:
;  1468          1518  1 !
;  1469          1519  1 !       None.
;  1470          1520  1 !
;  1471          1521  1 ! Routine Value:
;  1472          1522  1 !
;  1473          1523  1 !       Length in bytes of the converted key.
;  1474          1524  1 !
;  1475          1525  1 ! Side Effects:
;  1476          1526  1 !
;  1477          1527  1 !       None.
;  1478          1528  1 !
;  1479          1529  1 !--
;  1480          1530  2     BEGIN
;  1481          1531  2     EXTERNAL REGISTER
;  1482          1532  2         CTX=     COM_REG_CTX:    REF CTX_BLOCK,
;  1483          1533  2         CUR_PC= R_CUR_PC:        REF BLOCK;
;  1484          1534  2     LOCAL
;  1485          1535  2         FDGH:    REF BLOCK,
;  1486          1536  2         KBF:     REF KBF_BLOCK,          ! Key description
;  1487          1537  2         REG_SRC,                         ! Source register
;  1488          1538  2         TMP:     REF VECTOR[,BYTE];
;  1489          1539  2     ASSERT_(DSC$K_DTYPE_F MOD 5 EQL 0)
;  1490          1540  2     ASSERT_(DSC$K_DTYPE_D MOD 5 EQL 1)
;  1491          1541  2     ASSERT_(DSC$K_DTYPE_G MOD 5 EQL 2)
;  1492          1542  2     ASSERT_(DSC$K_DTYPE_H MOD 5 EQL 3)
;  1493          1543  2     MACRO
;  1494          1544  2         X_LEN = 0,0,8,0 %,      ! Length in bytes
;  1495          1545  2         X_MB1 = 0,16,16,0 %;    ! Mantissa bits in first word
;  1496          1546  2     OWN
;  1497          1547  2         OWN_FDGH: BLOCKVECTOR[4,1]
;  1498          1548  2             PSECT(SOR$RO_CODE) PRESET(
;  1499          1549  2                 [0,X_LEN]=      4,              ! Length in bytes
;  1500          1550  2                 [1,X_LEN]=      8,
;  1501          1551  2                 [2,X_LEN]=      8,
;  1502          1552  2                 [3,X_LEN]=      16,
```

```
;  1503          1553  2                [0,X_MB1]=      1^7-1,          ! Mantissa bits in first word
;  1504          1554  2                [1,X_MB1]=      1^7-1,
;  1505          1555  2                [2,X_MB1]=      1^4-1,
;  1506          1556  2                [3,X_MB1]=      1^0-1);
;  1507          1557  2
;  1508          1558  2      ! Normally, the source comes from the input record (COM_REG_SRC2).
;  1509          1559  2      ! However, for indexed sorts, the key may have already been moved to the
;  1510          1560  2      ! internal format record.  If so, use COM_REG_SRC2 as the input register.
;  1511          1561  2      !
;  1512          1562  2      KBF = PKBF[BASE_];
;  1513          1563  2      REG_SRC = COM_REG_SRC1;
;  1514          1564  2      IF .KBF[KBF_CVT] THEN REG_SRC = COM_REG_SRC2;
;  1515          1565  2
;  1516          1566  2      ! We convert the floating point numbers as follows:
;  1517          1567  2      !
;  1518          1568  2      ! The first word contains the exponent, and (except for H) a few mantissa
;  1519          1569  2      ! bits.  The low word is ordered as follows (smaller to larger):
;  1520          1570  2      !   FFF0..8010,800x(reserved),000x(zero),0010..7FFF
;  1521          1571  2      !
;  1522          1572  2      ! If the sign of the number is negative, negate all the bits (except the
;  1523          1573  2      ! sign bit).
;  1524          1574  2      ! If the sign is positive, check for zero.  If not zero, clear the result,
;  1525          1575  2      ! otherwise, copy the number.
;  1526          1576  2      !
;  1527          1577  2      ROOM(1+K_DISP+2+K_DISP+6+1+K_LITE+3+8*4+8+6);
;  1528          1578  2
;  1529          1579  2      ! Get a pointer to the appropriate entry in the OWN_FDGH table.
;  1530          1580  2      !
;  1531          1581  2      FDGH = OWN_FDGH[.KBF[KBF_TYPE] MOD 5, BASE_];
;  1532          1582  2
;  1533          1583  2      ! Get the address of the source
;  1534          1584  2      ! Get the address just past the destination
;  1535          1585  2      ! Fetch the (sign-extended) sign of the source
;  1536          1586  2      !
;  1537          1587  2      EMIT_BYTES(OPC_MOVAB);                                  ! MOVAB
;  1538          1588  2      EMIT_DISP(.KBF[KBF_POSITION], .REG_SRC);                !       (src1)
;  1539        P 1589  2      EMIT_BYTES(M_R+R_1,                                     !       R1
;  1540          1590  2          OPC_MOVAB);                                         ! MOVAB
;  1541          1591  2      EMIT_DISP(.DISP+.FDGH[X_LEN], COM_REG_SRC2);            !       4/8/8/16(src2)
;  1542        P 1592  2      EMIT_BYTES(M_R+R_2,                                     !       R2
;  1543        P 1593  2          OPC_EXTV, 15, 1, M_RD+R_1,                          ! EXTV  #15, #1, (R1),
;  1544          1594  2          M_R+R_0);                                           !       R0
;  1545          1595  2
;  1546          1596  2      ! Test for an exponent of zero.
;  1547          1597  2      !
;  1548          1598  2      EMIT_BYTES(OPC_CMPW);                                   ! CMPW
;  1549          1599  2      EMIT_LITE(K_WORD, .FDGH[X_MB1]);                        !       #^X7F/7F/0F/00
;  1550        P 1600  2      EMIT_BYTES(M_RD+R_1,                                    !       (R1)
;  1551          1601  2          OPC_BGEQU, 0);                                      ! BGEQU 1$
;  1552          1602  2      TMP = .CUR_PC;
;  1553          1603  2
;  1554          1604  2      ! Copy the number, complementing if negative
;  1555          1605  2      !
;  1556          1606  2      DECR I FROM .FDGH[X_LEN]/2-1 TO 0 DO                    ! Do 2/4/4/8 times
;  1557        P 1607  2          EMIT_BYTES(OPC_XORW3, M_R+R_0,                      ! XORW3 R0,
;  1558          1608  2                  M_AI+R_1, M_AD+R_2);                        !       (R1)+, -(R2)
;  1559          1609  2
```

```
;  1560          1610  2      ! Insert the sign bit
;  1561          1611  2      !
;  1562        P 1612  2      EMIT_BYTES(OPC_INSV, M_R+R_0, 7, 1,                 ! INSV  R0, #7, #1
;  1563        P 1613  2          M_BD+R_2, .FDGH[X_LEN]=1,                       !       4/8/8/16-1(R2)
;  1564          1614  2          OPC_BRB, 0);                                    ! BRB   2$
;  1565          1615  2      TMP[-1] = .CUR_PC - .TMP;                           !1$:
;  1566          1616  2      TMP = .CUR_PC;
;  1567          1617  2
;  1568          1618  2      ! Zero the destination
;  1569          1619  2      !
;  1570          1620  2      IF .FDGH[X_LEN] GEQ 16  THEN EMIT_BYTES(OPC_CLRQ, M_AD+R_2);
;  1571          1621  3      IF .FDGH[X_LEN] GEQ  8  THEN EMIT_BYTES(OPC_CLRQ, M_AD+R_2)
;  1572          1622  2                              ELSE EMIT_BYTES(OPC_CLRL, M_AD+R_2);
;  1573          1623  2      TMP[-1] = .CUR_PC - .TMP;                           !2$:
;  1574          1624  2
;  1575          1625  2      ! Store the new datatype
;  1576          1626  2      ! Note that this is stored in it's normalized form.
;  1577          1627  2      !
;  1578          1628  2      KBF[KBF_TYPE] = DSC$K_DTYPE_B;
;  1579          1629  2      KBF[KBF_POSITION] = .DISP;
;  1580          1630  2      KBF[KBF_CVT] = TRUE;
;  1581          1631  2
;  1582          1632  2      ! Return the length in bytes of the converted keys.
;  1583          1633  2      !
;  1584          1634  2      RETURN .FDGH[X_LEN];
;  1585          1635  2
;  1586          1636  1      END;


                                    0070F          .BLKB   1
                                04  00710 OWN_FDGH:
                                                   .BYTE   4                       ;
                                00  00711           .BYTE   0                       ;
                              007F  00712           .WORD   127                     ;
                                08  00714           .BYTE   8                       ;
                                00  00715           .BYTE   0                       ;
                              007F  00716           .WORD   127                     ;
                                08  00718           .BYTE   8                       ;
                                00  00719           .BYTE   0                       ;
                              000F  0071A           .WORD   15                      ;
                                10  0071C           .BYTE   16                      ;
                                00  0071D           .BYTE   0                       ;
                              0000  0071E           .WORD   0                       ;


                                    003C 00000 GEN_CONVERT_FLT:
                                                   .WORD   Save R2,R3,R4,R5        ; 1496
                        55    FB7E  CF  9E 00002           MOVAB   EMIT_DISP, R5   ;
                        54      04  AC  D0 00007           MOVL    PKBF, KBF       ; 1562
                        53          09  D0 0000B           MOVL    #9, REG_SRC     ; 1563
03          02          A4          01  E1 0000E           BBC     #1, 2(KBF), 1$  ; 1564
                        53          0A  D0 00013           MOVL    #10, REG_SRC    ;
                        50      4A  8F  9A 00016 1$:       MOVZBL  #74, R0         ; 1577
                              0113  C5  16 0001A           JSB     ROOM            ;
                        50          64  3C 0001E           MOVZWL  (KBF), R0       ; 1581
```

```
                7E              00      50              01  7A 00021        EMUL    #1, R0, #0, -(SP)                       ;
                50              50      8E              05  7B 00026        EDIV    #5, (SP)+, R0, R0                       ;
                                        51        C1 AF40   DE 0002B        MOVAL   OWN_FDGH[R0], FDGH                      ;
                                        8A        9E      8F  90 00030        MOVB    #-98, (CUR_PC)+                         ; 1587
                                        52        04      A4  3C 00034        MOVZWL  4(KBF), R2                              ; 1588
                                                          65  16 00038        JSB     EMIT_DISP                               ;
                                        8A      9E51      8F  B0 0003A        MOVW    #-25007, (CUR_PC)+                      ; 1590
                                        52                61  9A 0003F        MOVZBL  (FDGH), R2                              ; 1591
                                        52        08      AC  C0 00042        ADDL2   DISP, R2                                ;
                                        53                0A  D0 00046        MOVL    #10, R3                                 ;
                                                          65  16 00049        JSB     EMIT_DISP                               ;
                                        8A  010FEE52      8F  D0 0004B        MOVL    #17821266, (CUR_PC)+                    ; 1594
                                        8A      5061      8F  B0 00052        MOVW    #20577, (CUR_PC)+                       ;
                                        8A        B1      8F  90 00057        MOVB    #-79, (CUR_PC)+                         ; 1598
                                        53        02      A1  3C 0005B        MOVZWL  2(FDGH), R3                             ; 1599
                                        52                02  D0 0005F        MOVL    #2, R2                                  ;
                                                  56      A5  16 00062        JSB     EMIT_LITE                               ;
                                        8A      1E61      8F  B0 00065        MOVW    #7777, (CUR_PC)+                        ; 1601
                                                          8A  94 0006A        CLRB    (CUR_PC)+                               ;
                                        52                5A  D0 0006C        MOVL    CUR_PC, TMP                             ; 1602
                                        50                61  9A 0006F        MOVZBL  (FDGH), R0                              ; 1606
                                        50                02  C6 00072        DIVL2   #2, R0                                  ;
                                                          07  11 00075        BRB     3$                                      ;
                                        8A  728150AD      8F  D0 00077 2$:     MOVL    #1921077421, (CUR_PC)+                  ; 1608
                                        F6                50  F4 0007E 3$:     SOBGEQ  I, 2$                                   ;
                                        8A  010750F0      8F  D0 00081        MOVL    #17256688, (CUR_PC)+                    ; 1614
                                        50                61  9A 00088        MOVZBL  (FDGH), R0                              ;
                                                50        50  08 78 0008B        ASHL    #8, R0, R0                              ;
                                        8A  0010FFA2      E0  9E 0008F        MOVAB   1114018(R0), (CUR_PC)+                  ;
                                    FF    A2        5A      52  83 00096        SUBB3   TMP, CUR_PC, -1(TMP)                    ; 1615
                                        52                5A  D0 0009B        MOVL    CUR_PC, TMP                             ; 1616
                                        10                61  91 0009E        CMPB    (FDGH), #16                             ; 1620
                                                          05  1F 000A1        BLSSU   4$                                      ;
                                        8A      727C      8F  B0 000A3        MOVW    #29308, (CUR_PC)+                       ;
                                        08                61  91 000A8 4$:     CMPB    (FDGH), #8                              ; 1621
                                                          07  1F 000AB        BLSSU   5$                                      ;
                                        6A      727C      8F  B0 000AD        MOVW    #29308, (CUR_PC)                        ;
                                                          05  11 000B2        BRB     6$                                      ;
                                        6A      72D4      8F  B0 000B4 5$:     MOVW    #29396, (CUR_PC)                        ; 1622
                                        5A                02  C0 000B9 6$:     ADDL2   #2, CUR_PC                              ;
                                    FF    A2        5A      52  83 000BC        SUBB3   TMP, CUR_PC, -1(TMP)                    ; 1623
                                        64                06  B0 000C1        MOVW    #6, (KBF)                               ; 1628
                                04      A4        08      AC  B0 000C4        MOVW    DISP, 4(KBF)                            ; 1629
                                02      A4                02  88 000C9        BISB2   #2, 2(KBF)                              ; 1630
                                        50                61  9A 000CD        MOVZBL  (FDGH), R0                              ; 1634
                                                          04  000D0        RET                                             ; 1636

; Routine Size:  209 bytes,    Routine Base:  SOR$RO_CODE + 0720
```

```
;1588     1637  1 %IF NOT HOSTILE %THEN
;1589     1638  1 ROUTINE GEN_CONVERT_UDEF
;1590     1639  1         (
;1591     1640  1         PKBF:           REF KBF_BLOCK,  ! Key description
;1592     1641  1         DISP                            ! Displacement from SRC2
;1593     1642  1         ):              LINK_COMPARE =
;1594     1643  1 !++
;1595     1644  1 ! Functional Description:
;1596     1645  1 !
;1597     1646  1 !       This routine generates code to convert user-defined key data types.
;1598     1647  1 !
;1599     1648  1 ! Formal Parameters:
;1600     1649  1 !
;1601     1650  1 !       PKBF            Address of the key description.
;1602     1651  1 !                       This is modified to reflect the new key description.
;1603     1652  1 !       DISP            Displacement from SRC2 of where to write the key.
;1604     1653  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;1605     1654  1 !
;1606     1655  1 ! Implicit Inputs:
;1607     1656  1 !
;1608     1657  1 !       None.
;1609     1658  1 !
;1610     1659  1 ! Implicit Outputs:
;1611     1660  1 !
;1612     1661  1 !       None.
;1613     1662  1 !
;1614     1663  1 ! Routine Value:
;1615     1664  1 !
;1616     1665  1 !       Length in bytes of the converted key.
;1617     1666  1 !
;1618     1667  1 ! Side Effects:
;1619     1668  1 !
;1620     1669  1 !       None.
;1621     1670  1 !
;1622     1671  1 !--
;1623     1672  2     BEGIN
;1624     1673  2     EXTERNAL REGISTER
;1625     1674  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK,
;1626     1675  2         CUR_PC = R_CUR_PC:      REF BLOCK,
;1627     1676  2         BRANCH = R_BRANCH:      REF VECTOR;
;1628     1677  2     EXTERNAL ROUTINE
;1629     1678  2         SOR$$DTYPE_KBF;
;1630     1679  2     LOCAL
;1631     1680  2         KBF:    REF KBF_BLOCK,          ! Key description
;1632     1681  2         REG_SRC,                        ! Source register
;1633     1682  2         CVTKBF: KBF_BLOCK,              ! Converted key description
;1634     1683  2         CVTRTN: INITIAL(0),             ! Conversion routine
;1635     1684  2         CMPRTN: INITIAL(0),             ! Comparison routine
;1636     1685  2         STATUS;
;1637     1686  2
;1638     1687  2     ! Check before calling SOR$$DTYPE_xxx
;1639     1688  2     !
;1640     1689  2     IF FUN_K_KANJI THEN 0 ELSE RETURN 0;
;1641     1690  2
;1642     1691  2     ! Normally, the source comes from the input record (COM_REG_SRC2).
;1643     1692  2     ! However, for indexed sorts, the key may have already been moved to the
;1644     1693  2     ! internal format record.  If so, use COM_REG_SRC2 as the input register.
```

```
;  1645          1694  2      !
;  1646          1695  2      KBF = PKBF[BASE_];
;  1647          1696  2      REG_SRC = COM_REG_SRC1;
;  1648          1697  2      IF .KBF[KBF_CVT] THEN REG_SRC = COM_REG_SRC2;
;  1649          1698  2
;  1650          1699  2      ! Analyze the key
;  1651          1700  2      !
;  1652          1701  2      CVTKBF[KBF_TYPE] = .KBF[KBF_TYPE];
;  1653          1702  2      CVTKBF[KBF_ORDER] = .KBF[KBF_ORDER];
;  1654          1703  2      CVTKBF[KBF_POSITION] = 0;
;  1655          1704  2      CVTKBF[KBF_LENGTH] = .KBF[KBF_LENGTH];
;  1656          1705  2      STATUS = SOR$$DTYPE_KBF(KBF[BASE_], CVTKBF[BASE_], CVTRTN, CMPRTN);
;  1657          1706  2      IF NOT .STATUS
;  1658          1707  2      THEN
;  1659          1708  3          BEGIN
;  1660          1709  3          SOR$$ERROR(SOR$_RTNERROR, 0, .STATUS);
;  1661          1710  3          KBF[KBF_TYPE] = DSC$K_DTYPE_Z;
;  1662          1711  3          KBF[KBF_LENGTH] = 0;
;  1663          1712  3          RETURN 0;
;  1664          1713  2          END;
;  1665          1714  2
;  1666          1715  2      ! Call the conv: sion routine
;  1667          1716  2      !
;  1668          1717  2      IF .CVTRTN NEQ 0
;  1669          1718  2      THEN
;  1670          1719  3          BEGIN
;  1671          1720  3          LOCAL
;  1672          1721  3              TMP: REF VECTOR[,BYTE];
;  1673          1722  3
;  1674          1723  3          ROOM(4+K_LITE+4+2+K_LITE+2+K_DISP+2+K_ABSA+8+K_LITE+2+K_ABSA);
;  1675          1724  3
;  1676          1725  3          ! Call the conversion routine
;  1677          1726  3          !
;  1678          1727  3          IF .REG_SRC EQL COM_REG_SRC1 AND
;  1679          1728  3              .KBF[KBF_LENGTH]+.KBF[KBF_POSITION] GTR .CTX[COM_SRL]
;  1680          1729  3          THEN
;  1681          1730  4              BEGIN
;  1682        P 1731  4              EMIT_BYTES(
;  1683        P 1732  4                  OPC_MOVZWL, M_R+R_6, M_R+R_0,          ! MOVZWL R6, R0
;  1684          1733  4                  OPC_SUBW2);                            ! SUBW2
;  1685          1734  4              EMIT_LITE(K_WORD, .KBF[KBF_POSITION]);     !       #srcoff,
;  1686        P 1735  4              EMIT_BYTES(M_R+R_0,                        !       R0
;  1687        P 1736  4                  OPC_BGEQU, 2,                          ! BGEQU 0$
;  1688          1737  4                  OPC_CLRW);                             ! CLRW
;  1689          1738  4              END
;  1690          1739  3          ELSE
;  1691          1740  4              BEGIN
;  1692          1741  4              EMIT_BYTES(OPC_MOVZWL);                    ! MOVZWL
;  1693          1742  4              EMIT_LITE(K_WORD, .KBF[KBF_LENGTH]);       !       #n
;  1694          1743  3              END;
;  1695        P 1744  3          EMIT_BYTES(M_R+R_0,                            !       R0
;  1696          1745  3              OPC_MOVAB);                                ! MOVAB
;  1697          1746  3          EMIT_DISP(.KBF[KBF_POSITION],.REG_SRC);        !       xx(Rsrc), R1
;  1698          1747  3          EMIT_BYTES(M_R+R_1, OPC_MOVZWL);               ! MOVZWL
;  1699          1748  3          EMIT_LITE(K_WORD, .CVTKBF[KBF_LENGTH]);        !       #n, R2
;  1700          1749  3          EMIT_BYTES(M_R+R_2, OPC_MOVAB);                ! MOVAB
;  1701          1750  3          EMIT_DISP(.DISP, COM_REG_SRC2);                !       xx(Rsrc2), R3
```

```
;  1702          1751  3          EMIT_BYTES(M_R+R_3, OPC_JSB);                   ! JSB
;  1703          1752  3          EMIT_ABSA(.CVTRTN);                             !       CVTRTN
;  1704          1753  3
;  1705          1754  3          ! Check the status
;  1706          1755  3          !
;  1707          1756  3          ! It is tempting to also check for SOR$_DELxxx codes, or to
;  1708          1757  3          ! delete the record if an error occurs.
;  1709          1758  3          !
;  1710          1759  3          TMP = .CUR_PC + 3;
;  1711        P 1760  3          EMIT_BYTES(OPC_BLBS, M_R+R_0, 0,                ! BLBS  R0, 1$
;  1712        P 1761  3                  OPC_PUSHL, M_R+R_0,                     ! PUSHL R0
;  1713        P 1762  3                  OPC_CLRL, M_AD+R_SP,                    ! CLRL  -(SP)
;  1714          1763  3                  OPC_PUSHL);                             ! PUSHL #SOR$_RTNERROR
;  1715          1764  3          EMIT_LITE(K_LONG, SOR$_RTNERROR);
;  1716          1765  3          EMIT_BYTES(OPC_CALLS, 3);                       ! CALLS #3
;  1717          1766  3          EMIT_ABSA(SOR$$ERROR);                          !       SOR$$ERROR
;  1718          1767  3          TMP[-1] = .CUR_PC - .TMP;                       ! Correct displacement
;  1719          1768  3
;  1720          1769  3          ! Store the new datatype
;  1721          1770  3          !
;  1722          1771  3          KBF[KBF_TYPE] = .CVTKBF[KBF_TYPE];
;  1723          1772  3          KBF[KBF_ORDER] = .CVTKBF[KBF_ORDER];
;  1724          1773  3          KBF[KBF_POSITION] = .DISP;
;  1725          1774  3          KBF[KBF_LENGTH] = .CVTKBF[KBF_LENGTH];
;  1726          1775  3          KBF[KBF_CVT] = TRUE;
;  1727          1776  2          END;
;  1728          1777  2
;  1729          1778  2      ! Convert to it's normalized form.
;  1730          1779  2      !
;  1731          1780  2      IF .KBF[KBF_TYPE] GTRU MAX_SUPPORTED THEN 0
;  1732          1781  2      ELIF .DSC_BINARY[.KBF[KBF_TYPE]]
;  1733          1782  2      THEN
;  1734          1783  3          BEGIN
;  1735        P 1784  3          IF ONEOF (.KBF[KBF_TYPE], BMSK_(DSC$K_DTYPE_BU,DSC$K_DTYPE_WU,
;  1736          1785  4              DSC$K_DTYPE_LU,DSC$K_DTYPE_QU,DSC$K_DTYPE_OU))
;  1737          1786  3          THEN KBF[KBF_TYPE] = DSC$K_DTYPE_BU
;  1738          1787  3          ELSE KBF[KBF_TYPE] = DSC$K_DTYPE_B;
;  1739          1788  2          END;
;  1740          1789  2
;  1741          1790  2      ! Return the length in bytes of the converted keys.
;  1742          1791  2      !
;  1743          1792  2      IF .CVTRTN NEQ 0
;  1744          1793  2      THEN
;  1745          1794  2          RETURN .KBF[KBF_LENGTH]
;  1746          1795  2      ELSE
;  1747          1796  2          RETURN 0;
;  1748          1797  2
;  1749          1798  1      END;


                                                .EXTRN  SOR$$DTYPE_KBF

                                01FC 00000 GEN_CONVERT_UDEF:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8       ; 1638
                    58      FB03    CF  9E 00002        MOVAB   EMIT_LITE, R8           ;
                    5E              08  C2 00007        SUBL2   #8, SP                  ;                ; R
```

```
                                 7E  7C 0000A          CLRQ    CMPRTN                                  ; 1672
                           47    00G E9 0000C          BLBC    S^FUN_K_KANJI, 2$                       ; 1689
                           54  04    AC  D0 0000F          MOVL    PKBF, KBF                               ; 1695
                           55        09  D0 00013          MOVL    #9, REG_SRC                             ; 1696
                     03    02    A4  01  E1 00016          BBC     #1, 2(KBF), 1$                          ; 1697
                           55        0A  D0 0001B          MOVL    #10, REG_SRC                            ;
                        08 AE        64  D0 0001E 1$:      MOVL    (KBF), CVTKBF                           ; 1701
                           0C    AE  B4  00022          CLRW    CVTKBF+4                                ; 1703
                           56  06    A4  9E 00025          MOVAB   6(KBF), R6                              ; 1704
                        0E AE        66  B0 00029          MOVW    (R6), CVTKBF+6                          ;
                                     5E  DD 0002D          PUSHL   SP                                      ; 1705
                                 08  AE  9F 0002F          PUSHAB  CVTRTN                                  ;
                                 10  AE  9F 00032          PUSHAB  CVTKBF                                  ;
                                     54  DD 00035          PUSHL   KBF                                     ;
                 00000000G       00      04  FB 00037          CALLS   #4, SOR$$DTYPE_KBF                      ;
                           18            50  E8 0003E          BLBS    STATUS, 3$                              ; 1706
                                         50  DD 00041          PUSHL   STATUS                                  ; 1709
                                         7E  D4 00043          CLRL    -(SP)                                   ;
                              001C812A   8F  DD 00045          PUSHL   #1868074                                ;
                 00000000G       00      03  FB 0004B          CALLS   #3, SOR$$ERROR                          ;
                                         64  B4 00052          CLRW    (KBF)                                   ; 1710
                                         66  B4 00054          CLRW    (R6)                                    ; 1711
                                     00F3    31  00056 2$:      BRW     10$                                     ; 1712
                                         57  D4 00059 3$:      CLRL    R7                                      ; 1717
                                 04  AE  D5  0005B          TSTL    CVTRTN                                  ;
                                     03  12  0005E          BNEQ    4$                                      ;
                                     00C2    31  00060          BRW     7$                                      ;
                                         57  D6 00063 4$:      INCL    R7                                      ;
                           50            36  D0 00065          MOVL    #54, R0                                 ; 1723
                                 00BD    C8  16 00068          JSB     ROOM                                    ;
                           09            55  D1 0006C          CMPL    REG_SRC, #9                             ; 1727
                                         2C  12 0006F          BNEQ    5$                                      ;
                           50            66  3C 00071          MOVZWL  (R6), R0                                ; 1728
                           51      04    A4  3C 00074          MOVZWL  4(KBF), R1                              ;
                           50            51  C0 00078          ADDL2   R1, R0                                  ;
                     50    0086  CB    10      ED  0007B          CMPZV   #0, #16, 134(CTX), R0                   ;
                                         19  18 00082          BGEQ    5$                                      ;
                           8A A250563C   8F  D0 00084          MOVL    #-1571793348, (CUR_PC)+                 ; 1733
                           53      04    A4  3C 0008B          MOVZWL  4(KBF), R3                              ; 1734
                           52            02  D0 0008F          MOVL    #2, R2                                  ;
                                         68  16 00092          JSB     EMIT_LITE                               ;
                           8A B4021E50   8F  D0 00094          MOVL    #-1274929584, (CUR_PC)+                 ; 1737
                                         0B  11 0009B          BRB     6$                                      ; 1727
                           8A            3C  90 0009D 5$:      MOVB    #60, (CUR_PC)+                          ; 1741
                           53            66  3C 000A0          MOVZWL  (R6), R3                                ; 1742
                           52            02  D0 000A3          MOVL    #2, R2                                  ;
                                         68  16 000A6          JSB     EMIT_LITE                               ;
                           8A      9E50  8F  B0 000A8 6$:      MOVW    #-25008, (CUR_PC)+                      ; 1745
                           53            55  D0 000AD          MOVL    REG_SRC, R3                             ; 1746
                           52      04    A4  3C 000B0          MOVZWL  4(KBF), R2                              ;
                                   AA    A8  16 000B4          JSB     EMIT_DISP                               ;
                           8A      3C51  8F  B0 000B7          MOVW    #15441, (CUR_PC)+                       ; 1747
                           53      0E    AE  3C 000BC          MOVZWL  CVTKBF+6, R3                            ; 1748
                           52            02  D0 000C0          MOVL    #2, R2                                  ;
                                         68  16 000C3          JSB     EMIT_LITE                               ;
                           8A      9E52  8F  B0 000C5          MOVW    #-25006, (CUR_PC)+                      ; 1749
                           53            0A  D0 000CA          MOVL    #10, R3                                 ; 1750
```

```
                                52        08    AC  D0 000CD          MOVL    DISP, R2                                    ;
                                          AA    A8  16 000D1          JSB     EMIT_DISP                                   ;
                                8A      1653    8F  B0 000D4          MOVW    #5715, (CUR_PC)+                            ; 1751
                                8A        9F    8F  90 000D9          MOVB    #-97, (CUR_PC)+                             ; 1752
                                8A        04    AE  D0 000DD          MOVL    CVTRTN, (CUR_PC)+                           ;
                                51        03    AA  9E 000E1          MOVAB   3(R10), TMP                                 ; 1759
                                8A  DD0050E8    8F  D0 000E5          MOVL    #-587181848, (CUR_PC)+                      ; 1763
                                8A  DD7ED450    8F  D0 000EC          MOVL    #-578890672, (CUR_PC)+                      ;
                                53  001C812A    8F  D0 000F3          MOVL    #1868074, R3                                ; 1764
                                52              04  D0 000FA          MOVL    #4, R2                                      ;
                                                68  16 000FD          JSB     EMIT_LITE                                   ;
                                8A      03FB    8F  B0 000FF          MOVW    #1019, (CUR_PC)+                            ; 1765
                                8A        9F    8F  90 00104          MOVB    #-97, (CUR_PC)+                             ; 1766
                                8A  00000000G   00  9E 00108          MOVAB   SOR$$ERROR, (CUR_PC)+                       ;
                    FF    A1                5A    51  83 0010F          SUBB3   TMP, CUR_PC, -1(TMP)                        ; 1767
                                64        08    AE  D0 00114          MOVL    CVTKBF, TKBF                                ; 1771
                                      04  A4    08  AC  B0 00118          MOVW    DISP, 4(KBF)                                ; 1773
                                66        0E    AE  B0 0011D          MOVW    CVTKBF+6, (R6)                              ; 1774
                                      02  A4        02  88 00121          BISB2   #2, 2(KBF)                                  ; 1775
                                23              64  B1 00125 7$:      CMPW    (KBF), #35                                  ; 1780
                                                1B  1A 00128          BGTRU   9$                                          ;
                                50              64  3C 0012A          MOVZWL  (KBF), R0                                   ; 1781
                            12      F709        CF  50 E1 0012D          BBC     R0, DSC_BINARY, 9$                          ;
                            50  3C000040        8F  64 78 00133          ASHL    (KBF), #1006633024, R0                      ; 1785
                                                    05 18 0013B          BGEQ    8$                                          ;
                                64              02  B0 0013D          MOVW    #2, (KBF)                                   ; 1786
                                                    03 11 00140          BRB     9$                                          ;
                                64              06  B0 00142 8$:      MOVW    #6, (KBF)                                   ; 1787
                                04              57  E9 00145 9$:      BLBC    R7, 10$                                     ; 1792
                                50              66  3C 00148          MOVZWL  (R6), R0                                    ; 1796
                                                    04 0014B          RET                                                 ;
                                                50  D4 0014C 10$:      CLRL    R0                                          ; 1798
                                                    04 0014E          RET                                                 ;

; Routine Size:  335 bytes,    Routine Base:  SOR$RO_CODE + 07F1

; 1750          1799  1 %FI
```

```
;  1752          1800  1 ROUTINE GEN_COMPARE
;  1753          1801  1         (
;  1754          1802  1         PKBF:           REF KBF_BLOCK,          ! Key description
;  1755          1803  1         INDEX                                   ! Number of the key
;  1756          1804  1         ):      NOVALUE LINK_COMPARE =
;  1757          1805  1 !++
;  1758          1806  1 ! Functional Description:
;  1759          1807  1 !
;  1760          1808  1 !       This routine generates a single key compare.
;  1761          1809  1 !
;  1762          1810  1 ! Formal Parameters:
;  1763          1811  1 !
;  1764          1812  1 !       PKBF            Address of the key description.
;  1765          1813  1 !       INDEX           Index of the key, 0 indicates the first key
;  1766          1814  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1767          1815  1 !                               (used only for COM_COLLATE)
;  1768          1816  1 !
;  1769          1817  1 ! Implicit Inputs:
;  1770          1818  1 !
;  1771          1819  1 !       None.
;  1772          1820  1 !
;  1773          1821  1 ! Implicit Outputs:
;  1774          1822  1 !
;  1775          1823  1 !       None.
;  1776          1824  1 !
;  1777          1825  1 ! Routine Value:
;  1778          1826  1 !
;  1779          1827  1 !       Status code.
;  1780          1828  1 !
;  1781          1829  1 ! Side Effects:
;  1782          1830  1 !
;  1783          1831  1 !       None.
;  1784          1832  1 !
;  1785          1833  1 !--
;  1786          1834  2     BEGIN
;  1787          1835  2     EXTERNAL REGISTER
;  1788          1836  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK,
;  1789          1837  2         CUR_PC = R_CUR_PC:      REF BLOCK,
;  1790          1838  2         BRANCH = R_BRANCH:      REF VECTOR;
;  1791          1839  2     LITERAL
;  1792          1840  2         K_MAXDEC =      31;     ! Maximum length of decimal data
;  1793          1841  2     MACRO
;  1794        M 1842  2         CMP_(OPC, SU, OFF) =
;  1795        M 1843  2             BEGIN
;  1796        M 1844  2             %IF %NAME('OPC_',OPC) LEQU %X'FF'
;  1797        M 1845  2             %THEN EMIT_BYTE(%NAME('OPC_',OPC))
;  1798        M 1846  2             %ELSE EMIT_WORD(%NAME('OPC_',OPC))
;  1799        M 1847  2             %FI;
;  1800        M 1848  2             OPOPNEQ(
;  1801        M 1849  2                 .KBF[KBF_POSITION] %IF NOT %NULL(OFF) %THEN + OFF %FI,
;  1802        M 1850  2                 %NAME('K_',SU)+.KBF[KBF_ORDER]);
;  1803          1851  2             END %;
;  1804          1852  2
;  1805          1853  2     LOCAL
;  1806          1854  2         KBF:    REF KBF_BLOCK;          ! Local copy of pointer to key
;  1807          1855  2
;  1808          1856  2     KBF = PKBF[BASE_];
```

```
;  1809          1857  2
;  1810          1858  2      ! Case on the datatype to generate code
;  1811          1859  2      !
;  1812          1860  2      ROOM(2+K_OPOPNEQ);
;  1813          1861  2      CASE .KBF[KBF_TYPE] FROM 0 TO 28 OF
;  1814          1862  2          SET
;  1815          1863  2          [DSC$K_DTYPE_T]: IF .KBF[KBF_LENGTH] GTRU 0 THEN
;  1816        C 1864  2 %(
;  1817        C 1865  2              ! This section of code can be used to compare implementations
;  1818        C 1866  2              ! of the DEC_MULTINATIONAL collating sequence.
;  1819        C 1867  2              !
;  1820        C 1868  2              IF
;  1821        C 1869  2                  BEGIN
;  1822        C 1870  2                  LOCAL
;  1823        C 1871  2                      LOG:    VECTOR[2],
;  1824        C 1872  2                      RSL:    VECTOR[2],
;  1825        C 1873  2                      BUF:    VECTOR[1,BYTE],
;  1826        C 1874  2                      STATUS;
;  1827        C 1875  2                  LOG[0] = %CHARCOUNT('STR$COMPARE_MULTI');
;  1828        C 1876  2                  LOG[1] = UPLIT BYTE('STR$COMPARE_MULTI');
;  1829        C 1877  2                  RSL[0] = %ALLOCATION(BUF);
;  1830        C 1878  2                  RSL[1] = BUF[0];
;  1831        C 1879  2                  $TRNLOG(LOGNAM=LOG[0], RSLBUF=RSL[0]) EQL SS$_NORMAL
;  1832        C 1880  2                  END
;  1833        C 1881  2              THEN
;  1834        C 1882  2                  BEGIN
;  1835        C 1883  2                  EXTERNAL ROUTINE
;  1836        C 1884  2                      STR$COMPARE_MULTI:      ADDRESSING_MODE(GENERAL);
;  1837        C 1885  2                  ROOM(2*(1+K_DISP+1+K_LITE)+12+K_ABSA+5+K_BNEQ);
;  1838        C 1886  2                  EMIT_BYTE(OPC_PUSHAB);
;  1839        C 1887  2                  EMIT_DISP(.KBF[KBF_POSITION], COM_REG_SRC1);        ! xx(Rsrc1)
;  1840        C 1888  2                  EMIT_BYTE(OPC_PUSHL);
;  1841        C 1889  2                  EMIT_LITE(K_LONG, .KBF[KBF_LENGTH]);
;  1842        C 1890  2                  EMIT_BYTE(OPC_PUSHAB);
;  1843        C 1891  2                  EMIT_DISP(.KBF[KBF_POSITION], COM_REG_SRC2);        ! xx(Rsrc2)
;  1844        C 1892  2                  EMIT_BYTE(OPC_PUSHL);
;  1845        C 1893  2                  EMIT_LITE(K_LONG, .KBF[KBF_LENGTH]);
;  1846        C 1894  2                  EMIT_BYTES(OPC_PUSHL, 1, OPC_CLRL, M_AD+R_SP,
;  1847        C 1895  2                      OPC_PUSHAB, M_BD+R_SP, 8,
;  1848        C 1896  2                      OPC_PUSHAB, M_BD+R_SP, 20,
;  1849        C 1897  2                      OPC_CALLS, 4);
;  1850        C 1898  2                  EMIT_ABSA(STR$COMPARE_MULTI);
;  1851        C 1899  2                  EMIT_BYTES(OPC_ADDL2, 16, M_R+R_SP, OPC_TSTL, M_R+R_0);
;  1852        C 1900  2                  EMIT_BNEQ(K_S+.KBF[KBF_ORDER]);
;  1853        C 1901  2                  END
;  1854        C 1902  2              ELSE
;  1855          1903  2 )%
;  1856          1904  3                  BEGIN
;  1857          1905  3                  ROOM(K_SAVE_REGS+1+K_ABSA+1+
;  1858          1906  3                      MAX(1+K_LITE+K_DISP+K_DISP+5+K_BNEQ,
;  1859          1907  3                          1+K_LITE+2+K_DISP+5+K_DISP+4+K_BNEQ));
;  1860          1908  3                  SAVE_REGS(%B'111110');                              ! Save R1..R5
;  1861          1909  3                  EMIT_BYTE(OPC_MOVAB);
;  1862          1910  3                  EMIT_ABSA(.CTX[COM_COLLATE]);
;  1863          1911  3                  EMIT_BYTE(M_R+R_5);
;  1864          1912  3                  IF .VECTOR[.CTX[COM_COLLATE],1] NEQ 0
;  1865          1913  3                  THEN
```

```
;  1866          1914  4                BEGIN
;  1867          1915  4                LOCAL TMP: REF VECTOR[,BYTE];
;  1868          1916  4                EMIT_BYTE(OPC_CMPC3);
;  1869          1917  4                EMIT_LITE(K_WORD, .KBF[KBF_LENGTH]);
;  1870          1918  4                EMIT_DISP(.KBF[KBF_POSITION], COM_REG_SRC1);    ! xx(Rsrc1)
;  1871          1919  4                EMIT_DISP(.KBF[KBF_POSITION], COM_REG_SRC2);    ! xx(Rsrc2)
;  1872          1920  4                EMIT_BYTES(OPC_BEQL, 0);
;  1873          1921  4                TMP = .CUR_PC;
;  1874          1922  4                EMIT_BYTES(OPC_JSB, M_BDD+R_5, 4);
;  1875          1923  4                EMIT_BNEQ(K_S+.KBF[KBF_ORDER]);
;  1876          1924  4                TMP[-1] = .CUR_PC - .TMP;
;  1877          1925  4                END
;  1878          1926  3            ELSE
;  1879          1927  4                BEGIN
;  1880          1928  4                EMIT_BYTE(OPC_MOVZWL);
;  1881          1929  4                EMIT_LITE(K_WORD, .KBF[KBF_LENGTH]);
;  1882          1930  4                EMIT_BYTES(M_R+R_0, OPC_MOVAB);
;  1883          1931  4                EMIT_DISP(.KBF[KBF_POSITION], COM_REG_SRC1);    ! xx(Rsrc1)
;  1884          1932  4                EMIT_BYTES(M_R+R_1, OPC_MOVL, M_R+R_0, M_R+R_2, OPC_MOVAB);
;  1885          1933  4                EMIT_DISP(.KBF[KBF_POSITION], COM_REG_SRC2);    ! xx(Rsrc2)
;  1886          1934  4                EMIT_BYTES(M_R+R_3, OPC_JSB, M_BDD+R_5, 0);
;  1887          1935  4                !
;  1888          1936  4                ! This routine returns R0 = -1, 0, or +1.
;  1889          1937  4                ! The resultant condition codes of calling this routine
;  1890          1938  4                ! are equivalent to those from MOVL R0, R0.
;  1891          1939  4                ! Thus, we want to do a signed branch.
;  1892          1940  4                !
;  1893          1941  4                EMIT_BNEQ(K_S+.KBF[KBF_ORDER]);
;  1894          1942  3                END;
;  1895          1943  2            END;
;  1896          1944  2
;  1897          1945  2        [DSC$K_DTYPE_Z,
;  1898          1946  2         DSC$K_DTYPE_NU]: IF .KBF[KBF_LENGTH] GTRU 0 THEN
;  1899          1947  3            BEGIN
;  1900          1948  3            LITERAL
;  1901          1949  3                TUN_K_BYTE_COMPARE = TRUE;
;  1902          1950  3            ROOM(1+K_OPOPNEQ+K_SAVE_REGS+1+K_LITE+K_OPOPNEQ);
;  1903          1951  3            IF TUN_K_BYTE_COMPARE THEN CMP_(CMPB, U);
;  1904          1952  3            IF .KBF[KBF_LENGTH] GTRU TUN_K_BYTE_COMPARE
;  1905          1953  3            THEN
;  1906          1954  4                BEGIN
;  1907          1955  4                SAVE_REGS(%B'1110');                ! Save R1..R3
;  1908          1956  4                EMIT_BYTE(OPC_CMPC3);
;  1909          1957  4                EMIT_LITE(K_WORD, .KBF[KBF_LENGTH]-TUN_K_BYTE_COMPARE);
;  1910          1958  4                OPOPNEQ(.KBF[KBF_POSITION]+TUN_K_BYTE_COMPARE, K_U+.KBF[KBF_ORDER]);
;  1911          1959  4                END
;  1912          1960  2            END;
;  1913          1961  2        [DSC$K_DTYPE_B,
;  1914          1962  2         DSC$K_DTYPE_BU]:
;  1915          1963  3            BEGIN
;  1916          1964  3            BIND
;  1917          1965  3                CMP = UPLIT BYTE(OPC_CMPB, OPC_CMPW, OPC_CMPL): VECTOR[,BYTE];
;  1918          1966  3            LOCAL
;  1919          1967  3                L, Z, SU;
;  1920          1968  3            L = .KBF[KBF_LENGTH];
;  1921          1969  3            ROOM((2+.L^-2)*(1+K_OPOPNEQ));
;  1922          1970  3            SU = K_U + .KBF[KBF_ORDER];
```

```
;  1923          1971  3            IF .KBF[KBF_TYPE] EQL DSC$K_DTYPE_B THEN SU = .SU - K_U + K_S;
;  1924          1972  3            DECR I FROM 2 TO 0 DO WHILE (Z = .L - 1^.I) GEQ 0 DO
;  1925          1973  4                BEGIN
;  1926          1974  4                EMIT_BYTE(.CMP[.I]);
;  1927          1975  4                OPOPNEQ(
;  1928          1976  4                    .KBF[KBF_POSITION] + .Z,
;  1929          1977  4                    .SU);
;  1930          1978  4                L = .Z;
;  1931          1979  4                SU = K_U + .KBF[KBF_ORDER];     ! Other compares are unsigned
;  1932          1980  3                END;
;  1933          1981  2            END;
;  1934          1982  2        [DSC$K_DTYPE_F]:        CMP_(CMPF, S);
;  1935          1983  2        [DSC$K_DTYPE_D]:        CMP_(CMPD, S);
;  1936          1984  2        [DSC$K_DTYPE_G]:        CMP_(CMPG, S);
;  1937          1985  2        [DSC$K_DTYPE_H]:        CMP_(CMPH, S);
;  1938          1986  2        [DSC$K_DTYPE_P]:
;  1939          1987  3            BEGIN
;  1940          1988  3            ROOM(K_SAVE_REGS+1+K_LITE+K_OPOPNEQ);
;  1941          1989  3            SAVE_REGS(%B'1110');                    ! Save R1..R3
;  1942          1990  3            EMIT_BYTE(OPC_CMPP3);
;  1943          1991  3            EMIT_LITE(K_WORD, .KBF[KBF_LENGTH]);
;  1944          1992  3            OPOPNEQ(.KBF[KBF_POSITION], K_S+.KBF[KBF_ORDER]);
;  1945          1993  2            END;
;  1946          1994  2
;  1947          1995  2        [OUTRANGE]:
;  1948          1996  2 %IF NOT HOSTILE %THEN
;  1949          1997  2            IF FUN_K_KANJI          ! Check before calling SOR$$DTYPE_xxx
;  1950          1998  2            THEN
;  1951          1999  3                BEGIN
;  1952          2000  3                EXTERNAL ROUTINE
;  1953          2001  3                    SOR$$DTYPE_KBF;
;  1954          2002  3                LOCAL
;  1955          2003  3                    CVTKBF: KBF_BLOCK,
;  1956          2004  3                    CVTRTN: INITIAL(0),
;  1957          2005  3                    CMPRTN: INITIAL(0),
;  1958          2006  3                    STATUS;
;  1959          2007  3                STATUS = SOR$$DTYPE_KBF(KBF[BASE_], CVTKBF[BASE_], CVTRTN, CMPRTN);
;  1960          2008  3                IF NOT .STATUS THEN SOR$$ERROR(SOR$_RTNERROR, 0, .STATUS);
;  1961          2009  3                IF .CMPRTN EQL 0
;  1962          2010  3                THEN
;  1963          2011  3                    RETURN SOR$$ERROR(SOR$_SHR_BADLOGIC);
;  1964          2012  3                ROOM(K_SAVE_REGS+1+K_LITE+5+K_DISP+2+K_DISP+2+K_ABSA+2+K_BNEQ);
;  1965          2013  3                SAVE_REGS(%B'111111');                  ! Save R0..R5
;  1966          2014  3                EMIT_BYTES(OPC_MOVZWL);
;  1967          2015  3                EMIT_LITE(K_WORD, .KBF[KBF_LENGTH]);
;  1968          2016  3                EMIT_BYTES(M_R+R_0, OPC_MOVL, M_R+R_0, M_R+R_2, OPC_MOVAB);
;  1969          2017  3                EMIT_DISP(.KBF[KBF_POSITION], COM_REG_SRC1);        ! xx(Rsrc1)
;  1970          2018  3                EMIT_BYTES(M_R+R_1, OPC_MOVAB);
;  1971          2019  3                EMIT_DISP(.KBF[KBF_POSITION], COM_REG_SRC2);        ! xx(Rsrc2)
;  1972          2020  3                EMIT_BYTES(M_R+R_3, OPC_JSB);
;  1973          2021  3                EMIT_ABSA(.CMPRTN);
;  1974          2022  3                EMIT_BYTES(OPC_TSTL, M_R+R_0);
;  1975          2023  3                EMIT_BNEQ(K_S+.KBF[KBF_ORDER]);
;  1976          2024  3                END
;  1977          2025  2            ELSE
;  1978          2026  2 %FI
;  1979          2027  2                RETURN SOR$$ERROR(SOR$_SHR_BADLOGIC);
```

```
;  1980          2028  2
;  1981          2029  2          [INRANGE]:
;  1982          2030  2              RETURN SOR$$ERROR(SOR$_SHR_BADLOGIC);
;  1983          2031  2          TES;
;  1984          2032  2
;  1985          2033  2      RETURN;
;  1986          2034  1      END;


                                        D1  B1  91  00940 P.AAG:  .BYTE   -111, -79, -47                          ;
                                                          CMP=            P.AAG


                                                  01FC 00000 GEN_COMPARE:
                                                             .WORD   Save R2,R3,R4,R5,R6,R7,R8                    ; 1800
                                          5E          10  C2 00002        SUBL2   #16, SP                         ;
                                          55      04  AC  D0 00005        MOVL    PKBF, KBF                       ; 1856
                                          50          1E  D0 00009        MOVL    #30, R0                         ; 1860
                                                  FA65    30 0000C        BSBW    ROOM                            ;
                                    1C            00    65  AF 0000F        CASEW   (KBF), #0, #28                  ; 1861
                0232          019F          0232        015E     00013 1$:      .WORD   9$-1$,-                     ;
                0232          019F          0232        0232     0001B          26$-1$,-                            ;
                01F1          01EB          0232        0232     00023          12$-1$,-                            ;
                015E          00BC          0232        0232     0002B          26$-1$,-                            ;
                0232          0232          0232        0232     00033          26$-1$,-                            ;
                0232          0232          020A        0232     0003B          26$-1$,-                            ;
                01F9          0232          0232        0232     00043          12$-1$,-                            ;
                                                        0200     0004B          26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                17$-1$,-                            ;
                                                                                18$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                5$-1$,-                             ;
                                                                                9$-1$,-                             ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                23$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                26$-1$,-                            ;
                                                                                20$-1$,-                            ;
                                                                                21$-1$                              ;
                                          03            00G E8 0004D          BLBS    S^FUN_K_KANJI, 3$           ; 1997
                                                    01F2  31 00050 2$:          BRW     26$                         ;
                                                      6E  7C 00053 3$:          CLRQ    CMPRTN                      ; 1999
                                                      5E  DD 00055              PUSHL   SP                          ; 2007
                                                  09  AE  9F 00057              PUSHAB  CVTRTN                      ;
                                                  10  AE  9F 0005A              PUSHAB  CVTKBF                      ;
```

```
                        55  DD 0005D        PUSHL   KBF                             ;
00000000G  00           04  FB 0005F        CALLS   #4, SOR$$DTYPE_KBF              ;
           11           50  E8 00066        BLBS    STATUS, 4$                      ; 2008
                        50  DD 00069        PUSHL   STATUS                          ;
                        7E  D4 0006B        CLRL    -(SP)                           ;
             001C812A   8F  DD 0006D        PUSHL   #1868074                        ;
00000000G  00           03  FB 00073        CALLS   #3, SOR$$ERROR                  ;
                        6E  D5 0007A 4$:    TSTL    CMPRTN                          ; 2009
                        D2  13 0007C        BEQL    2$                              ;
           50           35  D0 0007E        MOVL    #53, R0                         ; 2012
                      F9F0  30 00081        BSBW    ROOM                            ;
           54           3F  D0 00084        MOVL    #63, R4                         ; 2013
                      F79F  30 00087        BSBW    SAVE_REGS                       ;
           8A           3C  90 0008A        MOVB    #60, (CUR_PC)+                  ; 2014
           53       06  A5  3C 0008D        MOVZWL  6(KBF), R3                      ; 2015
           52           02  D0 00091        MOVL    #2, R2                          ;
                      F920  30 00094        BSBW    EMIT_LITE                       ;
           8A 5250D050  8F  D0 00097        MOVL    #1381027920, (CUR_PC)+          ; 2016
           8A       9E  8F  90 0009E        MOVB    #-98, (CUR_PC)+                 ;
           53           09  D0 000A2        MOVL    #9, R3                          ; 2017
           52       04  A5  3C 000A5        MOVZWL  4(KBF), R2                      ;
                      F8B5  30 000A9        BSBW    EMIT_DISP                       ;
           8A     9E51  8F  B0 000AC        MOVW    #-25007, (CUR_PC)+              ; 2018
           53           0A  D0 000B1        MOVL    #10, R3                         ; 2019
           52       04  A5  3C 000B4        MOVZWL  4(KBF), R2                      ;
                      F8A6  30 000B8        BSBW    EMIT_DISP                       ;
           8A     1653  8F  B0 000BB        MOVW    #5715, (CUR_PC)+                ; 2020
           8A       9F  8F  90 000C0        MOVB    #-97, (CUR_PC)+                 ; 2021
           8A           6E  D0 000C4        MOVL    CMPRTN, (CUR_PC)+               ;
           8A     50D5  8F  B0 000C7        MOVW    #20693, (CUR_PC)+               ; 2022
                      0097  31 000CC        BRW     8$                              ; 2023
           53       06  A5  3C 000CF 5$:    MOVZWL  6(KBF), R3                      ; 1863
                        01  12 000D3        BNEQ    6$                              ;
                            04 000D5        RET                                     ;
           50           37  D0 000D6 6$:    MOVL    #55, R0                         ; 1905
                      F998  30 000D9        BSBW    ROOM                            ;
           54           3E  D0 000DC        MOVL    #62, R4                         ; 1908
                      F747  30 000DF        BSBW    SAVE_REGS                       ;
           8A     9F9E  8F  B0 000E2        MOVW    #40862, (CUR_PC)+               ; 1909
           50       68  AB  D0 000E7        MOVL    104(CTX), R0                    ; 1910
           8A           50  D0 000EB        MOVL    R0, (CUR_PC)+                   ;
           8A       55  8F  90 000EE        MOVB    #85, (CUR_PC)+                  ; 1911
                    04  A0  D5 000F2        TSTL    4(R0)                           ; 1912
                        3B  13 000F5        BEQL    7$                              ;
           8A           29  90 000F7        MOVB    #41, (CUR_PC)+                  ; 1916
           52           02  D0 000FA        MOVL    #2, R2                          ; 1917
                      F8B7  30 000FD        BSBW    EMIT_LITE                       ;
           53           09  D0 00100        MOVL    #9, R3                          ; 1918
           52       04  A5  3C 00103        MOVZWL  4(KBF), R2                      ;
                      F857  30 00107        BSBW    EMIT_DISP                       ;
           53           0A  D0 0010A        MOVL    #10, R3                         ; 1919
           52       04  A5  3C 0010D        MOVZWL  4(KBF), R2                      ;
                      F84D  30 00111        BSBW    EMIT_DISP                       ;
           8A           13  B0 00114        MOVW    #19, (CUR_PC)+                  ; 1920
           51           5A  D0 00117        MOVL    CUR_PC, TMP                     ; 1921
           8A     B516  8F  B0 0011A        MOVW    #-19178, (CUR_PC)+              ; 1922
           8A           04  90 0011F        MOVB    #4, (CUR_PC)+                   ;
```

```
                        54        02    A5  3C 00122          MOVZWL  2(KBF), R4                      ; 1923
                        54              02  C0 00126          ADDL2   #2, R4                          ;
                                      F753  30 00129          BSBW    EMIT_BNEQ                       ;
                    FF  A1              5A  83 0012C          SUBB3   TMP, CUR_PC, -1(TMP)            ; 1924
                                        04  00131            RET                                     ; 1912
                                        8A  90 00132 7$:      MOVB    #60, (CUR_PC)+                  ; 1928
                                        52  D0 00135          MOVL    #2, R2                          ; 1929
                                      F87C  30 00138          BSBW    EMIT_LITE                       ;
                                      8A    9E50  8F  B0 0013B  MOVW    #-25008, (CUR_PC)+              ; 1930
                                        53        09  D0 00140  MOVL    #9, R3                          ; 1931
                                        52        04  A5 3C 00143  MOVZWL  4(KBF), R2                  ;
                                      F817  30 00147          BSBW    EMIT_DISP                       ;
                                      8A 5250D051   8F  D0 0014A  MOVL    #1381027921, (CUR_PC)+          ; 1932
                                      8A    9E      8F  90 00151  MOVB    #-98, (CUR_PC)+                 ;
                                        53        0A  D0 00155  MOVL    #10, R3                         ; 1933
                                        52        04  A5 3C 00158  MOVZWL  4(KBF), R2                  ;
                                      F802  30 0015C          BSBW    EMIT_DISP                       ;
                                      8A 00B51653   8F  D0 0015F  MOVL    #11887731, (CUR_PC)+            ; 1934
                                        54        02  A5 3C 00166 8$:  MOVZWL  2(KBF), R4              ; 1941
                                        54              02  C0 0016A  ADDL2   #2, R4                  ;
                                      F70F  30 0016D          BSBW    EMIT_BNEQ                       ;
                                        04  00170            RET                                     ; 1863
                                        53        06  A5 3C 00171 9$:  MOVZWL  6(KBF), R3             ; 1946
                                              01    12 00175          BNEQ    10$                     ;
                                        04  00177            RET                                     ;
                                        50        42  8F  9A 00178 10$:  MOVZBL  #66, R0               ; 1950
                                      F8F5  30 0017C          BSBW    ROOM                            ;
                                      8A    91  8F  90 0017F  MOVB    #-111, (CUR_PC)+                ; 1951
                                        54        02  A5 3C 00183  MOVZWL  2(KBF), R4                  ;
                                        52        04  A5 3C 00187  MOVZWL  4(KBF), R2                  ;
                                      F809  30 0018B          BSBW    OPOPNEQ                         ;
                                        01              53  B1 0018E  CMPW    R3, #1                  ; 1952
                                              01    1A 00191          BGTRU   11$                     ;
                                        04  00193            RET                                     ;
                                        54              0E  D0 00194 11$:  MOVL    #14, R4              ; 1955
                                      F68F  30 00197          BSBW    SAVE_REGS                       ;
                                        8A              29  90 0019A  MOVB    #41, (CUR_PC)+          ; 1956
                                                        53  D7 0019D  DECL    R3                      ; 1957
                                        52              02  D0 0019F  MOVL    #2, R2                  ;
                                      F812  30 001A2          BSBW    EMIT_LITE                       ;
                                        52        04  A5 3C 001A5  MOVZWL  4(KBF), R2                  ; 1958
                                                  52  D6 001A9  INCL    R2                          ;
                                        54        02  A5 3C 001AB  MOVZWL  2(KBF), R4                  ;
                                      008F  31 001AF          BRW     25$                             ;
                                        53        06  A5 3C 001B2 12$:  MOVZWL  6(KBF), L             ; 1968
                                    50  53        FE  8F  78 001B6  ASHL    #-2, L, R0                  ; 1969
                                    50              1D  C4 001BB  MULL2   #29, R0                     ;
                                    50              3A  C0 001BE  ADDL2   #58, R0                     ;
                                      F8B0  30 001C1          BSBW    ROOM                            ;
                                        54        02  A5 3C 001C4  MOVZWL  2(KBF), SU                 ; 1970
                                        06              65  B1 001C8  CMPW    (KBF), #6               ; 1971
                                                        03  12 001CB  BNEQ    13$                     ;
                                        54              02  C0 001CD  ADDL2   #2, SU                  ;
                                        58        04  A5 9E 001D0 13$:  MOVAB   4(KBF), R8            ; 1976
                                        51              02  D0 001D4  MOVL    #2, I                   ;
                                    57  01              51  78 001D7 14$:  ASHL    I, #1, R7               ; 1972
                                    56  53              57  C3 001DB 15$:  SUBL3   R7, L, Z               ;
```

```
                        19  19 001DF          BLSS    16$                                     ;
                8A  FE17 CF41  90 001E1          MOVB    CMP[I], (CUR_PC)+                       ; 1974
                50       68  3C 001E7          MOVZWL  (R8), R0                                ; 1976
        52      50        56  C1 001EA          ADDL3   Z, R0, R2                               ;
                      F7A6  30 001EE          BSBW    OPOPNEQ                                 ;
                53        56  D0 001F1          MOVL    Z, L                                    ; 1978
                54     02 A5  3C 001F4          MOVZWL  2(KBF), SU                              ; 1979
                           E1  11 001F8          BRB     15$                                     ; 1972
                DA         51  F4 001FA 16$:     SOBGEQ  I, 14$                                  ;
                               04 001FD          RET                                             ; 1861
                6A     51  8F  90 001FE 17$:     MOVB    #81, (CUR_PC)                           ; 1982
                           04  11 00202          BRB     19$                                     ;
                6A     71  8F  90 00204 18$:     MOVB    #113, (CUR_PC)                          ; 1983
                           5A  D6 00208 19$:     INCL    CUR_PC                                  ;
                           2A  11 0020A          BRB     24$                                     ;
                6A   51FD  8F  B0 0020C 20$:     MOVW    #20989, (CUR_PC)                        ; 1984
                           05  11 00211          BRB     22$                                     ;
                6A   71FD  8F  B0 00213 21$:     MOVW    #29181, (CUR_PC)                        ; 1985
                5A         02  C0 00218 22$:     ADDL2   #2, CUR_PC                              ;
                           19  11 0021B          BRB     24$                                     ;
                50         25  D0 0021D 23$:     MOVL    #37, R0                                 ; 1988
                      F851  30 00220          BSBW    ROOM                                    ;
                54         0E  D0 00223          MOVL    #14, R4                                 ; 1989
                      F600  30 00226          BSBW    SAVE_REGS                               ;
                8A         35  90 00229          MOVB    #53, (CUR_PC)+                          ; 1990
                53     06  A5  3C 0022C          MOVZWL  6(KBF), R3                              ; 1991
                52         02  D0 00230          MOVL    #2, R2                                  ;
                      F781  30 00233          BSBW    EMIT_LITE                               ;
                54     02  A5  3C 00236 24$:     MOVZWL  2(KBF), R4                              ; 1992
                54         02  C0 0023A          ADDL2   #2, R4                                  ;
                52     04  A5  3C 0023D          MOVZWL  4(KBF), R2                              ;
                      F753  30 00241 25$:     BSBW    OPOPNEQ                                 ;
                               04 00244          RET                                             ; 1861
                    001C1124  8F  DD 00245 26$:     PUSHL   #1839396                                ; 2030
        00000000G   00            01  FB 0024B          CALLS   #1, SOR$$ERROR                  ;
                               04 00252          RET                                             ; 2034

; Routine Size:  595 bytes,    Routine Base:  SOR$RO_CODE + 0943
```

```
;1988   2035  1 ROUTINE MOVE_KEYS
;1989   2036  1         (
;1990   2037  1         KEY_BUFF:       REF KEY_BLOCK,          ! Key descriptions buffer
;1991   2038  1         DISP                                    ! Displacement from SRC2
;1992   2039  1         ):      LINK_COMPARE =
;1993   2040  1 !++
;1994   2041  1 ! Functional Description:
;1995   2042  1 !
;1996   2043  1 !       This routine generates code to save unconverted keys
;1997   2044  1 !       (for non-record sorts).
;1998   2045  1 !
;1999   2046  1 ! Formal Parameters:
;2000   2047  1 !
;2001   2048  1 !       KEY_BUFF        Address of the key descriptions buffer.
;2002   2049  1 !                       This is modified to reflect the new key descriptions.
;2003   2050  1 !       DISP            Displacement from SRC2 of where to write the key.
;2004   2051  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;2005   2052  1 !
;2006   2053  1 ! Implicit Inputs:
;2007   2054  1 !
;2008   2055  1 !       None.
;2009   2056  1 !
;2010   2057  1 ! Implicit Outputs:
;2011   2058  1 !
;2012   2059  1 !       None.
;2013   2060  1 !
;2014   2061  1 ! Routine Value:
;2015   2062  1 !
;2016   2063  1 !       Length in bytes of the copied keys.
;2017   2064  1 !
;2018   2065  1 ! Side Effects:
;2019   2066  1 !
;2020   2067  1 !       None.
;2021   2068  1 !
;2022   2069  1 !--
;2023   2070  2     BEGIN
;2024   2071  2     EXTERNAL REGISTER
;2025   2072  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK,
;2026   2073  2         CUR_PC = R_CUR_PC:      REF BLOCK;
;2027   2074  2     LOCAL
;2028   2075  2         BUFF:   REF KEY_BLOCK,  ! Local copy of KEY_BUFF
;2029   2076  2         CVTCNT;                 ! Number of bytes copied by this routine
;2030   2077  2     BUFF = KEY_BUFF[BASE_];
;2031   2078  2
;2032   2079  2     ! Analyze the unconverted keys to determine which bytes must be copied
;2033   2080  2     !
;2034   2081  2     CVTCNT = 0;
;2035   2082  2     WHILE TRUE DO
;2036   2083  3         BEGIN
;2037   2084  3         LOCAL
;2038   2085  3             LOPOS,
;2039   2086  3             HIPOS;
;2040   2087  3
;2041   2088  3         ! Find the first byte containing an unconverted key.
;2042   2089  3         !
;2043   2090  3         LOPOS = -1;
;2044   2091  3         DECR I FROM .BUFF[KEY_NUMBER]-1 TO 0 DO
```

```
;2045    2092  4          BEGIN
;2046    2093  4          BIND Y = BUFF[KEY_KBF(.I)]: KBF_BLOCK;
;2047    2094  4          IF NOT .Y[KBF_CVT] AND .Y[KBF_POSITION] LSSU .LOPOS
;2048    2095  4          THEN
;2049    2096  4              LOPOS = .Y[KBF_POSITION];
;2050    2097  3          END;
;2051    2098  3
;2052    2099  3      IF .LOPOS LSS 0 THEN EXITLOOP;          ! Exit of no more found
;2053    2100  3
;2054    2101  3      ! While we are finding unconverted keys that overlap this key,
;2055    2102  3      ! take the overlap of the two pieces.
;2056    2103  3      !
;2057    2104  3      HIPOS = .LOPOS;
;2058    2105  3      WHILE TRUE DO
;2059    2106  4          BEGIN
;2060    2107  4          LOCAL FOUND;
;2061    2108  4          FOUND = FALSE;
;2062    2109  4          DECR I FROM .BUFF[KEY_NUMBER]-1 TO 0 DO
;2063    2110  5              BEGIN
;2064    2111  5              BIND Y = BUFF[KEY_KBF(.I)]: KBF_BLOCK;
;2065    2112  5              IF NOT .Y[KBF_CVT] AND .Y[KBF_POSITION] LEQ .HIPOS
;2066    2113  5              THEN
;2067    2114  6                  BEGIN
;2068    2115  6                  FOUND = TRUE;
;2069    2116  6                  HIPOS = MAX(.HIPOS, .Y[KBF_POSITION] + LEN_(Y[BASE_]));
;2070    2117  6                  Y[KBF_CVT] = TRUE;
;2071    2118  6                  Y[KBF_POSITION] = .Y[KBF_POSITION]+.DISP+.CVTCNT-.LOPOS;
;2072    2119  5                  END;
;2073    2120  4              END;
;2074    2121  4          IF NOT .FOUND THEN EXITLOOP;
;2075    2122  3          END;
;2076    2123  3
;2077    2124  3      ! We found everything that overlaps this piece, so allocate it
;2078    2125  3      !
;2079    2126  3      IF .HIPOS NEQ .LOPOS            ! Check for zero length
;2080    2127  3      THEN
;2081    2128  4          BEGIN
;2082    2129  4          ROOM(K_MOVE);
;2083    2130  4          GEN_MOVE_VAR(.HIPOS - .LOPOS, .LOPOS, COM_REG_SRC1,
;2084    2131  4              .DISP + .CVTCNT, COM_REG_SRC2);
;2085    2132  4          CVTCNT = .CVTCNT + .HIPOS - .LOPOS;
;2086    2133  3          END;
;2087    2134  2      END;
;2088    2135  2
;2089    2136  2  RETURN .CVTCNT;     ! Return the number of bytes this routine converted
;2090    2137  1  END;


                                01FC 00000 MOVE_KEYS:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8                    ;2035
                    56      04  AC  D0 00002       MOVL    KEY_BUFF, BUFF                      ;2077
                                55  D4 00006       CLRL    CVTCNT                              ;2081
                    57          01  CE 00008 1$:    MNEGL   #1, LOPOS                           ;2090
                    51          66  3C 0000B       MOVZWL  (BUFF), I                           ;2091
```

```
                                          16    11 0000E           BRB     3$                                    ;
                                50     02 A641  7E 00010 2$:       MOVAQ   2(BUFF)[I], R0                        ; 2093
                  0C      02    A0          01  E0 00015           BBS     #1, 2(R0), 3$                         ; 2094
     57      04   A0            10          00  ED 0001A           CMPZV   #0, #16, 4(R0), LOPOS                 ;
                                            04  1E 00020           BGEQU   3$                                    ;
                                57     04   A0  3C 00022           MOVZWL  4(R0), LOPOS                          ; 2096
                                E7          51  F4 00026 3$:       SOBGEQ  I, 2$                                 ; 2091
                                            57  D5 00029           TSTL    LOPOS                                 ; 2099
                                            03  18 0002B           BGEQ    4$                                    ;
                                          008C  31 0002D           BRW     12$                                   ;
                                52          57  D0 00030 4$:       MOVL    LOPOS, HIPOS                          ; 2104
                                            58  D4 00033 5$:       CLRL    FOUND                                 ; 2108
                                54          66  3C 00035           MOVZWL  (BUFF), I                             ; 2109
                                            52  11 00038           BRB     10$                                   ;
                                50     02 A644  7E 0003A 6$:       MOVAQ   2(BUFF)[I], R0                        ; 2111
                  48      02    A0          01  E0 0003F           BBS     #1, 2(R0), 10$                        ; 2112
     52      04   A0            10          00  ED 00044           CMPZV   #0, #16, 4(R0), HIPOS                 ;
                                            40  14 0004A           BGTR    10$                                   ;
                                58          01  D0 0004C           MOVL    #1, FOUND                             ; 2115
                                15          60  B1 0004F           CMPW    (R0), #21                             ; 2116
                                            0B  12 00052           BNEQ    7$                                    ;
                                51     06   A0  3C 00054           MOVZWL  6(R0), R1                             ;
                                51          02  C6 00058           DIVL2   #2, R1                                ;
                                            51  D6 0005B           INCL    R1                                    ;
                                            04  11 0005D           BRB     8$                                    ;
                                51     06   A0  3C 0005F 7$:       MOVZWL  6(R0), R1                             ;
                                53     04   A0  3C 00063 8$:       MOVZWL  4(R0), R3                             ;
                                51          53  C0 00067           ADDL2   R3, R1                                ;
                                53          52  D0 0006A           MOVL    HIPOS, R3                             ;
                                51          53  D1 0006D           CMPL    R3, R1                                ;
                                            03  18 00070           BGEQ    9$                                    ;
                                53          51  D0 00072           MOVL    R1, R3                                ;
                                52          53  D0 00075 9$:       MOVL    R3, HIPOS                             ;
                          02    A0          02  88 00078           BISB2   #2, 2(R0)                             ; 2117
                                51     04   A0  3C 0007C           MOVZWL  4(R0), R1                             ; 2118
                                51     08   AC  C0 00080           ADDL2   DISP, R1                              ;
                                51          55  C0 00084           ADDL2   CVTCNT, R1                            ;
             04   A0            51          57  A3 00087           SUBW3   LOPOS, R1, 4(R0)                      ;
                                AB          54  F4 0008C 10$:      SOBGEQ  I, 6$                                 ; 2109
                                A1          58  E8 0008F           BLBS    FOUND, 5$                             ; 2121
                                57          52  D1 00092           CMPL    HIPOS, LOPOS                          ; 2126
                                            22  13 00095           BEQL    11$                                   ;
                                50     42   8F  9A 00097           MOVZBL  #66, R0                               ; 2129
                                          F783  30 0009B           BSBW    ROOM                                  ;
                                            0A  DD 0009E           PUSHL   #10                                   ; 2130
                                       08 BC45  9F 000A0           PUSHAB  @DISP[CVTCNT]                         ; 2131
                                            09  DD 000A4           PUSHL   #9                                    ; 2130
                                            57  DD 000A6           PUSHL   LOPOS                                 ;
                  7E            52          57  C3 000A8           SUBL3   LOPOS, HIPOS, -(SP)                   ;
                        F7FB    CF          05  FB 000AC           CALLS   #5, GEN_MOVE_VAR                      ;
                  50            55          52  C1 000B1           ADDL3   HIPOS, CVTCNT, R0                     ; 2132
                  55            50          57  C3 000B5           SUBL3   LOPOS, R0, CVTCNT                     ;
                                          FF4C  31 000B9 11$:      BRW     1$                                    ; 2082
                                50          55  D0 000BC 12$:      MOVL    CVTCNT, R0                            ; 2136
                                                04 000BF           RET                                           ; 2137

; Routine Size:  192 bytes,    Routine Base:  SOR$RO_CODE + 0B96
```

```
;  2092          2138  1 ROUTINE EXPAND
;  2093          2139  1         (
;  2094          2140  1         ORD,
;  2095          2141  1         CNT,
;  2096          2142  1         DISP:   REF VECTOR
;  2097          2143  1         ):      NOVALUE =
;  2098          2144  1 !++
;  2099          2145  1 ! Functional Description:
;  2100          2146  1 !
;  2101          2147  1 !       This routine adds CNT bytes to the field referenced by ORD.
;  2102          2148  1 !
;  2103          2149  1 ! Formal Parameters:
;  2104          2150  1 !
;  2105          2151  1 !       ORD     Index of the field to be expanded
;  2106          2152  1 !       CNT     Number of bytes by which to expand the field
;  2107          2153  1 !       DISP    Address of the displacements table
;  2108          2154  1 !
;  2109          2155  1 ! Implicit Inputs:
;  2110          2156  1 !
;  2111          2157  1 !       None.
;  2112          2158  1 !
;  2113          2159  1 ! Implicit Outputs:
;  2114          2160  1 !
;  2115          2161  1 !       None.
;  2116          2162  1 !
;  2117          2163  1 ! Routine Value:
;  2118          2164  1 !
;  2119          2165  1 !       None.
;  2120          2166  1 !
;  2121          2167  1 ! Side Effects:
;  2122          2168  1 !
;  2123          2169  1 !       None.
;  2124          2170  1 !
;  2125          2171  1 !--
;  2126          2172  2     BEGIN
;  2127          2173  2
;  2128          2174  2     ! Move all the following fields down.
;  2129          2175  2     ! Also, if this field hasn't been allocated yet, allocate it.
;  2130          2176  2     !
;  2131          2177  2     INCR I FROM .ORD+1 TO COM_ORD_MAX DO
;  2132          2178  3         BEGIN
;  2133          2179  3         IF .DISP[.I] GEQ 0
;  2134          2180  3         THEN
;  2135          2181  4             BEGIN
;  2136          2182  4             IF .DISP[.ORD] LSS 0 THEN DISP[.ORD] = .DISP[.I];
;  2137          2183  4             DISP[.I] = .DISP[.I] + .CNT;
;  2138          2184  3             END;
;  2139          2185  2         END;
;  2140          2186  2
;  2141          2187  1     END;


                                  0004 00000 EXPAND:  .WORD   Save R2                                 ; 2138
                        51    04    AC  D0 00002        MOVL   ORD, R1                               ; 2177
```

```
                    50              51  D0 00006          MOVL    R1, I                           ; 2179
                                    18  11 00009          BRB     3$
                    52      0C BC40     DE 0000B 1$:      MOVAL   @DISP[I], R2                    ;
                                    62  D5 00010          TSTL    (R2)                            ;
                                    0F  19 00012          BLSS    3$                              ;
                            0C BC41     D5 00014          TSTL    @DISP[R1]                       ; 2182
                                    05  18 00018          BGEQ    2$                              ;
                        0C BC41     62  D0 0001A          MOVL    (R2), @DISP[R1]                 ;
                            62      08  AC  C0 0001F 2$:   ADDL2   CNT, (R2)                      ; 2183
                E4              50      09  F3 00023 3$:   AOBLEQ  #9, I, 1$                      ; 2177
                                        04  00027          RET                                     ; 2187

; Routine Size:  40 bytes,    Routine Base:  SOR$RO_CODE + 0C56
```

```
;  2143          2188  1 GLOBAL ROUTINE SOR$$KEY_SUB
;  2144          2189  1         (
;  2145          2190  1         KEY_BUFFER:     REF KEY_BLOCK
;  2146          2191  1         ):      CAL_CTXREG =
;  2147          2192  1
;  2148          2193  1 !++
;  2149          2194  1 ! Functional Description:
;  2150          2195  1 !
;  2151          2196  1 !       This routine generates the key comparison routines, and optionally,
;  2152          2197  1 !       an input conversion routine and an output conversion routine.
;  2153          2198  1 !
;  2154          2199  1 !       The key comparison routine returns one of the following values:
;  2155          2200  1 !               -1 if the first record collates before the second record
;  2156          2201  1 !                0 if the records collate equal
;  2157          2202  1 !                1 if the first record collates after the second record
;  2158          2203  1 !
;  2159          2204  1 ! Formal Parameters:
;  2160          2205  1 !
;  2161          2206  1 !       KEY_BUFFER      Address of a counted list of key descriptions.
;  2162          2207  1 !
;  2163          2208  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  2164          2209  1 !
;  2165          2210  1 !       The following fields of the context area are used as input:
;  2166          2211  1 !               COM_SORT_TYPE   Type of sort (TYP_K_RECORD, etc)
;  2167          2212  1 !               COM_NUM_FILES   Number of input files
;  2168          2213  1 !               COM_LRL         Longest input record length (see below)
;  2169          2214  1 !               COM_MINVFC      Length of VFC area
;  2170          2215  1 !               COM_COLLATE     Collating sequence information
;  2171          2216  1 !               COM_HACK_STRIP  Flag to do key stripping
;  2172          2217  1 !               COM_STABLE      Flag indicating stable sort
;  2173          2218  1 !               COM_VAR         Flag indicating variable-length records
;  2174          2219  1 !               COM_NO_DUPS     Flag indicating to delete duplicate records
;  2175          2220  1 !               COM_MERGE       Indicates what to store for stable
;  2176          2221  1 !               COM_PAD         Pad character
;  2177          2222  1 !
;  2178          2223  1 !       The following fields are used as input/output:
;  2179          2224  1 !               COM_COMPARE     Comparison routine
;  2180          2225  1 !               COM_EQUAL       Equal-key routine
;  2181          2226  1 !               COM_TKS         Total key size (hack hack)
;  2182          2227  1 !
;  2183          2228  1 !       The following fields are used as output:
;  2184          2229  1 !               COM_INPUT       Routine to do input conversion of records
;  2185          2230  1 !               COM_LENADR      Routine to return length/address of record
;  2186          2231  1 !               COM_SRL         Shortest allowable input record length
;  2187          2232  1 !               COM_LRL_INT     Length of internal format record
;  2188          2233  1 !
;  2189          2234  1 !       Actually, the COM_LRL field may be increased if key stripping is being
;  2190          2235  1 !       done, since the longest record length specified by the user doesn't
;  2191          2236  1 !       account for the extra bytes he's put at the beginning of the record.
;  2192          2237  1 !
;  2193          2238  1 ! Implicit Inputs:
;  2194          2239  1 !
;  2195          2240  1 !       None.
;  2196          2241  1 !
;  2197          2242  1 ! Implicit Outputs:
;  2198          2243  1 !
;  2199          2244  1 !       None.
```

```
;2200        2245  1 !
;2201        2246  1 ! Routine Value:
;2202        2247  1 !
;2203        2248  1 !       Status code.
;2204        2249  1 !
;2205        2250  1 ! Side Effects:
;2206        2251  1 !
;2207        2252  1 !       None.
;2208        2253  1 !
;2209        2254  1 !--
;2210        2255  1
;2211        2256  2     BEGIN
;2212        2257  2     EXTERNAL REGISTER
;2213        2258  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK;
;2214        2259  2
;2215        2260  2     GLOBAL REGISTER
;2216        2261  2         CUR_PC = R_CUR_PC:      REF BLOCK,      ! PC of code we're generating
;2217        2262  2         BRANCH = R_BRANCH:      REF VECTOR;     ! Address of branches table
;2218        2263  2
;2219        2264  2     LITERAL
;2220        2265  2         RTN_SIZE =      128;                    ! Initial routine size
;2221        2266  2
;2222        2267  2     LOCAL
;2223        2268  2         BRANCHES:       VECTOR[BR_SIZE],        ! Branch addresses
;2224        2269  2         STACK,                                  ! Stack needed for input routine
;2225        2270  2         TMP:    REF VECTOR[,BYTE],              ! Temporary pointer to code
;2226        2271  2         DISP:   VECTOR[COM_ORD_MAX+1],          ! Field displacements
;2227        2272  2         KEY_BUFF:       KEY_BLOCK;              ! Space to save the key info
;2228        2273  2
;2229        2274  2     MACRO
;2230        2275  2         ! Check for writing too far
;2231        2276  2         !
;2232      M 2277  2         VERIFY_LEN(A) =
;2233      M 2278  2             IF .CUR_PC GTRA
;2234      M 2279  2                 .VECTOR[ CTX[COM_ROUTINES], 0 ] +
;2235      M 2280  2                 .VECTOR[ CTX[COM_ROUTINES], 1 ]
;2236      M 2281  2             THEN
;2237        2282  2                 RETURN SOR$$ERROR(SOR$_SHR_BADLOGIC) %,
;2238        2283  2
;2239        2284  2         ! Allocate some bytes in a field
;2240        2285  2         !
;2241      M 2286  2         EXPAND_(A,B) =
;2242        2287  2             EXPAND(%NAME('COM_ORD_',A), B, DISP[0]) %;
;2243        2288  2
;2244        2289  2
;2245        2290  2     ! Get a local copy of the key description buffer, since we may mung it.
;2246        2291  2     ! The presence of a user-comparison routine indicates no key buffer.
;2247        2292  2     ! If we don't have one or the other, default to the whole record.
;2248        2293  2     !
;2249        2294  2     IF KEY_BUFFER[BASE_] NEQ 0
;2250        2295  2     THEN
;2251        2296  3         BEGIN
;2252        2297  3         LOCAL
;2253        2298  3             LEN;
;2254        2299  3         LEN = 2 + .KEY_BUFFER[KEY_NUMBER] * KBF_K_SIZE;
;2255        2300  3         IF .LEN GTR %A[LOCATION(KEY_BUFF) THEN RETURN SOR$$ERROR(SOR$_NUM_KEY);
;2256        2301  3         CH$MOVE(.LEN, KEY_BUFFER[BASE_], KEY_BUFF[BASE_]);
```

```
;  2257          2302  3          END
;  2258          2303  2      ELIF
;  2259          2304  2          .CTX[COM_COMPARE] NEQ 0
;  2260          2305  2      THEN
;  2261          2306  2          KEY_BUFF[KEY_NUMBER] =  0                ! No key descriptions
;  2262          2307  2      ELSE
;  2263          2308  3          BEGIN
;  2264          2309  3          BIND
;  2265          2310  3              KBF = KEY_BUFF[KEY_KBF(0)]: KBF_BLOCK;
;  2266          2311  3          KEY_BUFF[KEY_NUMBER] =  1;               ! One key
;  2267          2312  3          KBF[KBF_TYPE] =         DSC$K_DTYPE_T;   ! Text
;  2268          2313  3          KBF[KBF_ORDER] =        0;               ! Ascending
;  2269          2314  3          KBF[KBF_POSITION] =     0;               ! Start at the beginning
;  2270          2315  3          KBF[KBF_LENGTH] =       .CTX[COM_LRL];   ! For the length of the record
;  2271          2316  2          END;
```

```
;  2273     2317  2 !+
;  2274     2318  2 !
;  2275     2319  2 ! Analyze the keys, et al.
;  2276     2320  2 ! Decide whether RFA,FILE,STAB,VFC,FORM,VAR and DATA (all but KEY) fields are
;  2277     2321  2 ! present.  If present, compute the displacement to the field.
;  2278     2322  2 !
;  2279     2323  2 !-
;  2280     2324  2
;  2281     2325  2
;  2282     2326  2     ! Versions V3 (and earlier) stripped keys before passing the record to the
;  2283     2327  2     ! key comparison routine, or returning the record from the sort.
;  2284     2328  2     ! For compatability, we must do the same (ugh).  See SORT_MERGE for
;  2285     2329  2     ! details on setting COM_HACK_STRIP (requests stripping).
;  2286     2330  2     !
;  2287     2331  2     ! Note that this is done before we analyze the keys.  Otherwise, we may not
;  2288     2332  2     ! strip enough bytes, due to keys being dropped or shortened.
;  2289     2333  2     !
;  2290     2334  2     TKS_HACK(KEY_BUFF[BASE_]);
;  2291     2335  2
;  2292     2336  2
;  2293     2337  2     ! A -1 in the displacements table indicates field not present
;  2294     2338  2     !
;  2295     2339  2     CH$FILL(%X'FF', COM_ORD_MAX * %UPVAL, DISP[0]);
;  2296     2340  2     DISP[COM_ORD_MAX] = 0;
;  2297     2341  2
;  2298     2342  2
;  2299     2343  2     ! RFA needed?
;  2300     2344  2     !
;  2301     2345  2     ! We need the RFA (and possibly file number) for non-record sorts.
;  2302     2346  2     !
;  2303     2347  2     IF .CTX[COM_SORT_TYPE] NEQ TYP_K_RECORD
;  2304     2348  2     THEN
;  2305     2349  3         BEGIN
;  2306     2350  3         EXPAND_('RFA', RAB$S_RFA);
;  2307     2351  3         IF .CTX[COM_NUM_FILES] GTRU 1
;  2308     2352  3         THEN
;  2309     2353  3             EXPAND_('FILE', 1);
;  2310     2354  2         END;
;  2311     2355  2
;  2312     2356  2
;  2313     2357  2     ! DATA, Record length, and VFC area needed?
;  2314     2358  2     !
;  2315     2359  2     ! We need the data portion for record sorts.
;  2316     2360  2     ! We need the length for variable-length records.
;  2317     2361  2     ! We may also need the VFC area.
;  2318     2362  2     !
;  2319     2363  2     IF .CTX[COM_SORT_TYPE] EQL TYP_K_RECORD
;  2320     2364  2     THEN
;  2321     2365  3         BEGIN
;  2322     2366  3         EXPAND_('DATA', .CTX[COM_LRL]);          !***
;  2323     2367  3         IF .CTX[COM_MINVFC] NEQ 0 THEN EXPAND_('VFC', .CTX[COM_MINVFC]);
;  2324     2368  3         IF .CTX[COM_VAR] THEN EXPAND_('VAR', 2);          ! A word for length
;  2325     2369  3         END
;  2326     2370  2     ELSE
;  2327     2371  3         BEGIN
;  2328     2372  3         !
;  2329     2373  3         ! If we don't have the data, don't call user-written routines.
```

```
;  2330     2374  3          !
;  2331     2375  3          IF .CTX[COM_COMPARE] NEQ 0 OR .CTX[COM_EQUAL] NEQ 0
;  2332     2376  3          THEN
;  2333     2377  3              RETURN SOR$$ERROR(SOR$_BAD_TYPE);
;  2334     2378  2          END;
;  2335     2379  2
;  2336     2380  2
;  2337     2381  2      ! Record format needed?
;  2338     2382  2      !
;  2339     2383  2      ! Needed if we have more than one record format.
;  2340     2384  2      !
;  2341     2385  2      IF .CTX[COM_FORMATS] GTRU 1
;  2342     2386  2      THEN
;  2343     2387  2          EXPAND_('FORM', 1);
;  2344     2388  2
;  2345     2389  2
;  2346     2390  2      ! Record number needed?
;  2347     2391  2      !
;  2348     2392  2      ! If a stable sort, use a longword to save the record number.
;  2349     2393  2      !
;  2350     2394  2      IF .CTX[COM_STABLE]
;  2351     2395  2      THEN
;  2352     2396  2          EXPAND_('STAB', 4);
;  2353     2397  2
;  2354     2398  2
;  2355     2399  2      ! Verify the keys, unless the user has his own comparison routine.
;  2356     2400  2      !
;  2357     2401  2      IF .CTX[COM_COMPARE] EQL 0
;  2358     2402  2      THEN
;  2359     2403  3          BEGIN
;  2360     2404  3
;  2361     2405  3          ! Loop through each key
;  2362     2406  3          !
;  2363     2407  3          DECR I FROM .KEY_BUFF[KEY_NUMBER]-1 TO 0 DO
;  2364     2408  4              BEGIN
;  2365     2409  4              LOCAL
;  2366     2410  4                  KEYLEN,                         ! Length of this key
;  2367     2411  4                  KBF:    REF KBF_BLOCK;          ! Local copy of key
;  2368     2412  4
;  2369     2413  4
;  2370     2414  4              ! Grab a local pointer to the key description buffer
;  2371     2415  4              !
;  2372     2416  4              KBF = KEY_BUFF[KEY_KBF(.I)];
;  2373     2417  4
;  2374     2418  4
;  2375     2419  4              ! Check the validity of the ascending/descending flag
;  2376     2420  4              !
;  2377     2421  4              IF .KBF[KBF_ORDER] GTRU 1
;  2378     2422  4              THEN
;  2379     2423  5                  BEGIN
;  2380     2424  5                  SOR$$ERROR(SOR$_BAD_KEY);
;  2381     2425  5                  KBF[KBF_ORDER] = .KBF[KBF_ORDER] AND NOT 1;
;  2382     2426  4                  END;
;  2383     2427  4
;  2384     2428  4
;  2385     2429  4              ! Check the validity of the length
;  2386     2430  4              !
```

```
;  2387          2431  5          IF  BEGIN
;  2388          2432  5              IF .KBF[KBF_TYPE] GTRU MAX_SUPPORTED
;  2389          2433  5              THEN %IF NOT HOSTILE %THEN FUN_K_KANJI %ELSE FALSE %FI
;  2390          2434  5              ELIF .KBF[KBF_LENGTH] GTRU .DSC_LENGTH[.KBF[KBF_TYPE]]
;  2391          2435  5              THEN FALSE
;  2392          2436  5              ELIF .DSC_FORCE[.KBF[KBF_TYPE]]
;  2393          2437  5              THEN
;  2394          2438  6                  BEGIN
;  2395          2439  6                  IF .KBF[KBF_LENGTH] EQL 0
;  2396          2440  6                  THEN
;  2397          2441  6                      KBF[KBF_LENGTH] = .DSC_LENGTH[.KBF[KBF_TYPE]];
;  2398          2442  6                  .KBF[KBF_LENGTH] EQL .DSC_LENGTH[.KBF[KBF_TYPE]]
;  2399          2443  6                  END
;  2400          2444  5              ELSE
;  2401          2445  5                  TRUE
;  2402          2446  5              END
;  2403          2447  4          THEN
;  2404          2448  4              0
;  2405          2449  4          ELSE
;  2406          2450  5              BEGIN
;  2407          2451  5              SOR$$ERROR(SOR$_BAD_KEY);
;  2408          2452  5              KBF[KBF_TYPE] = DSC$K_DTYPE_Z;
;  2409          2453  5              KBF[KBF_LENGTH] = 0;
;  2410          2454  4              END;
;  2411          2455  4
;  2412          2456  4
;  2413          2457  4          ! Compute the length in bytes of this key
;  2414          2458  4          !
;  2415          2459  4          KEYLEN = LEN_(KBF[BASE_]);
;  2416          2460  4
;  2417          2461  4
;  2418          2462  4          ! Check that the key fits within the longest record length
;  2419          2463  4          !
;  2420          2464  4          IF .KEYLEN + .KBF[KBF_POSITION] GTR .CTX[COM_LRL]
;  2421          2465  4          THEN
;  2422          2466  5              BEGIN
;  2423          2467  5              !
;  2424          2468  5              ! Part of the key extends past the longest record length.
;  2425          2469  5              ! Shorten string keys, and ignore all other keys.
;  2426          2470  5              !
;  2427          2471  5              IF   .KBF[KBF_TYPE] EQL DSC$K_DTYPE_Z OR
;  2428          2472  5                   .KBF[KBF_TYPE] EQL DSC$K_DTYPE_T OR
;  2429          2473  5                   .KBF[KBF_TYPE] GTRU MAX_SUPPORTED
;  2430          2474  5              THEN
;  2431          2475  6                  BEGIN
;  2432          2476  6                  KEYLEN = .CTX[COM_LRL] - .KBF[KBF_POSITION];
;  2433          2477  6                  IF .KEYLEN LSS 0 THEN KEYLEN = 0;   ! Don't get negative
;  2434          2478  6                  END
;  2435          2479  5              ELSE
;  2436          2480  6                  BEGIN
;  2437          2481  6                  KBF[KBF_TYPE] = DSC$K_DTYPE_Z;
;  2438          2482  6                  KEYLEN = 0;
;  2439          2483  5                  END;
;  2440          2484  5
;  2441          2485  5              ! Complain about the error.
;  2442          2486  5              ! If the entire key disappeared, make it a worse error.
;  2443          2487  5              !
```

```
;  2444          2488  5              ! Special-case a length of -1.
;  2445          2489  5              !
;  2446          2490  5              IF  .KBF[KBF_LENGTH] NEQ 1^%FIELDEXPAND(KBF_LENGTH,2)-1 OR
;  2447          2491  5                  .KEYLEN EQL 0
;  2448          2492  5              THEN
;  2449          2493  5                  SOR$$ERROR(
;  2450          2494  6                  (IF .KEYLEN EQL 0
;  2451          2495  6                  THEN SOR$_KEY_LEN AND NOT STS$M_SEVERITY OR STS$K_ERROR
;  2452          2496  5                  ELSE SOR$_KEY_LEN AND NOT STS$M_SEVERITY OR STS$K_INFO),
;  2453          2497  5                  2, .I+1, .KBF[KBF_LENGTH]);
;  2454          2498  5
;  2455          2499  5              ! Store the newly computed key length
;  2456          2500  5              !
;  2457          2501  5              KBF[KBF_LENGTH] = .KEYLEN;
;  2458          2502  4              END;
;  2459          2503  4
;  2460          2504  4
;  2461          2505  4          ! Determine the shortest record length that contains all keys
;  2462          2506  4          ! However, string keys are okay.
;  2463          2507  4          !
;  2464          2508  4          IF  .CTX[COM_SRL] LSS .KEYLEN + .KBF[KBF_POSITION] AND
;  2465          2509  4              .KBF[KBF_TYPE] NEQ DSC$K_DTYPE_T AND
;  2466          2510  4              .KBF[KBF_TYPE] NEQ DSC$K_DTYPE_Z AND
;  2467          2511  4              .KBF[KBF_TYPE] LEQU MAX_SUPPORTED
;  2468          2512  4          THEN
;  2469          2513  4              CTX[COM_SRL] = .KEYLEN + .KBF[KBF_POSITION];
;  2470          2514  4
;  2471          2515  4
;  2472          2516  4          ! For index sorts, allocate space for the original keys.
;  2473          2517  4          !
;  2474          2518  4          IF .CTX[COM_SORT_TYPE] EQL TYP_K_INDEX
;  2475          2519  4          THEN
;  2476          2520  4              EXPAND_('OKEY', .KEYLEN);
;  2477          2521  4
;  2478          2522  4
;  2479          2523  4          ! Convert keys to a normalized form; namely, the signed or
;  2480          2524  4          ! unsigned byte equivalents, with the appropriate length; change
;  2481          2525  4          ! datatype T to Z if there is no collating sequence.
;  2482          2526  4          !
;  2483          2527  4          IF .KBF[KBF_TYPE] EQL DSC$K_DTYPE_T AND
;  2484          2528  4              .CTX[COM_COLLATE] EQL 0
;  2485          2529  4          THEN
;  2486          2530  4              KBF[KBF_TYPE] = DSC$K_DTYPE_Z;
;  2487          2531  4          IF .KBF[KBF_TYPE] EQL DSC$K_DTYPE_Z
;  2488          2532  4          THEN
;  2489          2533  5              BEGIN
;  2490          2534  5              IF .KBF[KBF_LENGTH] EQL 1 THEN KBF[KBF_TYPE] = DSC$K_DTYPE_BU;
;  2491          2535  5              END
;  2492          2536  4          ELIF .KBF[KBF_TYPE] GTRU MAX_SUPPORTED
;  2493          2537  4          THEN
;  2494          2538  4              0
;  2495          2539  4          ELIF .DSC_BINARY[.KBF[KBF_TYPE]]
;  2496          2540  4          THEN
;  2497          2541  5              BEGIN
;  2498        P 2542  5              IF ONEOF (.KBF[KBF_TYPE], BMSK_(DSC$K_DTYPE_BU,DSC$K_DTYPE_WU,
;  2499          2543  6                  DSC$K_DTYPE_LU,DSC$K_DTYPE_QU,DSC$K_DTYPE_OU))
;  2500          2544  5              THEN KBF[KBF_TYPE] = DSC$K_DTYPE_BU
```

```
;  2501          2545  5                 ELSE KBF[KBF_TYPE] = DSC$K_DTYPE_B;
;  2502          2546  4                 END;
;  2503          2547  4
;  2504          2548  3             END;
;  2505          2549  3
;  2506          2550  2         END;
;  2507          2551  2
;  2508          2552  2
;  2509          2553  2     ! Try to do some key compression.
;  2510          2554  2     !
;  2511          2555  2     KEY_COMPRESS(KEY_BUFF[BASE_]);
;  2512          2556  2
;  2513          2557  2
;  2514          2558  2     ! Initialize code descriptor, and the current PC
;  2515          2559  2     !
;  2516          2560  2     VECTOR[ CTX[COM_ROUTINES], 0 ] = 0;
;  2517          2561  2     VECTOR[ CTX[COM_ROUTINES], 1 ] = CUR_PC = 0;
```

```
;  2519     2562  2 !+
;  2520     2563  2 !
;  2521     2564  2 ! Generate the input conversion routine
;  2522     2565  2 !
;  2523     2566  2 !-
;  2524     2567  2
;  2525     2568  2     !+
;  2526     2569  2     !
;  2527     2570  2     ! Note: The conversion routine is not entered here.
;  2528     2571  2     ! If we need converted keys, we will branch back here to convert them
;  2529     2572  2     !
;  2530     2573  2     !-
;  2531     2574  2
;  2532     2575  2
;  2533     2576  2     ! Save offset from beginning of the code of the current address
;  2534     2577  2     !
;  2535     2578  2     TMP = .CUR_PC - .CTX[S_START];
;  2536     2579  2
;  2537     2580  2
;  2538     2581  2     ! No stack space needed yet
;  2539     2582  2     !
;  2540     2583  2     STACK = 0;
;  2541     2584  2
;  2542     2585  2
;  2543     2586  2     ! Because we are using GEN_MOVE (et al), we must assert that the input
;  2544     2587  2     ! conversion routine does not preserve registers R0..R5 or R6 (the length).
;  2545     2588  2     !
;  2546     2589  2     ASSERT_((%B'1111111' AND NOT %NOPRESERVE(JSB_INPUT)) EQL 0)
;  2547     2590  2
;  2548     2591  2
;  2549     2592  2     ! Check whether the user is doing his own comparisons
;  2550     2593  2     !
;  2551     2594  2     IF .CTX[COM_COMPARE] EQL 0
;  2552     2595  2     THEN
;  2553     2596  3         BEGIN
;  2554     2597  3         !
;  2555     2598  3         ! Loop through, and convert keys as needed
;  2556     2599  3         !
;  2557     2600  3         ! Note that we may expand COM_ORD_KEY.  This is okay, as we've not yet
;  2558     2601  3         ! generated code to reference any of the other COM_ORD_xxx fields.
;  2559     2602  3         !
;  2560     2603  3         LOCAL
;  2561     2604  3             OKECNT,             ! Byte count of original keys
;  2562     2605  3             CVTCNT;             ! Byte count of converted keys
;  2563     2606  3
;  2564     2607  3         OKECNT = 0;
;  2565     2608  3         CVTCNT = 0;
;  2566     2609  3         INCR I FROM 0 TO .KEY_BUFF[KEY_NUMBER]-1 DO
;  2567     2610  4             BEGIN
;  2568     2611  4             LOCAL
;  2569     2612  4                 KBF:    REF KBF_BLOCK;  ! Local copy of key
;  2570     2613  4
;  2571     2614  4             KBF = KEY_BUFF[KEY_KBF(.I)];        ! Pointer to key description
;  2572     2615  4
;  2573     2616  4             ! For index sorts, copy the original keys.
;  2574     2617  4             ! Note that the OKEY area has already been allocated.
;  2575     2618  4             ! Since the KEY area may be expanded later for converted keys,
```

```
;  2576          2619  4          ! we require that the OKEY area not be moved (OKEY LSS KEY).
;  2577          2620  4          !
;  2578          2621  4          IF .CTX[COM_SORT_TYPE] EQL TYP_K_INDEX
;  2579          2622  4          THEN
;  2580          2623  5              BEGIN
;  2581          2624  5              ASSERT_(COM_ORD_OKEY LSS COM_ORD_KEY)
;  2582          2625  5              LOCAL
;  2583          2626  5                  KEYLEN;                     ! Length in bytes of the key
;  2584          2627  5              KEYLEN = LEN_(KBF[BASE_]);
;  2585          2628  5              ROOM(K_MOVE);
;  2586          2629  5              GEN_MOVE_VAR(.KEYLEN,
;  2587          2630  5                  .KBF[KBF_POSITION], COM_REG_SRC1,
;  2588          2631  5                  .DISP[COM_ORD_OKEY]+.OKECNT, COM_REG_SRC2);
;  2589          2632  5
;  2590          2633  5              ! Indicate that the key location is relative to the
;  2591          2634  5              ! beginning of the internal format record.
;  2592          2635  5              !
;  2593          2636  5              KBF[KBF_POSITION] = .DISP[COM_ORD_OKEY]+.OKECNT;
;  2594          2637  5              KBF[KBF_CVT] = TRUE;
;  2595          2638  5              OKECNT = .OKECNT + .KEYLEN;
;  2596          2639  4              END;
;  2597          2640  4
;  2598          2641  4          %IF NOT HOSTILE %THEN
;  2599          2642  4          !
;  2600          2643  4          ! If a user-defined key, convert it as needed.
;  2601          2644  4          !
;  2602          2645  4          IF .KBF[KBF_TYPE] GTRU MAX_SUPPORTED
;  2603          2646  4          THEN
;  2604          2647  5              BEGIN
;  2605          2648  5              EXPAND_('KEY', 0);          ! Make sure the key area is allocated
;  2606          2649  5              CVTCNT = .CVTCNT +          ! Generate code and add to length
;  2607          2650  5                  GEN_CONVERT_UDEF(KBF[BASE_], .DISP[COM_ORD_KEY]+.CVTCNT);
;  2608          2651  4              END;
;  2609          2652  4          %FI
;  2610          2653  4
;  2611          2654  4          ! If a decimal datatype (other than packed),
;  2612          2655  4          ! or G,H_floating with no hardware support, then convert the key.
;  2613          2656  4          !
;  2614        P 2657  4          IF ONEOF_(.KBF[KBF_TYPE], BMSK_(
;  2615        P 2658  4              DSC$K_DTYPE_NU, DSC$K_DTYPE_NZ, DSC$K_DTYPE_NL,
;  2616          2659  5              DSC$K_DTYPE_NLO, DSC$K_DTYPE_NR, DSC$K_DTYPE_NRO))
;  2617          2660  4          THEN
;  2618          2661  5              BEGIN
;  2619          2662  5              EXPAND_('KEY', 0);          ! Make sure the key area is allocated
;  2620          2663  5              CVTCNT = .CVTCNT +          ! Generate code and add to length
;  2621          2664  5                  GEN_CONVERT_DEC(KBF[BASE_], .DISP[COM_ORD_KEY]+.CVTCNT,
;  2622          2665  5                      STACK);
;  2623          2666  5              END
;  2624          2667  4          ELIF
;  2625        P 2668  4              ONEOF_(.KBF[KBF_TYPE], BMSK_(DSC$K_DTYPE_F, DSC$K_DTYPE_D,
;  2626          2669  4                  DSC$K_DTYPE_G, DSC$K_DTYPE_H)) AND
;  2627          2670  4                  NOT FDGH_HARDWARE_(.KBF[KBF_TYPE])
;  2628          2671  4          THEN
;  2629          2672  5              BEGIN
;  2630          2673  5              EXPAND_('KEY', 0);          ! Make sure the key area is allocated
;  2631          2674  5              CVTCNT = .CVTCNT +          ! Generate code and add to length
;  2632          2675  5                  GEN_CONVERT_FLT(KBF[BASE_], .DISP[COM_ORD_KEY]+.CVTCNT);
```

```
;  2633          2676  4                END;
;  2634          2677  3            END;
;  2635          2678  3
;  2636          2679  3
;  2637          2680  3        ! Save the other keys, unless we have them in the DATA area.
;  2638          2681  3        !
;  2639          2682  3        IF .DISP[COM_ORD_DATA] LSS 0            ! If we aren't saving data,
;  2640          2683  3        THEN                                    ! then we'd better save keys
;  2641          2684  4            BEGIN
;  2642          2685  4            EXPAND_('KEY', 0);          ! Make sure the key area is allocated
;  2643          2686  4            CVTCNT = .CVTCNT + MOVE_KEYS(
;  2644          2687  4                KEY_BUFF[BASE_],        ! Key descriptions
;  2645          2688  4                .DISP[COM_ORD_KEY]+.CVTCNT);    ! Displacement
;  2646          2689  3            END;
;  2647          2690  3
;  2648          2691  3
;  2649          2692  3        ! Round up stack requirements
;  2650          2693  3        !
;  2651          2694  3        STACK = ROUND_(.STACK);
;  2652          2695  3
;  2653          2696  3
;  2654          2697  3        ! If any keys were converted, actually allocate the space, and
;  2655          2698  3        ! issue a return, since we'll be coming back this way.
;  2656          2699  3        !
;  2657          2700  3        IF .DISP[COM_ORD_KEY] GEQ 0 OR .DISP[COM_ORD_OKEY] GEQ 0
;  2658          2701  3        THEN
;  2659          2702  4            BEGIN
;  2660          2703  4            EXPAND_('KEY', .CVTCNT);
;  2661          2704  4            ROOM(1+K_LITE+1+4);
;  2662          2705  4            IF .STACK NEQ 0
;  2663          2706  4            THEN
;  2664          2707  5                BEGIN
;  2665          2708  5                EMIT_BYTE(OPC_ADDL2);
;  2666          2709  5                EMIT_LITE(K_LONG, .STACK);
;  2667          2710  5                EMIT_BYTE(M_R+R_SP);
;  2668          2711  4                END;
;  2669          2712  4            EMIT_BYTES(OPC_MOVL, 1, M_R+R_0, OPC_RSB);  ! Return success
;  2670          2713  3            END;
;  2671          2714  2        END;
;  2672          2715  2
;  2673          2716  2
;  2674          2717  2        ! This is where we want to enter the conversion routine.
;  2675          2718  2        ! For now, just store the offset from the beginning of the string.
;  2676          2719  2        !
;  2677          2720  2        CTX[COM_INPUT] = .CUR_PC - .CTX[S_START];
;  2678          2721  2
;  2679          2722  2
;  2680          2723  2        ! If there is a record definition table, call SOR$$RDT to determine whether
;  2681          2724  2        ! to omit or include this record.
;  2682          2725  2        !
;  2683          2726  2        %IF NOT HOSTILE %THEN
;  2684          2727  2        IF .CTX[COM_RDT_ADR] NEQ 0
;  2685          2728  2        THEN
;  2686          2729  3            BEGIN
;  2687          2730  3            ROOM(7+K_ABSA+7+4+K_LITE+1+K_DISP);
;  2688        P 2731  3            EMIT_BYTES(
;  2689        P 2732  3                OPC_PUSHAL, M_AD+R_SP,              ! PUSHAL -(SP)
```

```
;  2690        P 2733  3              OPC_PUSHAB, M_RD+COM_REG_SRC1,       ! PUSHA (Rsrc1)
;  2691          2734  3              OPC_CALLS, 2);
;  2692          2735  3          EMIT_ABSA(SOR$$RDT);                     ! CALLS #2, SOR$RDT
;  2693        P 2736  3          EMIT_BYTES(
;  2694        P 2737  3              OPC_MOVL, M_AI+R_SP, M_R+R_1,        ! MOVL  (SP)+, R1
;  2695        P 2738  3              OPC_BLBS, M_R+R_0, 1,                ! BLBS  R0, 1$
;  2696          2739  3              OPC_RSB);                            ! RSB
;  2697        C 2740  3 %(
;  2698        C 2741  3          IF .DISP[COM_ORD_FORM] GEQ 0
;  2699        C 2742  3          THEN
;  2700        C 2743  3              BEGIN
;  2701        C 2744  3              ASSERT (RDT_UNIT LEQ SHORT_LIT)
;  2702        C 2745  3              EMIT_BYTES(OPC_DIVL2, RDT_UNIT, M_R+R_1, OPC_SUBB3);
;  2703        C 2746  3              EMIT_LITE(K_BYTE, .CTX[COM_RDT_ADR]/RDT_UNIT);
;  2704        C 2747  3              EMIT_BYTE(M_R+R_1);
;  2705        C 2748  3              EMIT_DISP(.DISP[COM_ORD_FORM], COM_REG_SRC2);
;  2706        C 2749  3              END;
;  2707          2750  3 )%
;  2708          2751  2          END;
;  2709          2752  2      %FI
;  2710          2753  2
;  2711          2754  2
;  2712          2755  2      ! Store the length, if needed
;  2713          2756  2      !
;  2714          2757  2      IF .DISP[COM_ORD_VAR] GEQ 0
;  2715          2758  2      THEN
;  2716          2759  3          BEGIN
;  2717          2760  3          ROOM(1+K_LITE+1+K_DISP);
;  2718          2761  3          IF .CTX[COM_TKS] NEQ 0
;  2719          2762  3          THEN
;  2720          2763  4              BEGIN
;  2721          2764  4              EMIT_BYTE(OPC_SUBW3);                ! SUBW3
;  2722          2765  4              EMIT_LITE(K_WORD, .CTX[COM_TKS]);    !       #tks
;  2723          2766  4              END
;  2724          2767  3          ELSE
;  2725          2768  4              BEGIN
;  2726          2769  4              EMIT_BYTE(OPC_MOVW);                 ! MOVW
;  2727          2770  3              END;
;  2728          2771  3          EMIT_BYTE(M_RD+COM_REG_SRC1);            !       (Rsrc1)
;  2729          2772  3          EMIT_DISP(.DISP[COM_ORD_VAR], COM_REG_SRC2);    !       n(Rsrc2)
;  2730          2773  2          END;
;  2731          2774  2
;  2732          2775  2
;  2733          2776  2      ! Store the original data, if needed.
;  2734          2777  2      !
;  2735          2778  2      IF .DISP[COM_ORD_DATA] GEQ 0
;  2736          2779  2      THEN
;  2737          2780  3          BEGIN
;  2738          2781  3          ROOM(MAX(4+K_MOVE,4+K_LITE+K_LITE+K_DISP));
;  2739          2782  3          IF .DISP[COM_ORD_VAR] LSS 0
;  2740          2783  3          THEN
;  2741          2784  4              BEGIN
;  2742          2785  4              !
;  2743          2786  4              ! Special-case fixed-length records
;  2744          2787  4              !
;  2745          2788  4              EMIT_BYTES(OPC_MOVL, M_BD+COM_REG_SRC1, 4, M_R+R_1);
;  2746          2789  4              GEN_MOVE(.CTX[COM_LRL],              !       #length
```

```
;  2747          2790  ;                0, 1,                                   !       0(R1)
;  2748          2791  4                .DISP[COM_ORD_DATA], COM_REG_SRC2);     !       src2disp(Rsrc2)
;  2749          2792  4            END
;  2750          2793  3        ELSE
;  2751          2794  4            BEGIN
;  2752        P 2795  4            EMIT_BYTES(OPC_MOVC5,                       ! MOVC5
;  2753        P 2796  4                M_RD+COM_REG_SRC1,                      !       (Rsrc1),
;  2754          2797  4                M_BDD+COM_REG_SRC1,4);                  !       @4(Rsrc1),
;  2755          2798  4            EMIT_LITE(K_BYTE, .CTX[COM_PAD]);           !       #pad
;  2756          2799  4            EMIT_LITE(K_WORD, .CTX[COM_LRL]);           !       #length
;  2757          2800  4            EMIT_DISP(
;  2758          2801  4                .DISP[COM_ORD_DATA], COM_REG_SRC2);     !       src2disp(Rsrc2)
;  2759          2802  3            END;
;  2760          2803  2        END;
;  2761          2804  2
;  2762          2805  2    ! Store the record number for stable sorts.
;  2763          2806  2    ! Store the stream number for stable merges.
;  2764          2807  2    !
;  2765          2808  2    IF .DISP[COM_ORD_STAB] GEQ 0
;  2766          2809  2    THEN
;  2767          2810  3        BEGIN
;  2768          2811  3        ROOM(1+K_DISP+K_DISP);
;  2769          2812  3        EMIT_BYTE(OPC_MOVL);                            ! MOVL
;  2770          2813  3        EMIT_DISP(                                      !       m(CTX)
;  2771          2814  4            (IF .CTX[COM_MERGE]
;  2772          2815  4            THEN %FIELDEXPAND(COM_MRG_STREAM,0)*%UPVAL
;  2773          2816  3            ELSE %FIELDEXPAND(COM_INP_RECNUM,0)*%UPVAL), COM_REG_CTX);
;  2774          2817  3        EMIT_DISP(.DISP[COM_ORD_STAB], COM_REG_SRC2);   !       n(Rsrc2)
;  2775          2818  2        END;
;  2776          2819  2
;  2777          2820  2
;  2778          2821  2    ! If we need the RFA, copy it too
;  2779          2822  2    !
;  2780          2823  2    IF .DISP[COM_ORD_RFA] GEQ 0
;  2781          2824  2    THEN
;  2782          2825  3        BEGIN
;  2783          2826  3        MACRO O_(O,P,S,E) = O %;
;  2784          2827  3        ASSERT (RAB$S_RFA EQL 6)
;  2785          2828  3        ROOM(1+K_DISP+K_LITE+3+K_DISP+2+K_DISP);
;  2786          2829  3        EMIT_BYTE(OPC_ADDL3);                           ! ADDL3
;  2787          2830  3        EMIT_DISP(                                      !       mm(CTX)
;  2788          2831  3            %FIELDEXPAND(COM_INP_CURR,0)*%UPVAL,COM_REG_CTX);
;  2789          2832  3        EMIT_LITE(K_LONG, DDB_RAB+O_(RAB$W_RFA));       !       #offset
;  2790        P 2833  3        EMIT_BYTES(M_R+R_0,                             !       R0
;  2791          2834  3            OPC_MOVL, M_AI+R_0);                        ! MOVL  (R0)+
;  2792          2835  3        EMIT_DISP(.DISP[COM_ORD_RFA],  COM_REG_SRC2);   !       nn(Rsrc2)
;  2793          2836  3        EMIT_BYTES(OPC_MOVW, M_RD+R_0);                 ! MOVW  (R0)
;  2794          2837  3        EMIT_DISP(.DISP[COM_ORD_RFA]+4,COM_REG_SRC2);   !       nn+4(Rsrc2)
;  2795          2838  3        !
;  2796          2839  3        ! If the file number is needed, get it, too
;  2797          2840  3        !
;  2798          2841  3        IF .DISP[COM_ORD_FILE] GEQ 0
;  2799          2842  3        THEN
;  2800          2843  4            BEGIN
;  2801          2844  4            ROOM(1+K_DISP+K_DISP);
;  2802          2845  4            EMIT_BYTE(OPC_MOVB);
;  2803          2846  4            EMIT_DISP(%FIELDEXPAND(DDB_FIL,0)
```

```
;  2804          2847  4                  -DDB_RAB
;  2805          2848  4                  -0_(RAB$W_RFA)
;  2806          2849  4                  -4, R_0);
;  2807          2850  4              EMIT_DISPT.DISP[COM_ORD_FILE], COM_REG_SRC2);
;  2808          2851  3              END;
;  2809          2852  2          END;
;  2810          2853  2
;  2811          2854  2
;  2812          2855  2      ! If the VFC area (record header buffer, RHB) is needed, get it, too.
;  2813          2856  2      ! Note that the VFC area, like the RFA, is passed through the context area.
;  2814          2857  2      ! On the chance that the code later decides to not allocate the VFC area,
;  2815          2858  2      ! check whether the address of the storage is zero.
;  2816          2859  2      !
;  2817          2860  2      IF .DISP[COM_ORD_VFC] GEQ 0
;  2818          2861  2      THEN
;  2819          2862  3          BEGIN
;  2820          2863  3          LOCAL
;  2821          2864  3              TMP2: REF VECTOR[,BYTE];            ! Temporary pointer to code
;  2822          2865  3          ROOM(1+K_DISP+3+K_MOVE);
;  2823          2866  3          EMIT_BYTE(OPC_MOVL);                                ! MOVL
;  2824          2867  3          EMIT_DISP(%FIELDEXPAND(COM_RHB_INP,0)*%UPVAL,       !       mm(CTX)
;  2825          2868  3              COM_REG_CTX);
;  2826        P 2869  3          EMIT_BYTES(M_R+R_0,                                 !       R0
;  2827          2870  3              OPC_BEQL, 0);                                   ! BEQL  0$
;  2828          2871  3          TMP2 = .CUR_PC;                                     ! Save PC
;  2829          2872  3          GEN_MOVE(.CTX[COM_MINVFC], 0, R_0,                  ! MOVE  #len, 0(R0)
;  2830          2873  3              .DISP[COM_ORD_VFC], COM_REG_SRC2);              !       nn(Rsrc2)
;  2831          2874  3          TMP2[-1] = .COR_PC - .TMP2;             ! Correct displacement
;  2832          2875  2          END;
;  2833          2876  2
;  2834          2877  2
;  2835          2878  2      ! If we need to convert keys, branch back and convert them
;  2836          2879  2      !
;  2837          2880  2      ROOM(1+K_LITE+1+10);
;  2838          2881  2      TMP = TMP[0] + .CTX[S_START];       ! Adjust TMP to be actual address
;  2839          2882  3      IF (.DISP[COM_ORD_KEY] GEQ 0 OR .DISP[COM_ORD_OKEY] GEQ 0)
;  2840          2883  3          AND .(TMP[0]) NEQ (OPC_MOVL + 1^8 + (M_R+R_0)^16 + OPC_RSB^24)
;  2841          2884  2      THEN
;  2842          2885  3          BEGIN
;  2843          2886  3          LOCAL
;  2844          2887  3              Z;
;  2845          2888  3          IF .STACK NEQ 0
;  2846          2889  3          THEN
;  2847          2890  4              BEGIN                                   ! Allocate stack space
;  2848          2891  4              EMIT_BYTE(OPC_SUBL2);                           ! SUBL2
;  2849          2892  4              EMIT_LITE(K_LONG, .STACK);                      !       #stack
;  2850          2893  4              EMIT_BYTE(M_R+R_SP);                            !       SP
;  2851          2894  3              END;
;  2852        P 2895  3          EMIT_BYTES(
;  2853        P 2896  3              OPC_MOVW, M_RD+COM_REG_SRC1, M_R+R_6,       ! MOVW  (Rsrc1), R6
;  2854        P 2897  3              OPC_MOVL,                                   ! MOVL
;  2855        P 2898  3              M_BD+COM_REG_SRC1, 4,                       !       4(Rsrc1),
;  2856          2899  3              M_R+COM_REG_SRC1);                          !       Rsrc1
;  2857          2900  3          Z = TMP[0] = .COR_PC - 3;                       ! Branch displacement
;  2858          2901  3          IF .Z<0,8,1> EQL .Z                             ! Will BRB suffice?
;  2859          2902  4          THEN (EMIT_BYTE(OPC_BRB); EMIT_BYTE(.Z+1))
;  2860          2903  3          ELSE (EMIT_BYTE(OPC_BRW); EMIT_WORD(.Z));
```

```
;  2861          2904  3          END
;  2862          2905  2      ELSE
;  2863        P 2906  2          EMIT_BYTES(OPC_MOVL, 1, M_R+R_0,        ! MOVL  #1, R0
;  2864          2907  2              OPC_RSB);                           ! RSB
;  2865          2908  2
;  2866          2909  2      VERIFY_LEN();                               ! Check for writing too far
```

```
;  2868          2910  2 !+
;  2869          2911  2 !
;  2870          2912  2 ! Generate the key comparison routine
;  2871          2913  2 !
;  2872          2914  2 !-
;  2873          2915  2
;  2874          2916  2     ! Store the offset from the beginning of the string.
;  2875          2917  2     !
;  2876          2918  2     TMP = .CUR_PC - .CTX[S_START];
;  2877          2919  2
;  2878          2920  2
;  2879          2921  2     ! Initialize the branches table.
;  2880          2922  2     !
;  2881          2923  2     CH$FILL(%X'FF', %ALLOCATION(BRANCHES), BRANCHES[0]);
;  2882          2924  2     BRANCH = BRANCHES[0];
;  2883          2925  2
;  2884          2926  2
;  2885          2927  2     ! We haven't saved any registers yet.
;  2886          2928  2     !
;  2887          2929  2     SAVED_REGS = 0;
;  2888          2930  2
;  2889          2931  2
;  2890          2932  2     ! If the user supplied a comparison routine, call it.
;  2891          2933  2     !
;  2892          2934  2     IF .CTX[COM_COMPARE] NEQ 0
;  2893          2935  2     THEN
;  2894          2936  3         BEGIN
;  2895          2937  3         ROOM(K_CALL4+4);
;  2896          2938  3         EMIT_CALL4(.CTX[COM_COMPARE], DISP[0]);
;  2897          2939  3         IF .DISP[COM_ORD_STAB] GEQ 0                    ! Stable?
;  2898          2940  3         THEN
;  2899        P 2941  3             EMIT_BYTES(OPC_BLBC, M_R+R_0, 1,        ! BLBC  R0, 1$
;  2900          2942  4                        OPC_RSB)                     ! RSB
;  2901          2943  3         ELSE
;  2902          2944  3             EMIT_BYTE(OPC_RSB);                     ! RSB
;  2903          2945  3         END
;  2904          2946  2     ELSE
;  2905          2947  3         BEGIN
;  2906          2948  3         INCR I FROM 0 TO .KEY_BUFF[KEY_NUMBER]-1 DO
;  2907          2949  4             BEGIN
;  2908          2950  4             LOCAL
;  2909          2951  4                 KBF:    REF KBF_BLOCK;
;  2910          2952  4             BUILTIN
;  2911          2953  4                 TESTBITCC;
;  2912          2954  4             KBF = KEY_BUFF[KEY_KBF(.I)];
;  2913          2955  4             IF TESTBITCC(KBF[KBF_CVT])
;  2914          2956  4             THEN
;  2915          2957  5                 BEGIN
;  2916          2958  5                 !
;  2917          2959  5                 ! The key has not been converted.
;  2918          2960  5                 ! Adjust the offset to the key.
;  2919          2961  5                 !
;  2920          2962  5                 KBF[KBF_POSITION] = .KBF[KBF_POSITION] + .DISP[COM_ORD_DATA];
;  2921          2963  4                 END;
;  2922          2964  4             GEN_COMPARE(KBF[BASE_], .I);
;  2923          2965  3             END;
;  2924          2966  2         END;
```

```
;  2925          2967  2
;  2926          2968  2
;  2927          2969  2      ! Generate a little more code for stable sorts
;  2928          2970  2      !
;  2929          2971  2      IF .DISP[COM_ORD_STAB] GEQ 0
;  2930          2972  2      THEN
;  2931          2973  3          BEGIN
;  2932          2974  3          !
;  2933          2975  3          ! Generate another comparison for stable sorts
;  2934          2976  3          !
;  2935          2977  3          LOCAL
;  2936          2978  3              KBF: KBF_BLOCK;
;  2937          2979  3          KBF[KBF_TYPE] = DSC$K_DTYPE_BU;
;  2938          2980  3          KBF[KBF_ORDER] = 0;
;  2939          2981  3          KBF[KBF_POSITION] = .DISP[COM_ORD_STAB];
;  2940          2982  3          KBF[KBF_LENGTH] = %UPVAL;
;  2941          2983  3          GEN_COMPARE(KBF[BASE_]);
;  2942          2984  2          END;
;  2943          2985  2
;  2944          2986  2
;  2945          2987  2      ! Store the length of an internal-format record
;  2946          2988  2      !
;  2947          2989  2      CTX[COM_LRL_INT] = .DISP[COM_ORD_MAX];
;  2948          2990  2      IF .DISP[COM_ORD_MAX] GTR MAX_REFSIZE
;  2949          2991  2      THEN
;  2950          2992  2          SOR$$ERROR(SOR$_SHR_BADLOGIC);  ! Not really bad logic, just rare.
;  2951          2993  2
;  2952          2994  2
;  2953          2995  2      ! Generate code to return a zero, and process saved registers
;  2954          2996  2      !
;  2955          2997  2      ROOM(5);
;  2956          2998  2      EMIT_BYTES(OPC_CLRL, M_R+R_0);                  ! CLRL   R0
;  2957          2999  2      IF .SAVED_REGS NEQ 0 THEN EMIT_BYTES(OPC_POPR, .SAVED_REGS);
;  2958          3000  2      EMIT_BYTE(OPC_RSB);                             ! RSB
;  2959          3001  2
;  2960          3002  2
;  2961          3003  2      ! Store the offset to the start of this routine
;  2962          3004  2      !
;  2963          3005  2      CTX[COM_COMPARE] = .TMP;
;  2964          3006  2
;  2965          3007  2
;  2966          3008  2      ! Check for writing too far
;  2967          3009  2      !
;  2968          3010  2      VERIFY_LEN();
```

```
;  2970          3011  2 !+
;  2971          3012  2 !
;  2972          3013  2 ! Generate the equal-key routine
;  2973          3014  2 !
;  2974          3015  2 !-
;  2975          3016  2     ! Store the offset from the beginning of the string.
;  2976          3017  2     !
;  2977          3018  2     TMP = .CUR_PC - .CTX[S_START];
;  2978          3019  2
;  2979          3020  2
;  2980          3021  2     ! If the user specified his own equal-key routine, call it.
;  2981          3022  2     !
;  2982          3023  2     IF .CTX[COM_EQUAL] NEQ 0
;  2983          3024  2     THEN
;  2984          3025  3         BEGIN
;  2985          3026  3         ROOM(K_CALL4+9+K_LITE+2+K_ABSA+4);
;  2986          3027  3         EMIT_CALL4(.CTX[COM_EQUAL], DISP[0]);
;  2987        P 3028  3         EMIT_BYTES(OPC_BLBC, M_R+R_0, 1,                    ! BLBC  R0, error
;  2988        P 3029  3                 OPC_RSB,                                  ! RSB
;  2989        P 3030  3                 OPC_PUSHL, M_R+R_0,                       ! error: PUSHL R0
;  2990        P 3031  3                 OPC_PUSHL, 0,                             ! PUSHL #0
;  2991          3032  3                 OPC_PUSHL);                               ! PUSHL #SOR$_RTNERROR
;  2992          3033  3         EMIT_LITE(K_LONG, SOR$_RTNERROR);
;  2993          3034  3         EMIT_BYTES(OPC_CALLS, 3);                         ! CALLS #3
;  2994          3035  3         EMIT_ABSA(SOR$$ERROR);                            !       SOR$$ERROR
;  2995        P 3036  3         EMIT_BYTES(OPC_MOVL, SS$_NORMAL, M_R+R_0,         ! MOVL  #SS$_NORMAL, R0
;  2996          3037  3                 OPC_RSB);                                 ! RSB
;  2997          3038  3         END
;  2998          3039  2     ELIF .CTX[COM_NODUPS]
;  2999          3040  2     THEN
;  3000          3041  3         BEGIN
;  3001          3042  3         EMIT_BYTES(OPC_MOVL, M_AI+R_PC);                  ! MOVL
;  3002          3043  3         EMIT_LONG(SOR$_DELETE2);                          !       #SOR$_DELETE2
;  3003        P 3044  3         EMIT_BYTES(M_R+R_0,                               !       R0
;  3004          3045  3             OPC_RSB);                                     ! RSB
;  3005          3046  3         END
;  3006          3047  2     ELSE
;  3007          3048  3         BEGIN
;  3008          3049  3         !
;  3009          3050  3         ! Emit a HALT instruction.
;  3010          3051  3         ! This indicates that COM_EQUAL should be set to zero below.
;  3011          3052  3         !
;  3012          3053  3         EMIT_BYTE(OPC_HALT);
;  3013          3054  2         END;
;  3014          3055  2
;  3015          3056  2
;  3016          3057  2     ! Store the offset to the start of this routine
;  3017          3058  2     !
;  3018          3059  2     CTX[COM_EQUAL] = .TMP;
;  3019          3060  2
;  3020          3061  2
;  3021          3062  2     ! Check for writing too far
;  3022          3063  2     !
;  3023          3064  2     VERIFY_LEN();
```

```
;  3025          3065  2 !+
;  3026          3066  2 !
;  3027          3067  2 ! Generate the length-address routine
;  3028          3068  2 !
;  3029          3069  2 !-
;  3030          3070  2     ! Store the offset to this routine
;  3031          3071  2     !
;  3032          3072  2     CTX[COM_LENADR] = .CUR_PC - .CTX[S_START];
;  3033          3073  2
;  3034          3074  2
;  3035          3075  2     ! If the VFC area (record header buffer, RHB) is needed, get it.
;  3036          3076  2     ! Note that the VFC area is passed through the context area.
;  3037          3077  2     !
;  3038          3078  2     IF .DISP[COM_ORD_VFC] GEQ 0
;  3039          3079  2     THEN
;  3040          3080  3         BEGIN
;  3041          3081  3         LOCAL
;  3042          3082  3             TMP2: REF VECTOR[,BYTE];            ! Temporary pointer to code
;  3043          3083  3         ROOM(1+K_DISP+1+2+8+K_MOVE+1);
;  3044          3084  3         EMIT_BYTE(OPC_MOVL);                                ! MOVL
;  3045          3085  3         EMIT_DISP(%FIELDEXPAND(COM_RHB_OUT,0)*%UPVAL,       !       mm(CTX)
;  3046          3086  3             COM_REG_CTX);
;  3047          3087  3         EMIT_BYTE(M_R+R_0);                                 !       R0
;  3048          3088  3         EMIT_BYTES(OPC_BEQL, 0);                            ! BEQL  4$
;  3049          3089  3         TMP2 = .CUR_PC;
;  3050          3090  3
;  3051          3091  3         ! If any of R0..R5 are %NOTUSED, call a routine to do the move.
;  3052          3092  3         ! If any of R0..R5 are %PRESERVE, save and restore the registers.
;  3053          3093  3         !
;  3054          3094  3         IF .CTX[COM_MINVFC] GTRU TUN_K_BINMOVE
;  3055          3095  3         THEN
;  3056        L 3096  3             %IF (%B'111111' AND %NOTUSED(JSB_LENADR)) NEQ 0
;  3057          3097  3             %THEN
;  3058          3098  4                 BEGIN
;  3059        P 3099  4                 EMIT_BYTES(OPC_CALLS, 0, M_BD+R_PC, 2,  ! CALLS #0, 2(PC)
;  3060        P 3100  4                     OPC_BRB, 0,                         ! BRB   4$
;  3061        P 3101  4                     %B'111111'                          ! .WORD ^M<mask>
;  3062          3102  4                     AND (%NOTUSED(JSB_LENADR) OR %PRESERVE(JSB_LENADR)), 0);
;  3063          3103  4                 TMP = .CUR_PC - 2;
;  3064          3104  4                 END
;  3065        U 3105  4             %ELSE %IF (%B'111111' AND %PRESERVE(JSB_LENADR)) NEQ 0
;  3066        U 3106  4             %THEN
;  3067        U 3107  4                 EMIT_BYTES(OPC_PUSHR, %B'111111' AND %PRESERVE(JSB_LENADR))
;  3068        U 3108  4             %ELSE
;  3069        U 3109  4                 0
;  3070          3110  3             %FI %FI;
;  3071          3111  3
;  3072          3112  3         GEN_MOVE(.CTX[COM_MINVFC],                      ! MOVE  #len,
;  3073          3113  3             .DISP[COM_ORD_VFC],  COM_REG_SRC2,          !       nn(Rsrc2)
;  3074          3114  3             0, R_0);                                    !       0(R0)
;  3075          3115  3
;  3076          3116  3         IF .CTX[COM_MINVFC] GTRU TUN_K_BINMOVE
;  3077          3117  3         THEN
;  3078        L 3118  3             %IF (%B'111111' AND %NOTUSED(JSB_LENADR)) NEQ 0
;  3079          3119  3             %THEN
;  3080          3120  4                 BEGIN
;  3081          3121  4                 EMIT_BYTE(OPC_RET);                             ! RET
```

```
;  3082          3122  4                TMP[-1] = .CUR_PC - .TMP;                          !4$:
;  3083          3123  4                END
;  3084        U 3124  4            %ELSE %IF (%B'111111' AND %PRESERVE(JSB_LENADR)) NEQ 0
;  3085        U 3125  4            %THEN
;  3086        U 3126  4                EMIT_BYTES(OPC_POPR, %B'111111' AND %PRESERVE(JSB_LENADR))
;  3087        U 3127  4            %ELSE
;  3088        U 3128  4                0
;  3089          3129  3            %FI %FI;
;  3090          3130  3
;  3091          3131  3        TMP2[-1] = .CUR_PC - .TMP2;
;  3092          3132  3
;  3093          3133  2        END;
;  3094          3134  2
;  3095          3135  2
;  3096          3136  2        ! Generate code to move the length/address into R0/R1
;  3097          3137  2        !
;  3098          3138  2        ! Set the longest output record length.
;  3099          3139  2        !
;  3100          3140  2        ROOM(MAX(                                           ! Make room for the code
;  3101          3141  2            1+MAX(K_DISP,K_LITE)+2+K_DISP+2,                ! RECORD
;  3102          3142  2            1+K_DISP+2+K_ABSA+1,                            ! TAG
;  3103          3143  2            4+K_DISP+2,                                     ! ADDRESS
;  3104          3144  2            1+K_LITE+2+K_DISP+2));                          ! INDEX
;  3105          3145  2        CASE .CTX[COM_SORT_TYPE] FROM TYP_K_RECORD TO TYP_K_MAX OF
;  3106          3146  2            SET
;  3107          3147  2
;  3108          3148  2            [TYP_K_RECORD]:
;  3109          3149  3                BEGIN
;  3110          3150  3                CTX[COM_LRL_OUT] = .CTX[COM_LRL]-.CTX[COM_TKS];
;  3111          3151  3                EMIT_BYTE(OPC_MOVZWL);
;  3112          3152  3                IF .DISP[COM_ORD_VAR] GEQ 0
;  3113          3153  3                THEN
;  3114          3154  3                    EMIT_DISP(.DISP[COM_ORD_VAR], COM_REG_SRC2)
;  3115          3155  3                ELSE
;  3116          3156  3                    EMIT_LITE(K_WORD, .CTX[COM_LRL_OUT]);
;  3117          3157  3                EMIT_BYTES(M_R+R_0, OPC_MOVAB);
;  3118          3158  3                EMIT_DISP(.DISP[COM_ORD_DATA]+.CTX[COM_TKS], COM_REG_SRC2);
;  3119          3159  3                EMIT_BYTES(M_R+R_1, OPC_RSB);
;  3120          3160  2                END;
;  3121          3161  2
;  3122          3162  2            %IF NOT HOSTILE %THEN
;  3123          3163  2
;  3124          3164  2            [TYP_K_TAG]:
;  3125          3165  3                BEGIN
;  3126          3166  3                CTX[COM_LRL_OUT] = .CTX[COM_LRL];
;  3127          3167  3                EMIT_BYTE(OPC_PUSHAB);                      ! PUSHAB
;  3128          3168  3                EMIT_DISP(.DISP[COM_ORD_RFA], COM_REG_SRC2); !       rfa(Rsrc1)
;  3129          3169  3                EMIT_BYTES(OPC_CALLS, 1);                   ! CALLS #1
;  3130          3170  3                EMIT_ABSA(SOR$$RFA_ACCESS);                 !       SOR$$RFA_ACCESS
;  3131          3171  3                EMIT_BYTE(OPC_RSB);                         ! RSB
;  3132          3172  2                END;
;  3133          3173  2
;  3134          3174  2            [TYP_K_ADDRESS]:
;  3135          3175  3                BEGIN
;  3136          3176  3                CTX[COM_LRL_OUT] = RAB$S_RFA;
;  3137          3177  3                IF .DISP[COM_ORD_FILE] GEQ 0
;  3138          3178  3                THEN
```

```
;  3139          3179  3                CTX[COM_LRL_OUT] = .CTX[COM_LRL_OUT] + 1;
;  3140          3180  3            ASSERT (COM_ORD_RFA+1 EQL COM_ORD_FILE)
;  3141          3181  3            EMIT_BYTE(OPC_MOVL);
;  3142          3182  3            ASSERT (RAB$S_RFA+1 LEQ SHORT_LIT)
;  3143          3183  3            EMIT_BYTE(.CTX[COM_LRL_OUT]);
;  3144        P 3184  3            EMIT_BYTES(M_R+R_0,
;  3145          3185  3                    OPC_MOVAB);
;  3146          3186  3            EMIT_DISP(.DISP[COM_ORD_RFA], COM_REG_SRC2);
;  3147          3187  3            EMIT_BYTES(M_R+R_1, OPC_RSB);
;  3148          3188  2            END;
;  3149          3189  2
;  3150          3190  2        [TYP_K_INDEX]:
;  3151          3191  3            BEGIN
;  3152          3192  3            LOCAL
;  3153          3193  3                Z;                          ! A temporary
;  3154          3194  3                !
;  3155          3195  3                ! The only fields we should output are:
;  3156          3196  3                !   RFA, FILE, OKEY, in that order.
;  3157          3197  3                ! The only other fields we may have in the internal record are:
;  3158          3198  3                !   KEY, STABLE
;  3159          3199  3                !
;  3160          3200  3            ASSERT_(COM_ORD_RFA  LSS COM_ORD_FILE)
;  3161          3201  3            ASSERT_(COM_ORD_FILE LSS COM_ORD_OKEY)
;  3162          3202  3                !
;  3163          3203  3                ! Assert that neither the KEY or STABLE fields
;  3164          3204  3                ! are between the RFA and OKEY fields.
;  3165          3205  3                !
;  3166          3206  3            ASSERT_((COM_ORD_KEY -COM_ORD_RFA) GTRU (COM_ORD_OKEY-COM_ORD_RFA))
;  3167          3207  3            ASSERT_((COM_ORD_STAB-COM_ORD_RFA) GTRU (COM_ORD_OKEY-COM_ORD_RFA))
;  3168          3208  3                !
;  3169          3209  3                ! Find the displacement of the first field after COM_ORD_OKEY.
;  3170          3210  3                ! We will find something, since DISP[COM_ORD_MAX] is geq 0.
;  3171          3211  3                !
;  3172          3212  3            INCR I FROM COM_ORD_OKEY+1 TO COM_ORD_MAX DO
;  3173          3213  3                IF (Z = .DISP[.I]) GEQ 0 THEN EXITLOOP;
;  3174          3214  3            CTX[COM_LRL_OUT] = .Z - .DISP[COM_ORD_RFA];
;  3175          3215  3            EMIT_BYTE(OPC_MOVZWL);
;  3176          3216  3            EMIT_LITE(K_WORD, .CTX[COM_LRL_OUT]);
;  3177        P 3217  3            EMIT_BYTES(M_R+R_0,
;  3178          3218  3                OPC_MOVAB);
;  3179          3219  3            EMIT_DISP(.DISP[COM_ORD_RFA], COM_REG_SRC2);
;  3180          3220  3            EMIT_BYTES(M_R+R_1, OPC_RSB);
;  3181          3221  2            END;
;  3182          3222  2
;  3183        U 3223  2        %ELSE
;  3184        U 3224  2
;  3185        U 3225  2        [INRANGE,OUTRANGE]:
;  3186        U 3226  2            RETURN SOR$$ERROR(SOR$_SHR_BADLOGIC);
;  3187        U 3227  2
;  3188          3228  2        %FI
;  3189          3229  2
;  3190          3230  2        TES;
;  3191          3231  2
;  3192          3232  2
;  3193          3233  2    ! Make sure there is a free byte following the last one we executed.
;  3194          3234  2    ! This avoids a 11/750 problem if the next byte is not readable.
;  3195          3235  2    !
```

```
;  3196          3236  2     ROOM(1);
;  3197          3237  2
;  3198          3238  2
;  3199          3239  2     ! Check for writing too far
;  3200          3240  2     !
;  3201          3241  2     VERIFY_LEN();
```

```
;  3203          3242  2     ! Adjust the entry points to the generated routines.
;  3204          3243  2     !
;  3205          3244  2     CTX[COM_INPUT]      = .CTX[COM_INPUT]      + .CTX[S_START];
;  3206          3245  2     CTX[COM_COMPARE]    = .CTX[COM_COMPARE]    + .CTX[S_START];
;  3207          3246  2     CTX[COM_EQUAL]      = .CTX[COM_EQUAL]      + .CTX[S_START];
;  3208          3247  2     CTX[COM_LENADR]     = .CTX[COM_LENADR]     + .CTX[S_START];
;  3209          3248  2
;  3210          3249  2     ! Is the COM_EQUAL routine really needed?
;  3211          3250  2     !
;  3212          3251  2     IF CH$RCHAR(.CTX[COM_EQUAL]) EQL OPC_HALT
;  3213          3252  2     THEN
;  3214          3253  2         CTX[COM_EQUAL] = 0;
;  3215          3254  2
;  3216        L 3255  2     %IF %SWITCHES(DEBUG)
;  3217        U 3256  2     %THEN
;  3218        U 3257  2         BEGIN
;  3219        U 3258  2         EXTERNAL ROUTINE
;  3220        U 3259  2             SOR$$OUTPUT;
;  3221        U 3260  2         MACRO
;  3222        U 3261  2             DESC_(A) = UPLIT(%CHARCOUNT(A), UPLIT BYTE(A)) %;
;  3223        U 3262  2         SOR$$OUTPUT(DESC_(%STRING(
;  3224        U 3263  2             'routine input,  !_!XL-!XL!/',
;  3225        U 3264  2             'routine compare,!_!XL-!XL!/',
;  3226        U 3265  2             'routine equal,  !_!XL-!XL!/',
;  3227        U 3266  2             'routine lenadr, !_!XL-!XL!/')),
;  3228        U 3267  2             .CTX[COM_INPUT],    .CTX[COM_COMPARE]-1,
;  3229        U 3268  2             .CTX[COM_COMPARE],  .CTX[COM_EQUAL]-1,
;  3230        U 3269  2             .CTX[COM_EQUAL],    .CTX[COM_LENADR]-1,
;  3231        U 3270  2             .CTX[COM_LENADR],
;  3232        U 3271  2             .VECTOR[ CTX[COM_ROUTINES], 0 ] +
;  3233        U 3272  2             .VECTOR[ CTX[COM_ROUTINES], 1 ] - 1 );
;  3234        U 3273  2         END;
;  3235          3274  2     %FI
;  3236          3275  2
;  3237          3276  2     ! Execute an REI instruction to guarantee that instruction prefetch gets
;  3238          3277  2     ! the instructions we've just written.
;  3239          3278  2     !
;  3240          3279  2     DO_REI();
;  3241          3280  2
;  3242          3281  2     RETURN SS$_NORMAL;                                                                    ;
;  3243          3282  2
;  3244          3283  1     END;                                                                                  ;

                                43  2C  15  00  00C7E P.AAH:  .BYTE   0, 21, 44, 67                                    ;


                                                    07FC 00000          .ENTRY  SOR$$KEY_SUB, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 2188
                                                                                R10
                                                                                                                       ;
                                5E      F79C    CE  9E 00002            MOVAB   -2148(SP), SP                          ;
                                53        04    AC  D0 00007            MOVL    KEY_BUFFER, R3                         ; 2294
                                                  22  13 0000B            BEQL    2$                                     ;
                                52              63  3C 0000D            MOVZWL  (R3), R2                               ; 2299
                                52              08  C4 00010            MULL2   #8, LEN                                ;
                                52              02  C0 00013            ADDL2   #2, LEN                                ;
```

```
                    000007FA  8F              52  D1 00016          CMPL    LEN, #2042                          ; 2300
                                              09  15 0001D          BLEQ    1$                                  ;
                          001C803C  8F          DD 0001F          PUSHL   #1867836                            ;
                                    080E      31 00025          BRW     102$                                ;
                OC  AE              63          52  28 00028 1$:  MOVC3   LEN, (R3), KEY_BUFF                 ; 2301
                                              1A  11 0002D          BRB     4$                                  ; 2294
                                              6B  D5 0002F 2$:      TSTL    (CTX)                               ; 2304
                                              05  13 00031          BEQL    3$                                  ;
                                    0C    AE  B4 00033          CLRW    KEY_BUFF                            ; 2306
                                              11  11 00036          BRB     4$                                  ;
                          0C    AE  000E0001  8F  D0 00038 3$:      MOVL    #917505, KEY_BUFF                   ; 2311
                                    10    AE  D4 00040          CLRL    KBF+2                               ; 2313
                                    14    AE  0084  CB  B0 00043  MOVW    132(CTX), KBF+6                     ; 2315
                                    0C    AE  9F 00049 4$:      PUSHAB  KEY_BUFF                            ; 2334
                              F365  CF          01  FB 0004C      CALLS   #1, TKS_HACK                        ;
                24  FF  8F          6E          00  2C 00051      MOVC5   #0, (SP), #255, #36, DISP           ; 2339
                                          A4  AD      00057                                                   ;
                                          C8  AD  D4  00059      CLRL    DISP+36                             ; 2340
                                    01    58  AB  91  0005C      CMPB    88(CTX), #1                         ; 2347
                                              1E  13  00060      BEQL    5$                                  ;
                                          A4  AD  9F  00062      PUSHAB  DISP                                ; 2350
                                              06  DD  00065      PUSHL   #6                                  ;
                                              7E  D4  00067      CLRL    -(SP)                               ;
                              FF66  CF          03  FB 00069      CALLS   #3, EXPAND                          ;
                                    01    59  AB  91  0006E      CMPB    89(CTX), #1                         ; 2351
                                              0C  1B  00072      BLEQU   5$                                  ;
                                          A4  AD  9F  00074      PUSHAB  DISP                                ; 2353
                                              01  DD  00077      PUSHL   #1                                  ;
                                              01  DD  00079      PUSHL   #1                                  ;
                              FF54  CF          03  FB 0007B      CALLS   #3, EXPAND                          ;
                                    01    58  AB  91  00080 5$:  CMPB    88(CTX), #1                         ; 2363
                                              36  12  00084      BNEQ    7$                                  ;
                                          A4  AD  9F  00086      PUSHAB  DISP                                ; 2366
                                    7E    0084  CB  3C  00089    MOVZWL  132(CTX), -(SP)                     ;
                                              08  DD  0008E      PUSHL   #8                                  ;
                              FF3F  CF          03  FB 00090      CALLS   #3, EXPAND                          ;
                                    50    0081  CB  9A  00095    MOVZBL  129(CTX), R0                        ; 2367
                                              0C  13  0009A      BEQL    6$                                  ;
                                          A4  AD  9F  0009C      PUSHAB  DISP                                ;
                                              50  DD  0009F      PUSHL   R0                                  ;
                                              07  DD  000A1      PUSHL   #7                                  ;
                              FF2C  CF          03  FB 000A3      CALLS   #3, EXPAND                          ;
                20    0080  CB              01  E1  000A8 6$:   BBC     #1, 128(CTX), 9$                    ; 2368
                                          A4  AD  9F  000AE      PUSHAB  DISP                                ;
                                              02  DD  000B1      PUSHL   #2                                  ;
                                              06  DD  000B3      PUSHL   #6                                  ;
                              FF1A  CF          03  FB 000B5      CALLS   #3, EXPAND                          ;
                                              12  11  000BA      BRB     9$                                  ; 2363
                                              6B  D5  000BC 7$:  TSTL    (CTX)                               ; 2375
                                              05  12  000BE      BNEQ    8$                                  ;
                                    04    AB  D5  000C0      TSTL    4(CTX)                              ;
                                              09  13  000C3      BEQL    9$                                  ;
                          001C806C  8F          DD 000C5 8$:      PUSHL   #1867884                            ; 2377
                                    0768      31 000CB          BRW     102$                                ;
                01    0083  CB              91  000CE 9$:      CMPB    131(CTX), #1                        ; 2385
                                              0C  1B  000D3      BLEQU   10$                                 ;
                                          A4  AD  9F  000D5      PUSHAB  DISP                                ; 2387
```

```
                                01 DD 000D8          PUSHL   #1                              ;
                                02 DD 000DA          PUSHL   #2                              ;
          FEF3  CF              03 FB 000DC          CALLS   #3, EXPAND                      ;
                0C        5B    AB E9 000E1  10$:    BLBC    91(CTX), 11$                    ; 2394
                          A4    AD 9F 000E5          PUSHAB  DISP                            ; 2396
                                04 DD 000E8          PUSHL   #4                              ;
                                04 DD 000EA          PUSHL   #4                              ;
          FEE3  CF              03 FB 000EC          CALLS   #3, EXPAND                      ;
                                6B D5 000F1  11$:    TSTL    (CTX)                           ; 2401
                                03 13 000F3          BEQL    12$                             ;
                              015F 31 000F5          BRW     38$                             ;
                56      0084    CB 9E 000F8  12$:    MOVAB   132(CTX), R6                    ; 2464
                54        0C    AE 3C 000FD          MOVZWL  KEY_BUFF, I                     ;
                              014B 31 00101          BRW     36$                             ;
                53     0E AE44  7E 00104  13$:    MOVAQ   KEY_BUFF+2[I], KBF              ; 2416
                01        02    A3 B1 00109          CMPW    2(KBF), #1                      ; 2421
                                11 1B 0010D          BLEQU   14$                             ;
                      001C8034  8F DD 0010F          PUSHL   #1867828                        ; 2424
       00000000G  00            01 FB 00115          CALLS   #1, SOR$$ERROR                  ;
              02  A3            01 8A 0011C          BICB2   #1, 2(KBF)                      ; 2425
                50              63 3C 00120  14$:    MOVZWL  (KBF), R0                       ; 2432
                23              50 B1 00123          CMPW    R0, #35                         ;
                                05 1B 00126          BLEQU   15$                             ;
                33             00G E9 00128          BLBC    S^FUN_K_KANJI, 17$              ;
                                43 11 0012B          BRB     18$                             ;
                51        06    A3 3C 0012D  15$:    MOVZWL  6(KBF), R1                      ; 2434
 51      F246 CF40  08          00 EC 00131          CMPV    #0, #8, DSC_LENGTH[R0], R1      ;
                                23 1F 00139          BLSSU   17$                             ;
      2F      F262  CF          50 E1 0013B          BBC     R0, DSC_FORCE, 18$              ; 2436
                          06    A3 B5 00141          TSTW    6(KBF)                          ; 2439
                                07 12 00144          BNEQ    16$                             ;
              06  A3      F233 CF40 99 00146          CVTBW   DSC_LENGTH[R0], 6(KBF)          ; 2441
                50              63 3C 0014D  16$:    MOVZWL  (KBF), R0                       ; 2442
                51        06    A3 3C 00150          MOVZWL  6(KBF), R1                      ;
 51      F223 CF40  08          00 EC 00154          CMPV    #0, #8, DSC_LENGTH[R0], R1      ;
                                12 13 0015C          BEQL    18$                             ;
                      001C8034  8F DD 0015E  17$:    PUSHL   #1867828                        ; 2451
       00000000G  00            01 FB 00164          CALLS   #1, SOR$$ERROR                  ;
                                63 B4 0016B          CLRW    (KBF)                           ; 2452
                          06    A3 B4 0016D          CLRW    6(KBF)                          ; 2453
                55        06    A3 9E 00170  18$:    MOVAB   6(KBF), R5                      ; 2459
                15              63 B1 00174          CMPW    (KBF), #21                      ;
                                0C 12 00177          BNEQ    19$                             ;
                50              65 3C 00179          MOVZWL  (R5), R0                        ;
                50              02 C6 0017C          DIVL2   #2, R0                          ;
                52        01    A0 9E 0017F          MOVAB   1(R0), KEYLEN                   ;
                                03 11 00183          BRB     20$                             ;
                52              65 3C 00185  19$:    MOVZWL  (R5), KEYLEN                    ;
                50        04    A3 3C 00188  20$:    MOVZWL  4(KBF), R0                      ; 2464
                50              52 C0 0018C          ADDL2   KEYLEN, R0                      ;
 50         66      10          00 ED 0018F          CMPZV   #0, #16, (R6), R0               ;
                                4F 18 00194          BGEQ    29$                             ;
                                63 B5 00196          TSTW    (KBF)                           ; 2471
                                0A 13 00198          BEQL    21$                             ;
                0E              63 B1 0019A          CMPW    (KBF), #14                      ; 2472
                                05 13 0019D          BEQL    21$                             ;
                23              63 B1 0019F          CMPW    (KBF), #35                      ; 2473
```

```
                                   0E  1B 001A2        BLEQU   22$                              ;
                         52        66  3C 001A4 21$:   MOVZWL  (R6), KEYLEN                     ; 2476
                         50    04  A3  3C 001A7        MOVZWL  4(KBF), R0                       ;
                         52        50  C2 001AB        SUBL2   R0, KEYLEN                       ;
                                   06  18 001AE        BGEQ    24$                              ; 2477
                                   02  11 001B0        BRB     23$                              ;
                                   63  B4 001B2 22$:   CLRW    (KBF)                            ; 2481
                                   52  D4 001B4 23$:   CLRL    KEYLEN                           ; 2482
               FFFF      8F        65  B1 001B6 24$:   CMPW    (R5), #65535                     ; 2490
                                   04  12 001BB        BNEQ    25$                              ;
                                   52  D5 001BD        TSTL    KEYLEN                           ; 2491
                                   21  12 001BF        BNEQ    28$                              ;
                         7E        65  3C 001C1 25$:   MOVZWL  (R5), -(SP)                      ; 2497
                               01  A4  9F 001C4        PUSHAB  1(I)                             ;
                                   02  DD 001C7        PUSHL   #2                               ; 2493
                                   52  D5 001C9        TSTL    KEYLEN                           ; 2494
                                   08  12 001CB        BNEQ    26$                              ;
                         001C80AA  8F  DD 001CD        PUSHL   #1867946                         ; 2495
                                   06  11 001D3        BRB     27$                              ;
                         001C80AB  8F  DD 001D5 26$:   PUSHL   #1867947                         ; 2496
           00000000G     00        04  FB 001DB 27$:   CALLS   #4, SOR$$ERROR                   ; 2494
                         65        52  B0 001E2 28$:   MOVW    KEYLEN, (R5)                     ; 2501
                         50    04  A3  3C 001E5 29$:   MOVZWL  4(KBF), R0                       ; 2508
                         50        52  C0 001E9        ADDL2   KEYLEN, R0                       ;
  50       02    A6      10        00  ED 001EC        CMPZV   #0, #16, 2(R6), R0               ;
                                   12  18 001F2        BGEQ    30$                              ;
                         0E        63  B1 001F4        CMPW    (KBF), #14                       ; 2509
                                   0D  13 001F7        BEQL    30$                              ;
                                   63  B5 001F9        TSTW    (KBF)                            ; 2510
                                   09  13 001FB        BEQL    30$                              ;
                         23        63  B1 001FD        CMPW    (KBF), #35                       ; 2511
                                   04  1A 00200        BGTRU   30$                              ;
                 02      A6        50  B0 00202        MOVW    R0, 2(R6)                        ; 2513
                         03    58  AB  91 00206 30$:   CMPB    88(CTX), #3                      ; 2518
                                   0C  12 0020A        BNEQ    31$                              ;
                               A4  AD  9F 0020C        PUSHAB  DISP                             ; 2520
                                   52  DD 0020F        PUSHL   KEYLEN                           ;
                                   03  DD 00211        PUSHL   #3                               ;
               FDBC      CF        03  FB 00213        CALLS   #3, EXPAND                       ;
                         0E        63  B1 00218 31$:   CMPW    (KBF), #14                       ; 2527
                                   07  12 0021B        BNEQ    32$                              ;
                               68  AB  D5 0021D        TSTL    104(CTX)                         ; 2528
                                   02  12 00220        BNEQ    32$                              ;
                                   63  B4 00222        CLRW    (KBF)                            ; 2530
                                   63  B5 00224 32$:   TSTW    (KBF)                            ; 2531
                                   07  12 00226        BNEQ    33$                              ;
                         01        65  B1 00228        CMPW    (R5), #1                         ; 2534
                                   22  12 0022B        BNEQ    36$                              ;
                                   18  11 0022D        BRB     34$                              ;
                         23        63  B1 0022F 33$:   CMPW    (KBF), #35                       ; 2536
                                   1B  1A 00232        BGTRU   36$                              ;
                         50        63  3C 00234        MOVZWL  (KBF), R0                        ; 2539
           12    F16E    CF        50  E1 00237        BBC     R0, DSC_BINARY, 36$              ;
           50 3C000040   8F        63  78 0023D        ASHL    (KBF), #1006633024, R0           ; 2543
                                   05  18 00245        BGEQ    35$                              ;
                         63        02  B0 00247 34$:   MOVW    #2, (KBF)                        ; 2544
                                   03  11 0024A        BRB     36$                              ;
```

```
                63              06  B0 0024C 35$:    MOVW    #6, (KBF)                                   ; 2545
                02              54  F4 0024F 36$:    SOBGEQ  I, 37$                                      ; 2407
                                03  11 00252         BRB     38$
                            FEAD    31 00254 37$:    BRW     13$
                          0C    AE  9F 00257 38$:    PUSHAB  KEY_BUFF                                    ; 2555
                      F1AF      CF  01 FB 0025A      CALLS   #1, KEY_COMPRESS
                        50      18  AB 9E 0025F      MOVAB   24(CTX), R0                                 ; 2560
                                5A  D4 00263         CLRL    CUR_PC                                      ; 2561
                                60  7C 00265         CLRQ    (R0)                                        ; 2560
                56              5A  1C AB C3 00267   SUBL3   28(CTX), CUR_PC, TMP                        ; 2578
                                6E  D4 0026C         CLRL    STACK                                       ; 2583
                                6B  D5 0026E         TSTL    (CTX)                                       ; 2594
                                03  13 00270         BEQL    39$
                            013D    31 00272         BRW     53$
                                54  7C 00275 39$:    CLRQ    CVTCNT                                      ; 2608
                                58  0C AE 3C 00277   MOVZWL  KEY_BUFF, R8                               ; 2609
                                57  01 CE 0027B      MNEGL   #1, I
                            00CA    31 0027E         BRW     47$
                                52  0E AE47 7E 00281 40$:    MOVAQ   KEY_BUFF+2[I], KBF                  ; 2614
                                03  58 AB 91 00286   CMPB    88(CTX), #3                                 ; 2621
                                    3E 12 0028A      BNEQ    43$
                                15      62 B1 0028C  CMPW    (KBF), #21                                  ; 2627
                                        0B 12 0028F  BNEQ    41$
                                53  06 A2 3C 00291   MOVZWL  6(KBF), R3
                                53      02 C6 00295  DIVL2   #2, R3
                                        53 D6 00298  INCL    KEYLEN
                                        04 11 0029A  BRB     42$
                                53  06 A2 3C 0029C 41$:    MOVZWL  6(KBF), KEYLEN
                                50  42 8F 9A 002A0 42$:    MOVZBL  #66, R0                             ; 2628
                                    F48E 30 002A4        BSBW    ROOM
                                        0A DD 002A7      PUSHL   #10                                     ; 2629
                                    B0 BD45 9F 002A9     PUSHAB  @DISP+12[OKECNT]                        ; 2631
                                        09 DD 002AD      PUSHL   #9                                      ; 2629
                                7E      04 A2 3C 002AF   MOVZWL  4(KBF), -(SP)                           ; 2630
                                        53 DD 002B3      PUSHL   KEYLEN                                  ; 2629
                            F506    CF  05 FB 002B5      CALLS   #5, GEN_MOVE_VAR
                50              55      B0 AD C1 002BA   ADDL3   DISP+12, OKECNT, R0                     ; 2636
                          04    A2      50 B0 002BF      MOVW    R0, 4(KBF)
                          02    A2      02 88 002C3      BISB2   #2, 2(KBF)                              ; 2637
                                55      53 C0 002C7      ADDL2   KEYLEN, OKECNT                          ; 2638
                                23      62 B1 002CA 43$:    CMPW    (KBF), #35                           ; 2645
                                        19 1B 002CD      BLEQU   44$
                                    A4  AD 9F 002CF      PUSHAB  DISP                                    ; 2648
                                7E      05 7D 002D2      MOVQ    #5, -(SP)
                            FCFA    CF  03 FB 002D5      CALLS   #3, EXPAND
                                    B8 BD44 9F 002DA     PUSHAB  @DISP+20[CVTCNT]                        ; 2650
                                        52 DD 002DE      PUSHL   KBF
                            F88A    CF  02 FB 002E0      CALLS   #2, GEN_CONVERT_UDEF
                                54      50 C0 002E5      ADDL2   R0, CVTCNT
                50 0001F800     8F      62 78 002E8 44$:    ASHL    (KBF), #129024, R0                  ; 2659
                                        1A 18 002F0      BGEQ    45$
                                    A4  AD 9F 002F2      PUSHAB  DISP                                    ; 2662
                                7E      05 7D 002F5      MOVQ    #5, -(SP)
                            FCD7    CF  03 FB 002F8      CALLS   #3, EXPAND
                                        5E DD 002FD      PUSHL   SP                                      ; 2664
                                    B8 BD44 9F 002FF     PUSHAB  @DISP+20[CVTCNT]
                                        52 DD 00303      PUSHL   KBF
```

```
                        F57D   CF              03  FB 00305          CALLS   #3, GEN_CONVERT_DEC                    ;
                                               3C  11 0030A          BRB     46$                                   ;
                50 00300018    8F              62  78 0030C 45$:     ASHL    (KBF), #3145752, R0                   ; 2669
                                               35  18 00314          BGEQ    47$                                   ;
                               50              62  3C 00316          MOVZWL  (KBF), R0                             ; 2670
             7E                00              01  7A 00319          EMUL    #1, R0, #0, -(SP)                     ;
             50                50              05  7B 0031E          EDIV    #5, (SP)+, R0, R0                     ;
                               50       FCD4 CF40  9A 00323          MOVZBL  P.AAH[R0], R0                         ;
                        F15B CF40               00  FB 00329          CALLS   #0, FFLT_HARDWARE[R0]                 ;
                               19               50  E8 0032F          BLBS    R0, 47$                               ;
                                          A4    AD  9F 00332          PUSHAB  DISP                                  ; 2673
                               7E               05  7D 00335          MOVQ    #5, -(SP)                             ;
                        FC97   CF               03  FB 00338          CALLS   #3, EXPAND                            ;
                                          B8 BD44  9F 0033D          PUSHAB  @DISP+20[CVTCNT]                      ; 2675
                                               52  DD 00341          PUSHL   KBF                                   ;
                        F756   CF              02  FB 00343          CALLS   #2, GEN_CONVERT_FLT                   ;
                               54              50  C0 00348 46$:     ADDL2   R0, CVTCNT                            ;
                02             57              58  F2 0034B 47$:     AOBLSS  R8, I, 48$                            ; 2609
                                               03  11 0034F          BRB     49$                                   ;
                                        FF2D   31  31 00351 48$:     BRW     40$                                   ;
                                          C4    AD  D5 00354 49$:     TSTL    DISP+32                               ; 2682
                                               1A  18 00357          BGEQ    50$                                   ;
                                          A4    AD  9F 00359          PUSHAB  DISP                                  ; 2685
                               7E               05  7D 0035C          MOVQ    #5, -(SP)                             ;
                        FC70   CF               03  FB 0035F          CALLS   #3, EXPAND                            ;
                                          B8 BD44  9F 00364          PUSHAB  @DISP+20[CVTCNT]                      ; 2688
                                          10    AE  9F 00368          PUSHAB  KEY_BUFF                              ; 2687
                        FBA4   CF              02  FB 0036B          CALLS   #2, MOVE_KEYS                         ;
                               54              50  C0 00370          ADDL2   R0, CVTCNT                            ;
                50             6E              03  C1 00373 50$:     ADDL3   #3, STACK, R0                         ; 2694
                6E             50              03  CB 00377          BICL3   #3, R0, STACK                         ;
                                          B8    AD  D5 0037B          TSTL    DISP+20                               ; 2700
                                               05  18 0037E          BGEQ    51$                                   ;
                                          B0    AD  D5 00380          TSTL    DISP+12                               ;
                                               2D  19 00383          BLSS    53$                                   ;
                                          A4    AD  9F 00385 51$:     PUSHAB  DISP                                  ; 2703
                                               54  DD 00388          PUSHL   CVTCNT                                ;
                                               05  DD 0038A          PUSHL   #5                                    ;
                        FC43   CF              03  FB 0038C          CALLS   #3, EXPAND                            ;
                               50              0B  D0 00391          MOVL    #11, R0                               ; 2704
                                        F39E   30  30 00394          BSBW    ROOM                                  ;
                                               6E  D5 00397          TSTL    STACK                                 ; 2705
                                               10  13 00399          BEQL    52$                                   ;
                               8A              3F  92 0039B          MCOMB   #63, (CUR_PC)+                        ; 2708
                               53              6E  D0 0039E          MOVL    STACK, R3                             ; 2709
                               52              04  D0 003A1          MOVL    #4, R2                                ;
                                        F2D1   30  30 003A4          BSBW    EMIT_LITE                             ;
                               8A        5E    8F  90 003A7          MOVB    #94, (CUR_PC)+                        ; 2710
                               8A 055001D0     8F  D0 003AB 52$:     MOVL    #89129424, (CUR_PC)+                  ; 2712
          08   AB              5A        1C    AB  C3 003B2 53$:     SUBL3   28(CTX), CUR_PC, 8(CTX)               ; 2720
                                      0104     CB  D5 003B8          TSTL    260(CTX)                              ; 2727
                                               2C  13 003BC          BEQL    54$                                   ;
                               50              22  D0 003BE          MOVL    #34, R0                               ; 2730
                                        F371   30  30 003C1          BSBW    ROOM                                  ;
                               8A 699F7EDF     8F  D0 003C4          MOVL    #1772060383, (CUR_PC)+                ; 2734
                               8A       02FB   8F  B0 003CB          MOVW    #763, (CUR_PC)+                       ;
                               8A         9F   8F  90 003D0          MOVB    #-97, (CUR_PC)+                       ; 2735
```

```
                8A 00000000G   00  9E 003D4          MOVAB   SOR$$RDT, (CUR_PC)+
                8A E8518ED0    8F  D0 003DB          MOVL    #-397308208, (CUR_PC)+          ;2739
                8A      0150   8F  B0 003E2          MOVW    #336, (CUR_PC)+
                8A             05  90 003E7          MOVB    #5, (CUR_PC)+
                54        BC   AD  D0 003EA 54$:     MOVL    DISP+24, R4                     ;2757
                               2C  19 003EE          BLSS    57$
                50             0C  D0 003F0          MOVL    #12, R0                         ;2760
                           F33F    30 003F3          BSBW    ROOM
                          78   AB  95 003F6          TSTB    120(CTX)                        ;2761
                               10  13 003F9          BEQL    55$
                8A        A3   8F  90 003FB          MOVB    #-93, (CUR_PC)+                 ;2764
                53        78   AB  9A 003FF          MOVZBL  120(CTX), R3                    ;2765
                52             02  D0 00403          MOVL    #2, R2
                           F26F    30 00406          BSBW    EMIT_LITE
                               04  11 00409          BRB     56$                             ;2761
                8A        B0   8F  90 0040B 55$:     MOVB    #-80, (CUR_PC)+                 ;2769
                8A        69   8F  90 0040F 56$:     MOVB    #105, (CUR_PC)+                 ;2771
                53             0A  D0 00413          MOVL    #10, R3                         ;2772
                52             54  D0 00416          MOVL    R4, R2
                           F206    30 00419          BSBW    EMIT_DISP
                57        C4   AD  D0 0041C 57$:     MOVL    DISP+32, R7                     ;2778
                               4C  19 00420          BLSS    59$
                50        46   8F  9A 00422          MOVZBL  #70, R0                         ;2781
                           F30C    30 00426          BSBW    ROOM
                               54  D5 00429          TSTL    R4                              ;2782
                               1B  18 0042B          BGEQ    58$
                8A 5104A9D0    8F  D0 0042D          MOVL    #1359260112, (CUR_PC)+          ;2788
                               0A  DD 00434          PUSHL   #10                             ;2789
                               57  DD 00436          PUSHL   R7                              ;2791
                               01  DD 00438          PUSHL   #1                              ;2789
                               7E  D4 0043A          CLRL    -(SP)
                7E      0084   CB  3C 0043C          MOVZWL  132(CTX), -(SP)
        F37A    CF             05  FB 00441          CALLS   #5, GEN_MOVE
                               26  11 00446          BRB     59$                             ;2782
                8A 04B9692C    8F  D0 00448 58$:     MOVL    #79259948, (CUR_PC)+            ;2797
                53      0101   CB  9A 0044F          MOVZBL  257(CTX), R3                    ;2798
                52             01  D0 00454          MOVL    #1, R2
                           F21E    30 00457          BSBW    EMIT_LITE
                53      0084   CB  3C 0045A          MOVZWL  132(CTX), R3                    ;2799
                52             02  D0 0045F          MOVL    #2, R2
                           F213    30 00462          BSBW    EMIT_LITE
                53             0A  D0 00465          MOVL    #10, R3                         ;2801
                52             57  D0 00468          MOVL    R7, R2
                           F1B4    30 0046B          BSBW    EMIT_DISP
                          B4   AD  D5 0046E 59$:     TSTL    DISP+16                         ;2808
                               28  19 00471          BLSS    62$
                50             0B  D0 00473          MOVL    #11, R0                         ;2811
                           F2BC    30 00476          BSBW    ROOM
                8A             30  8E 00479          MNEGB   #48, (CUR_PC)+                  ;2812
                          5C   AB  95 0047C          TSTB    92(CTX)                         ;2814
                               06  18 0047F          BGEQ    60$
                52        64   8F  9A 00481          MOVZBL  #100, R2                        ;2815
                               04  11 00485          BRB     61$
                52        7C   8F  9A 00487 60$:     MOVZBL  #124, R2                        ;2816
                53             0B  D0 0048B 61$:     MOVL    #11, R3                         ;2814
                           F191    30 004 _          BSBW    EMIT_DISP
                53             0A  D0 00491          MOVL    #10, R3                         ;2817
```

```
52          B4    AD   D0 00494          MOVL    DISP+16, R2                                  ;
                F187   30 00498          BSBW    EMIT_DISP                                    ;
54          A4    AD   D0 0049B 62$:     MOVL    DISP, R4                                     ; 2823
                  5C   19 0049F          BLSS    63$                                          ;
50                1A   D0 004A1          MOVL    #26, R0                                      ; 2828
                F28E   30 004A4          BSBW    ROOM                                         ;
8A                3F   8E 004A7          MNEGB   #63, (CUR_PC)+                               ; 2829
53                0B   D0 004AA          MOVL    #11, R3                                      ; 2831
52          A0    8F   9A 004AD          MOVZBL  #160, R2                                     ;
                F16E   30 004B1          BSBW    EMIT_DISP                                    ;
53                24   D0 004B4          MOVL    #36, R3                                      ; 2832
52                04   D0 004B7          MOVL    #4, R2                                       ;
                F1BB   30 004BA          BSBW    EMIT_LITE                                    ;
8A        D050    8F   B0 004BD          MOVW    #-12208, (CUR_PC)+                           ; 2834
8A          80    8F   90 004C2          MOVB    #-128, (CUR_PC)+                             ; 2835
53                0A   D0 004C6          MOVL    #10, R3                                      ;
52                54   D0 004C9          MOVL    R4, R2                                       ;
                F153   30 004CC          BSBW    EMIT_DISP                                    ;
8A        60B0    8F   B0 004CF          MOVW    #24752, (CUR_PC)+                            ; 2836
52          04    A4   9E 004D4          MOVAB   4(R4), R2                                    ; 2837
53                0A   D0 004D8          MOVL    #10, R3                                      ;
                F144   30 004DB          BSBW    EMIT_DISP                                    ;
              A8  AD   D5 004DE          TSTL    DISP+4                                       ; 2841
                  1A   19 004E1          BLSS    63$                                          ;
50                0B   D0 004E3          MOVL    #11, R0                                      ; 2844
                F24C   30 004E6          BSBW    ROOM                                         ;
8A          90    8F   90 004E9          MOVB    #-112, (CUR_PC)+                             ; 2845
52                30   7D 004ED          MOVQ    #48, R2                                      ; 2849
                F12F   30 004F0          BSBW    EMIT_DISP                                    ;
53                0A   D0 004F3          MOVL    #10, R3                                      ; 2850
52          A8    AD   D0 004F6          MOVL    DISP+4, R2                                   ;
                F125   30 004FA          BSBW    EMIT_DISP                                    ;
              C0  AD   D5 004FD 63$:     TSTL    DISP+28                                      ; 2860
                  34   19 00500          BLSS    64$                                          ;
50          4B    8F   9A 00502          MOVZBL  #75, R0                                      ; 2865
                F22C   30 00506          BSBW    ROOM                                         ;
8A                30   8E 00509          MNEGB   #48, (CUR_PC)+                               ; 2866
53                0B   D0 0050C          MOVL    #11, R3                                      ; 2867
52          8C    8F   9A 0050F          MOVZBL  #140, R2                                     ;
                F10C   30 00513          BSBW    EMIT_DISP                                    ;
8A        1350    8F   B0 00516          MOVW    #4944, (CUR_PC)+                             ; 2870
                  8A   94 0051B          CLRB    (CUR_PC)+                                    ;
52                5A   D0 0051D          MOVL    CUR_PC, TMP2                                 ; 2871
                  0A   DD 00520          PUSHL   #10                                          ; 2872
              C0  AD   DD 00522          PUSHL   DISP+28                                      ; 2873
                  7E   7C 00525          CLRQ    -(SP)                                        ; 2872
7E        0081    CB   9A 00527          MOVZBL  129(CTX), -(SP)                              ;
      F28F    CF         05 FB 0052C     CALLS   #5, GEN_MOVE                                 ;
FF  A2            5A         52 83 00531 SUBB3   TMP2, CUR_PC, -1(TMP2)                       ; 2874
                  50         11 D0 00536 64$:    MOVL    #17, R0                              ; 2880
                F1F9   30 00539          BSBW    ROOM                                         ;
56          1C    AB   C0 0053C          ADDL2   28(CTX), TMP                                 ; 2881
            B8    AD   D5 00540          TSTL    DISP+20                                      ; 2882
                  05   18 00543          BGEQ    65$                                          ;
            B0    AD   D5 00545          TSTL    DISP+12                                      ;
                  4C   19 00548          BLSS    68$                                          ;
    055001D0    8F         66 D1 0054A 65$:    CMPL    (TMP), #89129424                     ; 2883
```

```
                                                43  13 00551          BEQL    68$                                                 ;
                                                6E  D5 00553          TSTL    STACK                                               ; 2888
                                                10  13 00555          BEQL    66$
                                          8A    3E  8E 00557          MNEGB   #62, (CUR_PC)+                                      ; 2891
                                          53    6E  D0 0055A          MOVL    STACK, R3                                           ; 2892
                                          52    04  D0 0055D          MOVL    #4, R2
                                              F115  30 00560          BSBW    EMIT_LITE
                                          8A    5E  8F 90 00563       MOVB    #94, (CUR_PC)+                                      ; 2893
                                          8A D05669B0 8F D0 00567 66$: MOVL    #-799643216, (CUR_PC)+                              ; 2899
                                          8A  04A9  8F B0 0056E       MOVW    #1193, (CUR_PC)+
                                          8A    59  8F 90 00573       MOVB    #89, (CUR_PC)+
                        50                56        5A  C3 00577      SUBL3   CUR_PC, TMP, R0                                     ; 2900
                                          50        03  C2 0057B      SUBL2   #3, Z
                50      50                08        00  EC 0057E      CMPV    #0, #8, Z, Z                                        ; 2901
                                                    09  12 00583      BNEQ    67$
                                          8A        11  90 00585      MOVB    #17, (CUR_PC)+                                      ; 2902
                        8A                50        01  81 00588      ADDB3   #1, Z, (CUR_PC)+
                                                    0F  11 0058C      BRB     69$                                                 ; 2901
                                          8A        31  90 0058E 67$:  MOVB    #49, (CUR_PC)+                                      ; 2903
                                          8A        50  B0 00591      MOVW    Z, (CUR_PC)+
                                                    07  11 00594      BRB     69$                                                 ; 2882
                                          8A 055001D0 8F  D0 00596 68$: MOVL    #89129424, (CUR_PC)+                                ; 2907
                                          50      18  AB  9E 0059D 69$: MOVAB   24(CTX), R0                                       ; 2909
                        50                60      04  A0  C1 005A1    ADDL3   4(R0), (R0), R0
                                          50          5A  D1 005A6      CMPL    CUR_PC, R0
                                                      03  1B 005A9      BLEQU   70$
                                                  0282  31 005AB      BRW     101$
                        56                5A      1C  AB  C3 005AE 70$: SUBL3   28(CTX), CUR_PC, TMP                             ; 2918
                34      FF  8F            6E          00  2C 005B3      MOVC5   #0, (SP), #255, #52, BRANCHES                       ; 2923
                                                    CC  AD    005B9
                                          59        CC  AD  9E 005BB      MOVAB   BRANCHES, BRANCH                                    ; 2924
                                                      69  D4 005BF      CLRL    (BRANCH)                                            ; 2929
                                                      6B  D5 005C1      TSTL    (CTX)                                               ; 2934
                                                      23  13 005C3      BEQL    72$
                                          50          29  D0 005C5      MOVL    #41, R0                                             ; 2937
                                                  F16A  30 005C8      BSBW    ROOM
                                                    A4  AD  9F 005CB      PUSHAB  DISP                                                ; 2938
                                                      6B  DD 005CE      PUSHL   (CTX)
                                      F0C5        CF          02  FB 005D0      CALLS   #2, EMIT_CALL4
                                                    B4  AD  D5 005D5      TSTL    DISP+16                                             ; 2939
                                                      09  19 005D8      BLSS    71$
                                          8A 050150E9 8F  D0 005DA      MOVL    #83972329, (CUR_PC)+                                ; 2942
                                                      27  11 005E1      BRB     76$                                                 ; 2944
                                          8A          05  90 005E3 71$: MOVB    #5, (CUR_PC)+
                                                      22  11 005E6      BRB     76$                                                 ; 2934
                                          53        0C    AE  3C 005E8 72$: MOVZWL  KEY_BUFF, R3                                    ; 2948
                                          52              01  CE 005EC      MNEGL   #1, I
                                                          15  11 005EF      BRB     75$
                                          50      0E AE42   7E  9F 005F1 73$: MOVAQ   KEY_BUFF+2[I], KBF                          ; 2954
                        04      02        A0              01  E4 005F6      BBSC    #1, 2(KBF), 74$                                 ; 2955
                                04        A0              57  A0 005FB      ADDW2   R7, 4(KBF)                                      ; 2962
                                                          05  BB 005FF 74$: PUSHR   #^M<R0,R2>                                      ; 2964
                                      F6BB        CF          02  FB 00601      CALLS   #2, GEN_COMPARE
                                E7                52          53  F2 00606 75$: AOBLSS  R3, I, 73$                                ; 2948
                                                    B4  AD  D5 0060A 76$: TSTL    DISP+16                                         ; 2971
                                                      15  19 0060D      BLSS    77$
                                          04        AE          02  D0 0060F      MOVL    #2, KBF                                     ; 2979
```

```
                    08   AE        B4   AD  B0 00613          MOVW    DISP+16, KBF+4                          ; 2981
                    0A   AE             04  B0 00618          MOVW    #4, KBF+6                               ; 2982
                              04    AE   9F 0061C             PUSHAB  KBF                                     ; 2983
                  F69D   CF             01  FB 0061F          CALLS   #1, GEN_COMPARE
                  0088   CB        C8   AD  B0 00624 77$:     MOVW    DISP+36, 136(CTX)                       ; 2989
             0000FFFF    8F        C8   AD  D1 0062A          CMPL    DISP+36, #65535                         ; 2990
                                    0D  15 00632              BLEQ    78$
                         001C1124   8F   DD 00634             PUSHL   #1839396                                ; 2992
             00000000G   00             01  FB 0063A          CALLS   #1, SOR$$ERROR
                         50             05  D0 00641 78$:     MOVL    #5, R0                                  ; 2997
                                   F0EE  30 00644             BSBW    ROOM
                         8A   50D4   8F   B0 00647             MOVW    #20692, (CUR_PC)+                       ; 2998
                                     69  D5 0064C             TSTL    (BRANCH)                                ; 2999
                                     0A  13 0064E             BEQL    79$
           50            69             08  78 00650          ASHL    #8, (BRANCH), R0
           8A            50   00BA      8F  A1 00654          ADDW3   #186, R0, (CUR_PC)+
                         8A             05  90 0065A 79$:     MOVB    #5, (CUR_PC)+                           ; 3000
                         6B             56  D0 0065D          MOVL    TMP, (CTX)                              ; 3005
                         50        18   AB  9E 00660          MOVAB   24(CTX), R0                             ; 3010
           50            60        04   A0  C1 00664          ADDL3   4(R0), (R0), R0
                         50             5A  D1 00669          CMPL    CUR_PC, R0
                                        7C  1A 0066C          BGTRU   83$
           56            5A        1C   AB  C3 0066E          SUBL3   28(CTX), CUR_PC, TMP                    ; 3018
                                   04   AB  D5 00673          TSTL    4(CTX)                                  ; 3023
                                        48  13 00676          BEQL    80$
                         50             3E  D0 00678          MOVL    #62, R0                                 ; 3026
                                   F0B7  30 0067B             BSBW    ROOM
                                   A4   AD  9F 0067E          PUSHAB  DISP                                    ; 3027
                                   04   AB  DD 00681          PUSHL   4(CTX)
                    F011   CF             02  FB 00684          CALLS   #2, EMIT_CALL4
                         8A   050150E9    8F  D0 00689          MOVL    #83972329, (CUR_PC)+                    ; 3032
                         8A   00DD50DD    8F  D0 00690          MOVL    #14504157, (CUR_PC)+
                         8A             23  8E 00697          MNEGB   #35, (CUR_PC)+
                         53   001C812A    8F  D0 0069A          MOVL    #1868074, R3                            ; 3033
                         52             04  D0 006A1          MOVL    #4, R2
                                   EFD1  30 006A4             BSBW    EMIT_LITE
                         8A        03FB   8F  B0 006A7          MOVW    #1019, (CUR_PC)+                        ; 3034
                         8A          9F   8F  90 006AC          MOVB    #-97, (CUR_PC)+                         ; 3035
                         8A   00000000G   00  9E 006B0          MOVAB   SOR$$ERROR, (CUR_PC)+
                         8A   055001D0    8F  D0 006B7          MOVL    #89129424, (CUR_PC)+                    ; 3037
                                        1A  11 006BE          BRB     82$                                     ; 3023
              13         5B   AB             05  E1 006C0 80$:     BBC     #5, 91(CTX), 81$                        ; 3039
                         8A        8FD0   8F  B0 006C5          MOVW    #-28720, (CUR_PC)+                      ; 3042
                         8A   001C8111    8F  D0 006CA          MOVL    #1868049, (CUR_PC)+                     ; 3043
                         8A        0550   8F  B0 006D1          MOVW    #1360, (CUR_PC)+                        ; 3045
                                        02  11 006D6          BRB     82$                                     ; 3038
                                        8A  94 006D8 81$:     CLRB    (CUR_PC)+                               ; 3053
                    04   AB             56  D0 006DA 82$:     MOVL    TMP, 4(CTX)                             ; 3059
                         50        18   AB  9E 006DE          MOVAB   24(CTX), R0                             ; 3064
              50            60        04   A0  C1 006E2          ADDL3   4(R0), (R0), R0
                         50             5A  D1 006E7          CMPL    CUR_PC, R0
                                        03  1B 006EA 83$:     BLEQU   84$
                                   0141  31 006EC             BRW     101$
          10   AB             5A        1C   AB  C3 006EF 84$:     SUBL3   28(CTX), CUR_PC, 16(CTX)          ; 3072
                                   C0   AD  D5 006F5          TSTL    DISP+28                                 ; 3078
                                        5C  19 006F8          BLSS    87$
                         50        54   8F  9A 006FA          MOVZBL  #84, R0                                 ; 3083
```

```
                                  F034  30 006FE          BSBW     ROOM                                    ;
                    8A              30  8E 00701          MNEGB    #48, (CUR_PC)+                          ; 3084
                    53              0B  D0 00704          MOVL     #11, R3                                 ; 3085
                    52        90    8F  9A 00707          MOVZBL   #144, R2                                ;
                                  EF14  30 0070B          BSBW     EMIT_DISP                               ;
                    8A        50    8F  90 0070E          MOVB     #80, (CUR_PC)+                          ; 3087
                    8A              13  B0 00712          MOVW     #19, (CUR_PC)+                          ; 3088
                    52              5A  D0 00715          MOVL     CUR_PC, TMP2                            ; 3089
                    20      0081    CB  91 00718          CMPB     129(CTX), #32                           ; 3094
                                    12  1B 0071D          BLEQU    85$                                     ;
                    8A 02AF00FB     8F  D0 0071F          MOVL     #45023483, (CUR_PC)+                    ; 3102
                    8A 003C0011     8F  D0 00726          MOVL     #3932177, (CUR_PC)+                     ;
                    56        FE    AA  9E 0072D          MOVAB    -2(R10), TMP                            ; 3103
                                    7E  7C 00731 85$:     CLRQ     -(SP)                                   ; 3112
                                    0A  DD 00733          PUSHL    #10                                     ;
                              C0    AD  DD 00735          PUSHL    DISP+28                                 ; 3113
                    7E      0081    CB  9A 00738          MOVZBL   129(CTX), -(SP)                         ; 3112
              F07E  CF              05  FB 0073D          CALLS    #5, GEN_MOVE                            ;
                    20      0081    CB  91 00742          CMPB     129(CTX), #32                           ; 3116
                                    08  1B 00747          BLEQU    86$                                     ;
                    8A              04  90 00749          MOVB     #4, (CUR_PC)+                           ; 3121
      FF   A6       5A              56  83 0074C          SUBB3    TMP, CUR_PC, -1(TMP)                    ; 3122
      FF   A2       5A              52  83 00751 86$:     SUBB3    TMP2, CUR_PC, -1(TMP2)                  ; 3131
                    50              0F  D0 00756 87$:     MOVL     #15, R0                                 ; 3140
                                  EFD9  30 00759          BSBW     ROOM                                    ;
                    51      0088    CB  9E 0075C          MOVAB    136(CTX), R1                            ; 3150
           03       01        58    AB  8F 00761          CASEB    88(CTX), #1, #3                         ; 3145
006C     0081     0043            0008     00766 88$:     .WORD    89$-88$,-                               ;
                                                                   92$-88$,-                               ;
                                                                   95$-88$,-                               ;
                                                                   93$-88$                                 ;
                    50        78    AB  9A 0076E 89$:     MOVZBL   120(CTX), R0                            ; 3150
      02   A1     0084  CB          50  A3 00772          SUBW3    R0, 132(CTX), 2(R1)                     ;
                    8A              3C  90 00779          MOVB     #60, (CUR_PC)+                          ; 3151
                              BC    AD  D5 0077C          TSTL     DISP+24                                 ; 3152
                                    0C  19 0077F          BLSS     90$                                     ;
                    53              0A  D0 00781          MOVL     #10, R3                                 ; 3154
                    52        BC    AD  D0 00784          MOVL     DISP+24, R2                             ;
                                  EE97  30 00788          BSBW     EMIT_DISP                               ;
                                    0A  11 0078B          BRB      91$                                     ;
                    53        02    A1  3C 0078D 90$:     MOVZWL   2(R1), R3                               ; 3156
                    52              02  D0 00791          MOVL     #2, R2                                  ;
                                  EEE1  30 00794          BSBW     EMIT_LITE                               ;
                    8A      9E50    8F  B0 00797 91$:     MOVW     #-25008, (CUR_PC)+                      ; 3157
                    52        78    AB  9A 0079C          MOVZBL   120(CTX), R2                            ; 3158
                    52        C4    AD  C0 007A0          ADDL2    DISP+32, R2                             ;
                    53              0A  D0 007A4          MOVL     #10, R3                                 ;
                                    6B  11 007A7          BRB      99$                                     ;
                02  A1      0084    CB  B0 007A9 92$:     MOVW     132(CTX), 2(R1)                         ; 3166
                    8A        9F    8F  90 007AF          MOVB     #-97, (CUR_PC)+                         ; 3167
                    53              0A  D0 007B3          MOVL     #10, R3                                 ; 3168
                    52        A4    AD  D0 007B6          MOVL     DISP, R2                                ;
                                  EE65  30 007BA          BSBW     EMIT_DISP                               ;
                    8A      01FB    8F  B0 007BD          MOVW     #507, (CUR_PC)+                         ; 3169
                    8A        9F    8F  90 007C2          MOVB     #-97, (CUR_PC)+                         ; 3170
                    8A 00000000G    00  9E 007C6          MOVAB    SOR$$RFA_ACCESS, (CUR_PC)+              ;
                    8A              05  90 007CD          MOVB     #5, (CUR_PC)+                           ; 3171
```

```
                                      4A  11 007D0         BRB     100$                                   ; 3145
              02    A1                06  B0 007D2 93$:    MOVW    #6, 2(R1)                              ; 3176
                              A8      AD  D5 007D6         TSTL    DISP+4                                 ; 3177
                                      03  19 007D9         BLSS    94$                                    ;
                              02      A1  B6 007DB         INCW    2(R1)                                  ; 3179
                    8A                30  8E 007DE 94$:    MNEGB   #48, (CUR_PC)+                         ; 3181
                    8A        02      A1  90 007E1         MOVB    2(R1), (CUR_PC)+                       ; 3183
                                      21  11 007E5         BRB     98$                                    ; 3185
                    50                04  D0 007E7 95$:    MOVL    #4, I                                  ; 3212
                    52        A4    AD40  D0 007EA 96$:    MOVL    DISP[I], Z                             ; 3213
                                      04  18 007EF         BGEQ    97$                                    ;
          F5        50                09  F3 007F1         AOBLEQ  #9, I, 96$                             ;
      02  A1        52        A4      AD  A3 007F5 97$:    SUBW3   DISP, Z, 2(R1)                         ; 3214
                    8A                3C  90 007FB         MOVB    #60, (CUR_PC)+                         ; 3215
                    53        02      A1  3C 007FE         MOVZWL  2(R1), R3                              ; 3216
                    52                02  D0 00802         MOVL    #2, R2                                 ;
                                    EE70  30 00805         BSBW    EMIT_LITE                              ;
                    8A      9E50      8F  B0 00808 98$:    MOVW    #-25008, (CUR_PC)+                     ; 3218
                    53                0A  D0 0080D         MOVL    #10, R3                                ; 3219
                    52        A4      AD  D0 00810         MOVL    DISP, R2                               ;
                                    EE0B  30 00814 99$:    BSBW    EMIT_DISP                              ;
                    8A      0551      8F  B0 00817         MOVW    #1361, (CUR_PC)+                       ; 3220
                    50                01  D0 0081C 100$:   MOVL    #1, R0                                 ; 3236
                                    EF13  30 0081F         BSBW    ROOM                                   ;
                    50        18      AB  9E 00822         MOVAB   24(CTX), R0                            ; 3241
          50        60        04      A0  C1 00826         ADDL3   4(R0), (R0), R0                        ;
                    50                5A  D1 0082B         CMPL    CUR_PC, R0                             ;
                                      0E  1B 0082E         BLEQU   103$                                   ;
                          001C1124    8F  DD 00830 101$:   PUSHL   #1839396                               ;
          00000000G 00                01  FB 00836 102$:   CALLS   #1, SOR$$ERROR                         ;
                                      04  00083D           RET                                            ;
              08    AB        1C      AB  C0 0083E 103$:   ADDL2   28(CTX), 8(CTX)                        ; 3244
                    6B        1C      AB  C0 00843         ADDL2   28(CTX), (CTX)                         ; 3245
              04    AB        1C      AB  C0 00847         ADDL2   28(CTX), 4(CTX)                        ; 3246
              10    AB        1C      AB  C0 0084C         ADDL2   28(CTX), 16(CTX)                       ; 3247
                              04      BB  95 00851         TSTB    @4(CTX)                                ; 3251
                                      03  12 00854         BNEQ    104$                                   ;
                              04      AB  D4 00856         CLRL    4(CTX)                                 ; 3253
            EC83    CF                00  FB 00859 104$:   CALLS   #0, DO_REI                             ; 3279
                    50                01  D0 0085E         MOVL    #1, R0                                 ; 3281
                                      04  00861            RET                                            ; 3283

; Routine Size:  2146 bytes,    Routine Base:  SOR$RO_CODE + 0C82
```

```
;  3246          3284  1 %IF NOT HOSTILE %THEN
;  3247          3285  1 EXTERNAL ROUTINE
;  3248          3286  1          LIB$FIND_IMAGE_SYMBOL: ADDRESSING_MODE(GENERAL);
;  3249          3287  1 BIND    DTYPE1 = UPLIT BYTE('SOR$DTYPE');
;  3250          3288  1 MACRO   DTYPE_DECL = VECTOR[2] INITIAL (%CHARCOUNT('SOR$DTYPE'), DTYPE1) %;
;  3251          3289  1 MACRO   NAMSTR_DECL(X) = VECTOR[2] INITIAL (%CHARCOUNT(X), UPLIT BYTE(X)) %;
;  3252        M 3290  1 MACRO   DTYPE_(X) =
;  3253        M 3291  1              BEGIN
;  3254        M 3292  1              OWN Z:      INITIAL(0);
;  3255        M 3293  1              BUILTIN     AP, CALLG;
;  3256        M 3294  1              IF .Z EQL 0
;  3257        M 3295  1              THEN
;  3258        M 3296  1                  BEGIN
;  3259        M 3297  1                  LOCAL DTYPE: DTYPE_DECL, NAMSTR: NAMSTR_DECL(X), STATUS;
;  3260        M 3298  1                  STATUS = LIB$FIND_IMAGE_SYMBOL(DTYPE, NAMSTR, Z);
;  3261        M 3299  1                  IF NOT .STATUS THEN RETURN SOR$$ERROR(SOR$_SHR_SYSERROR,0,.STATUS);
;  3262        M 3300  1                  END;
;  3263        M 3301  1              RETURN CALLG(.AP, .Z);
;  3264          3302  1              END %;
;  3265          3303  1 GLOBAL ROUTINE SOR$$DTYPE_KBF = DTYPE_('SOR$DTYPE_KBF');


                                                        .PSECT  SOR$RW_PICDATA,NOEXE,  PIC,2

                                        00000000  00000 Z:      .LONG   0                                               ;

                                                        .PSECT  SOR$RO_CODE,NOWRT,  SHR,  PIC,2

                45  50  59  54  44  24  52  4F  53  014E4 P.AAI:  .ASCII  \SOR$DTYPE\                                    ;
46  42  4B  5F 45  50  59  54  44  24  52  4F  53  014ED P.AAJ:  .ASCII  \SOR$DTYPE_KBF\                                ;

                                                DTYPE1=         P.AAI
                                                        .EXTRN  LIB$FIND_IMAGE_SYMBOL

                                          0004 00000            .ENTRY  SOR$$DTYPE_KBF, Save R2                         ; 3303
                                    52 00000000'  EF    9E 00002        MOVAB   Z, R2                                   ;
                                    5E            10    C2 00009        SUBL2   #16, SP                                 ;
                                                  62    D5 0000C        TSTL    Z                                       ;
                                                  35    12 0000E        BNEQ    1$                                      ;
                                 08 AE            09    D0 00010        MOVL    #9, DTYPE                               ;
                                 0C AE      D3    AF    9E 00014        MOVAB   DTYPE1, DTYPE+4                         ;
                                    6E            0D    D0 00019        MOVL    #13, NAMSTR                             ;
                                 04 AE      D4    AF    9E 0001C        MOVAB   P.AAJ, NAMSTR+4                         ;
                                                  52    DD 00021        PUSHL   R2                                      ;
                                          04      AE    9F 00023        PUSHAB  NAMSTR                                  ;
                                          10      AE    9F 00026        PUSHAB  DTYPE                                   ;
                          00000000G       00      03    FB 00029        CALLS   #3, LIB$FIND_IMAGE_SYMBOL               ;
                                    12            50    E8 00030        BLBS    STATUS, 1$                              ;
                                                  50    DD 00033        PUSHL   STATUS                                  ;
                                                  7E    D4 00035        CLRL    -(SP)                                   ;
                                    001C11B4      8F    DD 00037        PUSHL   #1839540                                ;
                          00000000G       00      03    FB 0003D        CALLS   #3, SOR$$ERROR                          ;
                                                  04    00044           RET                                             ;
                                 00 B2            6C    FA 00045 1$:    CALLG   (AP), @Z                                ;
                                                  04    00049           RET                                             ;
```

```
; Routine Size:  74 bytes,    Routine Base:  SOR$RO_CODE + 14FA

; 3266          3304  1 GLOBAL ROUTINE SOR$$DTYPE_T_W = DTYPE_('SOR$DTYPE_T_W');

                                                        .PSECT  SOR$RW_PICDATA,NOEXE,  PIC,2

                                            00000000  00004 Z:      .LONG   0                                       ;
                                                        .PSECT  SOR$RO_CODE,NOWRT,  SHR,  PIC,2
      57  5F  54  5F  45  50  59  54  44  24  52  4F  53  01544 P.AAK:  .ASCII  \SOR$DTYPE_T_W\                       ;

                                              0004  00000           .ENTRY  SOR$$DTYPE_T_W, Save R2                 ; 3304
                                  52 00000000'  EF   9E  00002      MOVAB   Z, R2
                                  5E            10   C2  00009      SUBL2   #16, SP
                                                62   D5  0000C      TSTL    Z
                                                36   12  0000E      BNEQ    1$
                              08  AE            09   D0  00010      MOVL    #9, DTYPE
                              0C  AE      FF7B  CF   9E  00014      MOVAB   DTYPE1, DTYPE+4
                                  6E            0D   D0  0001A      MOVL    #13, NAMSTR
                              04  AE        D3  AF   9E  0001D      MOVAB   P.AAK, NAMSTR+4
                                                52   DD  00022      PUSHL   R2
                                            04  AE   9F  00024      PUSHAB  NAMSTR
                                            10  AE   9F  00027      PUSHAB  DTYPE
                      00000000G   00            03   FB  0002A      CALLS   #3, LIB$FIND_IMAGE_SYMBOL
                                  12            50   E8  00031      BLBS    STATUS, 1$
                                                50   DD  00034      PUSHL   STATUS
                                                7E   D4  00036      CLRL    -(SP)
                                    001C11B4    8F   DD  00038      PUSHL   #1839540
                      00000000G   00            03   FB  0003E      CALLS   #3, SOR$$ERROR
                                                04   00045      RET
                            00    B2            6C   FA  00046 1$:  CALLG   (AP), @Z
                                                04   0004A      RET

; Routine Size:  75 bytes,    Routine Base:  SOR$RO_CODE + 1551

; 3267          3305  1 %FI
```

```
;  3269          3306  1 ROUTINE CLEAN_UP: CAL_CTXREG NOVALUE =
;  3270          3307  1
;  3271          3308  1 !++
;  3272          3309  1 !
;  3273          3310  1 ! FUNCTIONAL DESCRIPTION:
;  3274          3311  1 !
;  3275          3312  1 !       Release resources allocated by this module.
;  3276          3313  1 !
;  3277          3314  1 ! FORMAL PARAMETERS:
;  3278          3315  1 !
;  3279          3316  1 !       NONE
;  3280          3317  1 !
;  3281          3318  1 ! IMPLICIT INPUTS:
;  3282          3319  1 !
;  3283          3320  1 !       NONE
;  3284          3321  1 !
;  3285          3322  1 ! IMPLICIT OUTPUTS:
;  3286          3323  1 !
;  3287          3324  1 !       NONE
;  3288          3325  1 !
;  3289          3326  1 ! ROUTINE VALUE:
;  3290          3327  1 !
;  3291          3328  1 !       NONE (signals errors)
;  3292          3329  1 !
;  3293          3330  1 ! SIDE EFFECTS:
;  3294          3331  1 !
;  3295          3332  1 !       NONE
;  3296          3333  1 !
;  3297          3334  1 !--
;  3298          3335  2     BEGIN
;  3299          3336  2     EXTERNAL REGISTER
;  3300          3337  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK;
;  3301          3338  2
;  3302          3339  2     ! Deallocate the code we generated
;  3303          3340  2     !
;  3304          3341  2     SOR$$DEALLOCATE(.VECTOR[CTX[COM_ROUTINES],0], VECTOR[CTX[COM_ROUTINES],1]);
;  3305          3342  2
;  3306          3343  1     END;


                                        0000 00000 CLEAN_UP:
                                                   .WORD   Save nothing                    ; 3306
                                1C  AB  9F 00002           PUSHAB  28(CTX)                 ; 3341
                           50           5B  D0 00005       MOVL    CTX, R0                 ;
                                18  A0  DD 00008           PUSHL   24(R0)                  ;
                      00000000G  00     02  FB 0000B       CALLS   #2, SOR$$DEALLOCATE     ;
                                        04 00012           RET                             ; 3343

; Routine Size:  19 bytes,    Routine Base:  SOR$RO_CODE + 159C

;  3307          3344  1
;  3308          3345  1 END
;  3309          3346  0 ELUDOM
```

```
;                                  PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  SOR$RO_CODE------------------2       4  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  SOR$RO_CODE                       5551  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  SOR$RW_PICDATA                       8  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                               Library Statistics

;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time

;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        53         0        1000        00:01.8
;  _$255$DUA28:[SORT32.SRC]OPCODES.L32;1    343        77        22          18        00:00.6
;  _$255$DUA28:[SORT32.SRC]SORLIB.L32;1     409       163        39          34        00:00.6


;                                   COMMAND QUALIFIERS

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

; Size:          5396 code + 167 data bytes
; Run Time:         02:03.7
; Elapsed Time:     06:18.8
; Lines/CPU Min:     1623
; Lexemes/CPU-Min: 29315
; Memory Used:  788 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SORINTERF
LIS

SORKEYSUB
LIS

SORFILNAM
LIS

SORCOMPAR
LIS

SORENTRY
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SORRMSIO
LIS

SORMSG
LIS

SORSCRIO
LIS

SOROUTPUT
LIS

SORLIB
LIS