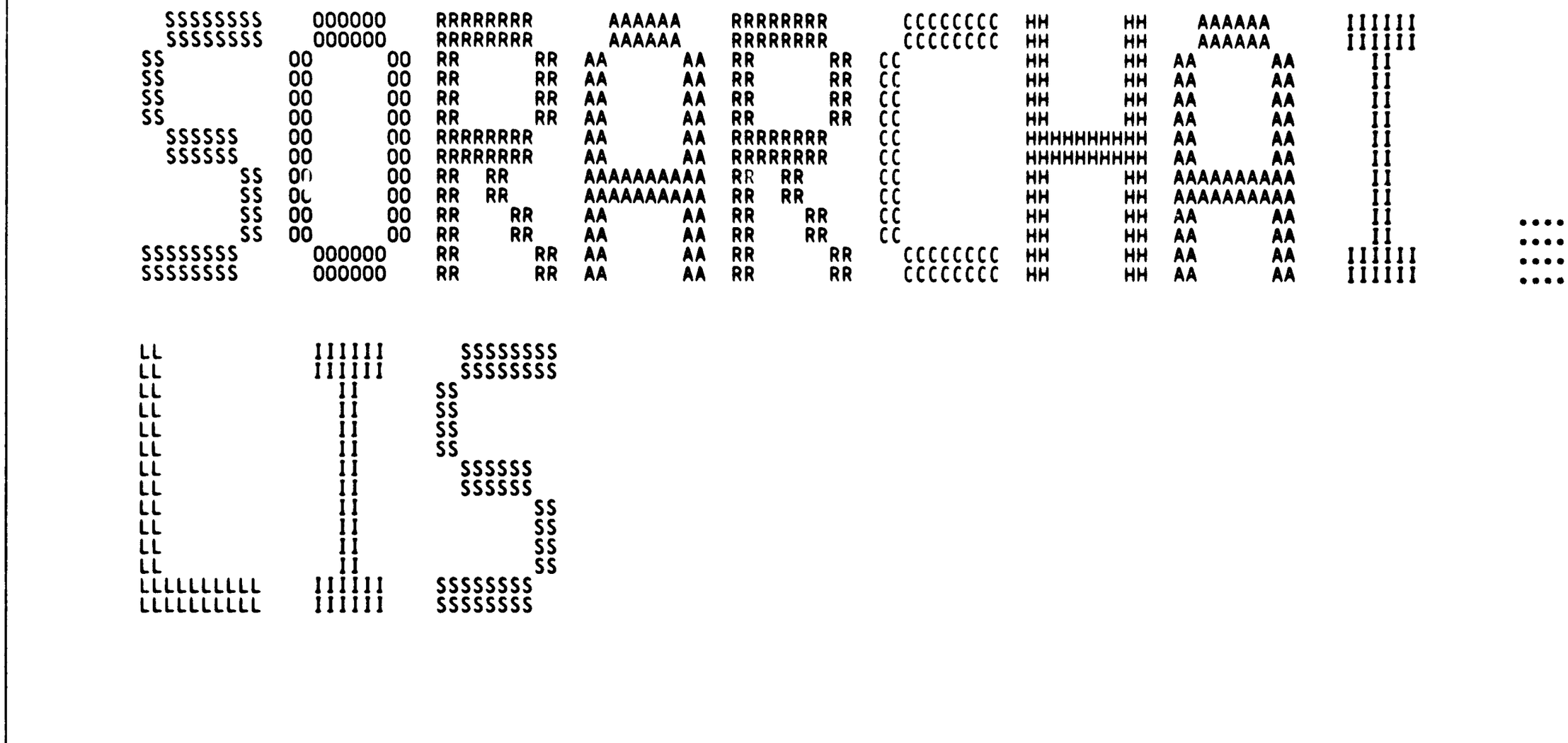
```
   SSSSSSSSSSS     000000000     RRRRRRRRRRR     TTTTTTTTTTTTTTT     333333333       222222222
   SSSSSSSSSSS     000000000     RRRRRRRRRRRR    TTTTTTTTTTTTTTT     333333333       222222222
   SSSSSSSSSSS     000000000     RRRRRRRRRRRR    TTTTTTTTTTTTTTT     333333333       222222222
SSS              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
```


SSSSSSSS 000000 RRRRRRRR AAAAAA RRRRRRRR CCCCCCCC HH HH AAAAAA IIIIII
SSSSSSSS 000000 RRRRRRRR AAAAAA RRRRRRRR CCCCCCCC HH HH AAAAAA IIIIII
SS OO OO RR RR AA AA RR RR CC HH HH AA AA II
SS OO OO RR RR AA AA RR RR CC HH HH AA AA II
SS OO OO RR RR AA AA RR RR CC HH HH AA AA II
SS OO OO RR RR AA AA RR RR CC HH HH AA AA II
SSSSSS OO OO RRRRRRRR AA AA RRRRRRRR CC HHHHHHHHHH AA AA II
SSSSSS OO OO RRRRRRRR AA AA RRRRRRRR CC HHHHHHHHHH AA AA II
SS OO OO RR RR AAAAAAAAAA RR RR CC HH HH AAAAAAAAAA II
SS OO OO RR RR AAAAAAAAAA RR RR CC HH HH AAAAAAAAAA II
SS OO OO RR RR AA AA RR RR CC HH HH AA AA II
SS OO OO RR RR AA AA RR RR CC HH HH AA AA II
SSSSSSSS 000000 RR RR AA AA RR RR CCCCCCCC HH HH AA AA IIIIII
SSSSSSSS 000000 RR RR AA AA RR RR CCCCCCCC HH HH AA AA IIIIII
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$ARCHAIC (
;     2          0002  0                 IDENT = 'V04-000'       ! File: SORARCHAI.B32 Edit: PDG3019
;     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 archaic and atrophying features of Sort/Merge.
;    37          0037  1 !
;    38          0038  1 !       This module makes use of the following non-user-visible aspects of
;    39          0039  1 !       VAX-11 Sort/Merge:
;    40          0040  1 !               SOR$PASS_FILES returns the address of the context area in R1.
;    41          0041  1 !               Within this area, it accesses COM_TKS, COM_HACK_STRIP and
;    42          0042  1 !               COM_HACK_2ARGS.
;    43          0043  1 !       Thus, if these are relocated within the context area, this module
;    44          0044  1 !       must be recompiled (and user programs that use it must be relinked).
;    45          0045  1 !
;    46          0046  1 !       The archaic global literals are defined as weak literals.  A user who
;    47          0047  1 !       tries to use them without referencing SOR$INIT_SORT or SOR$INIT_MERGE
;    48          0048  1 !       (i.e., he's using the new sort), he'll get a link-time error.
;    49          0049  1 !
;    50          0050  1 ! ENVIRONMENT:  VAX/VMS user mode
;    51          0051  1 !
;    52          0052  1 ! AUTHOR: Peter D Gilbert, CREATION DATE: 25-Jun-1982
;    53          0053  1 !
;    54          0054  1 ! MODIFIED BY:
;    55          0055  1 !
;    56          0056  1 !       T03-015         Original
;    57          0057  1 !       T03-016 Corrected the order of USER_EQUAL and USER_COMPARE parameters
```

```
;   58          0058  1 !                   in call to SOR$BEGIN_MERGE.  PDG 9-Dec-1982
;   59          0059  1 !           T03-017 Do not pass the SIGNAL option.  PDG 29-Dec-1982
;   60          0060  1 !           T03-018 Improve index check before referencing DSC_BIN.  PDG 2-Feb-1983
;   61          0061  1 !           T03-019 Make NOSIGNAL a default option.  PDG 11-Apr-1983
;   62          0062  1 !--
```

```
;    64          0063  1 LIBRARY 'SYS$LIBRARY:STARLET';
;    65          0064  1 REQUIRE 'SRC$:COM';
;    66          0134  1
;    67          0135  1
;    68          0136  1 FORWARD ROUTINE
;    69          0137  1     DSC_DTYPE,                  ! Convert to DSC datatypes
;    70          0138  1     SOR$INIT_SORT,              ! Initialize the sort
;    71          0139  1     SOR$INIT_MERGE,             ! Initialize the merge
;    72          0140  1     SOR$DO_MERGE;               ! "Perform the merge"
;    73          0141  1
;    74          0142  1 LINKAGE
;    75          0143  1     AND_RETURN_R1 = CALL(;REGISTER=1);
;    76          0144  1
;    77          0145  1 EXTERNAL ROUTINE
;    78          0146  1     SOR$PASS_FILES:     ADDRESSING_MODE(GENERAL) AND_RETURN_R1,
;    79          0147  1     SOR$BEGIN_SORT:     ADDRESSING_MODE(GENERAL),       ! Initialize the sort
;    80          0148  1     SOR$BEGIN_MERGE:    ADDRESSING_MODE(GENERAL),       ! Initialize the merge
;    81          0149  1     SOR$END_SORT:       ADDRESSING_MODE(GENERAL);       ! Finish the sort/merge
```

```
;    83          0150  1 ! Define the archaic global literals
;    84          0151  1 !
;    85          0152  1 ! These are defined as weak literals.  Thus, if the user tries to use them
;    86          0153  1 ! without referencing SOR$INIT_SORT or SOR$INIT_MERGE (i.e., he's using the
;    87          0154  1 ! new sort), he'll get an link-time error.
;    88          0155  1 !
;    89          0156  1 GLOBAL LITERAL
;    90          0157  1     SOR$GK_CHAR_KEY =   KEY_K_CHAR:     WEAK,
;    91          0158  1     SOR$GK_BIN_KEY =    KEY_K_BIN:      WEAK,
;    92          0159  1     SOR$GK_ZONE_KEY =   KEY_K_ZONE:     WEAK,
;    93          0160  1     SOR$GK_PACK_KEY =   KEY_K_PACK:     WEAK,
;    94          0161  1     SOR$GK_USB_KEY =    KEY_K_USB:      WEAK,
;    95          0162  1     SOR$GK_DLO_KEY =    KEY_K_DLO:      WEAK,
;    96          0163  1     SOR$GK_DLS_KEY =    KEY_K_DLS:      WEAK,
;    97          0164  1     SOR$GK_DTO_KEY =    KEY_K_DTO:      WEAK,
;    98          0165  1     SOR$GK_DTS_KEY =    KEY_K_DTS:      WEAK,
;    99          0166  1     SOR$GK_FLT_KEY =    KEY_K_FLT:      WEAK,
;   100          0167  1     SOR$GK_FLTD_KEY =   KEY_K_FLTD:     WEAK,
;   101          0168  1     SOR$GK_FLTG_KEY =   KEY_K_FLTG:     WEAK,
;   102          0169  1     SOR$GK_FLTH_KEY =   KEY_K_FLTH:     WEAK;
```

```
;  104          0170  1 ! Macros to test for optional parameters.
;  105          0171  1 !
;  106          0172  1 !   FIRSTPARAMETER_
;  107          0173  1 !       Define the first parameter, for use by PRESENT_ and NULL_.
;  108          0174  1 !
;  109          0175  1 !   PRESENT_
;  110          0176  1 !       Test for a parameter present.
;  111          0177  1 !
;  112          0178  1 !   NULL_
;  113          0179  1 !       Test for a parameter present, and whether it equals zero.
;  114          0180  1 !
;  115          0181  1 MACRO
;  116        M 0182  1     PRESENT_(X) =
;  117        M 0183  1         BEGIN
;  118        M 0184  1         BUILTIN ACTUALCOUNT;
;  119        M 0185  1         LITERAL Y__ = 1+(X-FIRSTPARAMETER__)/%UPVAL;
;  120        M 0186  1         ACTUALCOUNT() GEQU Y__
;  121          0187  1         END %,
;  122        M 0188  1     NULL_(X) =
;  123        M 0189  1         BEGIN
;  124        M 0190  1         BUILTIN NULLPARAMETER;
;  125        M 0191  1         LITERAL Y__ = 1+(X-FIRSTPARAMETER__)/%UPVAL;
;  126        M 0192  1         NULLPARAMETER(Y__)
;  127          0193  1         END %,
;  128        M 0194  1     FIRSTPARAMETER_(X) =
;  129          0195  1         MACRO FIRSTPARAMETER__ = X %QUOTE % %;
```

```
;  131          0196  1 ROUTINE DSC_DTYPE
;  132          0197  1         (
;  133          0198  1         KEY_BUF1:       REF KEY_BLOCK,
;  134          0199  1         KEY_BUF2:       REF KEY_BLOCK
;  135          0200  1         ) =
;  136          0201  1
;  137          0202  1 !++
;  138          0203  1 ! Functional Description:
;  139          0204  1 !
;  140          0205  1 !       This routine converts old format key descriptions to DSC format
;  141          0206  1 !       key descriptions.
;  142          0207  1 !
;  143          0208  1 ! Formal Parameters:
;  144          0209  1 !
;  145          0210  1 !   KEY_BUF1    Address of old format key descriptions.
;  146          0211  1 !   KEY_BUF2    Address of DSC format key descriptions.
;  147          0212  1 !
;  148          0213  1 ! Implicit Inputs:
;  149          0214  1 !
;  150          0215  1 !       None.
;  151          0216  1 !
;  152          0217  1 ! Implicit Outputs:
;  153          0218  1 !
;  154          0219  1 !       None.
;  155          0220  1 !
;  156          0221  1 ! Routine Value:
;  157          0222  1 !
;  158          0223  1 !       None (may signal errors).
;  159          0224  1 !
;  160          0225  1 ! Side Effects:
;  161          0226  1 !
;  162          0227  1 !       None.
;  163          0228  1 !
;  164          0229  1 !--
;  165          0230  1
;  166          0231  2     BEGIN
;  167          0232  2     LOCAL
;  168          0233  2         KBF1:   REF KBF_BLOCK,
;  169          0234  2         KBF2:   REF KBF_BLOCK;
;  170          0235  2     LITERAL
;  171          0236  2         K_MAXDEC =      31;     ! Maximum length of decimal data
;  172          0237  2     OWN
;  173          0238  2         DSC_DTYPES: VECTOR[KEY_K_MAX+1,BYTE]
;  174          0239  2             PSECT(SOR$RO_CODE) PRESET(
;  175          0240  2                 [KEY_K_CHAR]=   DSC$K_DTYPE_T,
;  176          0241  2                 [KEY_K_BIN]=    0,
;  177          0242  2                 [KEY_K_ZONE]=   DSC$K_DTYPE_NZ,
;  178          0243  2                 [KEY_K_PACK]=   DSC$K_DTYPE_P,
;  179          0244  2                 [KEY_K_USB]=    0,
;  180          0245  2                 [KEY_K_DLO]=    DSC$K_DTYPE_NLO,
;  181          0246  2                 [KEY_K_DLS]=    DSC$K_DTYPE_NL,
;  182          0247  2                 [KEY_K_DTO]=    DSC$K_DTYPE_NRO,
;  183          0248  2                 [KEY_K_DTS]=    DSC$K_DTYPE_NR,
;  184          0249  2                 [KEY_K_FLT]=    DSC$K_DTYPE_F,
;  185          0250  2                 [KEY_K_FLTD]=   DSC$K_DTYPE_D,
;  186          0251  2                 [KEY_K_FLTG]=   DSC$K_DTYPE_G,
;  187          0252  2                 [KEY_K_FLTH]=   DSC$K_DTYPE_H),
```

```
;  188          0253  2          DSC_LENGTH: VECTOR[KEY_K_MAX+1]
;  189          0254  2              PSECT(SOR$RO_CODE) PRESET(
;  190          0255  2                  [KEY_K_CHAR]=   0,
;  191          0256  2                  [KEY_K_BIN]=    1^1+1^2+1^4+1^8+1^16,
;  192          0257  2                  [KEY_K_ZONE]=   -1,
;  193          0258  2                  [KEY_K_PACK]=   -1,
;  194          0259  2                  [KEY_K_USB]=    1^1+1^2+1^4+1^8+1^16,
;  195          0260  2                  [KEY_K_DLO]=    -1,
;  196          0261  2                  [KEY_K_DLS]=    -1,
;  197          0262  2                  [KEY_K_DTO]=    -1,
;  198          0263  2                  [KEY_K_DTS]=    -1,
;  199          0264  2                  [KEY_K_FLT]=    1^0+1^4,
;  200          0265  2                  [KEY_K_FLTD]=   1^0+1^8,
;  201          0266  2                  [KEY_K_FLTG]=   1^0+1^8,
;  202          0267  2                  [KEY_K_FLTH]=   1^0+1^16),
;  203          0268  2          DSC_BIN: VECTOR[5,BYTE]
;  204          0269  2              PSECT(SOR$RO_CODE) PRESET(
;  205          0270  2                  [0]=    DSC$K_DTYPE_B,
;  206          0271  2                  [1]=    DSC$K_DTYPE_W,
;  207          0272  2                  [2]=    DSC$K_DTYPE_L,
;  208          0273  2                  [3]=    DSC$K_DTYPE_Q,
;  209          0274  2                  [4]=    DSC$K_DTYPE_O),
;  210          0275  2          DSC_USB: VECTOR[5,BYTE]
;  211          0276  2              PSECT(SOR$RO_CODE) PRESET(
;  212          0277  2                  [0]=    DSC$K_DTYPE_BU,
;  213          0278  2                  [1]=    DSC$K_DTYPE_WU,
;  214          0279  2                  [2]=    DSC$K_DTYPE_LU,
;  215          0280  2                  [3]=    DSC$K_DTYPE_QU,
;  216          0281  2                  [4]=    DSC$K_DTYPE_OU);
;  217          0282  2
;  218          0283  2      BUILTIN
;  219          0284  2          FFS;
;  220          0285  2
;  221          0286  2      KEY_BUF2[KEY_NUMBER] = .KEY_BUF1[KEY_NUMBER];
;  222          0287  2      IF .KEY_BUF2[KEY_NUMBER] GTRU MAX_KEYS THEN RETURN SOR$_BAD_KEY;
;  223          0288  2      DECR I FROM .KEY_BUF1[KEY_NUMBER]-1 TO 0 DO
;  224          0289  3          BEGIN
;  225          0290  3          KBF1 = KEY_BUF1[KEY_KBF(.I)];
;  226          0291  3          KBF2 = KEY_BUF2[KEY_KBF(.I)];
;  227          0292  3          KBF2[KBF_ORDER] = .KBF1[KBF_ORDER];                     ! Get order
;  228          0293  3          IF .KBF2[KBF_ORDER] GTRU 1 THEN RETURN SOR$_BAD_KEY;    ! Check order
;  229          0294  3          IF .KBF1[KBF_TYPE] NEQ KEY_K_CHAR
;  230          0295  3          THEN                                                    ! Check length
;  231          0296  4              BEGIN
;  232          0297  4              IF .KBF1[KBF_LENGTH] GTRU K_MAXDEC THEN RETURN SOR$_BAD_KEY;
;  233          0298  4              IF NOT .(DSC_LENGTH[.KBF1[KBF_TYPE]])<.KBF1[KBF_LENGTH],1,0>
;  234          0299  4              THEN
;  235          0300  4                  RETURN SOR$_BAD_KEY;
;  236          0301  3              END;
;  237          0302  3          KBF2[KBF_LENGTH] = .KBF1[KBF_LENGTH];                   ! Get length
;  238          0303  3          IF .KBF1[KBF_POSITION] EQL 0 THEN RETURN SOR$_BAD_KEY;
;  239          0304  3          KBF2[KBF_POSITION] = .KBF1[KBF_POSITION] - 1;           ! Get position
;  240          0305  3          KBF2[KBF_TYPE] = .DSC_DTYPES[.KBF1[KBF_TYPE]];          ! Get type
;  241          0306  3
;  242          0307  3
;  243          0308  3          ! For binary datatypes, compute the DSC datatype based on the length.
;  244          0309  3          !
```

```
;  245          0310  3          IF .KBF2[KBF_TYPE] EQL 0
;  246          0311  3          THEN
;  247          0312  4              BEGIN
;  248          0313  4              !
;  249          0314  4              ! KEY_K_BIN or KEY_X_USB
;  250          0315  4              !
;  251          0316  4              LOCAL
;  252          0317  4                  BITNUM;
;  253          0318  4              FFS(%REF(0), %REF(%FIELDEXPAND(KBF_LENGTH,2)),
;  254          0319  4                  KBF1[KBF_LENGTH], BITNUM);
;  255          0320  4              IF (.BITNUM GEQ %ALLOCATION(DSC_BIN)) OR
;  256          0321  5                  (.KBF1[KBF_LENGTH] NEQ 1^.BITNUM)
;  257          0322  4              THEN
;  258          0323  4                  RETURN SOR$_BAD_KEY;
;  259          0324  4              IF .KBF1[KBF_TYPE] EQL KEY_K_BIN
;  260          0325  4                  THEN KBF2[KBF_TYPE] = .DSC_BIN[.BITNUM]
;  261          0326  4                  ELSE KBF2[KBF_TYPE] = .DSC_USB[.BITNUM];
;  262          0327  3              END;
;  263          0328  3
;  264          0329  3
;  265          0330  3          ! Adjust the length for NR and NL datatypes;
;  266          0331  3          ! For floating datatypes, compute the length based on the datatype.
;  267          0332  3          !
;  268          0333  3          IF
;  269        P 0334  3              ONEOF_(.KBF2[KBF_TYPE], BMSK_(
;  270          0335  4                  DSC$K_DTYPE_NR,DSC$K_DTYPE_NL))
;  271          0336  3          THEN
;  272          0337  3              KBF2[KBF_LENGTH] = .KBF2[KBF_LENGTH] + 1
;  273          0338  3          ELIF
;  274        P 0339  3              ONEOF_(.KBF2[KBF_TYPE], BMSK_(
;  275          0340  4                  DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H))
;  276          0341  3          THEN
;  277          0342  4              BEGIN
;  278          0343  4              LOCAL
;  279          0344  4                  BITNUM;
;  280          0345  4              FFS(%REF(1), %REF(%BPVAL-1),
;  281          0346  4                  DSC_LENGTH[.KBF1[KBF_TYPE]], BITNUM);
;  282          0347  4              KBF2[KBF_LENGTH] = .BITNUM;
;  283          0348  3              END;
;  284          0349  3
;  285          0350  2              END;
;  286          0351  2
;  287          0352  2          RETURN SS$_NORMAL;
;  288          0353  2
;  289          0354  1          END;


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

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

                                                    00  00000 DSC_DTYPES:
                                                                .BYTE   0
        1C  1B  0B  0A  12  13  10  11  00  15  14  00  0E  00001          .BYTE   14, 0, 20, 21, 0, 17, 16, 19, 18, 10, 11, -
                                                                27, 28
                                                        0000E           .BLKB   2
```

```
                                                        00# 00010 DSC_LENGTH:
                                                                   .BYTE   0[4]
FFFFFFFF  00010116  FFFFFFFF  FFFFFFFF  00010116  00000000  00014          .LONG   0, 65814, -1, -1, 65814, -1, -1, -1, -1, -
00000101  00000101  00000011  FFFFFFFF  FFFFFFFF  FFFFFFFF  0002C                  17, 257, 257, 65537
                                                  00010001  00044
                                                  1A  09  08  07  06  00048 DSC_BIN:.BYTE   6, 7, 8, 9, 26
                                                                      0004D        .BLKB   3
                                                  19  05  04  03  02  00050 DSC_USB:.BYTE   2, 3, 4, 5, 25

                                                                   SOR$GK_CHAR_KEY==   1
                                                                   SOR$GK_BIN_KEY==    2
                                                                   SOR$GK_ZONE_KEY==   3
                                                                   SOR$GK_PACK_KEY==   4
                                                                   SOR$GK_USB_KEY==    5
                                                                   SOR$GK_DLO_KEY==    6
                                                                   SOR$GK_DLS_KEY==    7
                                                                   SOR$GK_DTO_KEY==    8
                                                                   SOR$GK_DTS_KEY==    9
                                                                   SOR$GK_FLT_KEY==    10
                                                                   SOR$GK_FLTD_KEY==   11
                                                                   SOR$GK_FLTG_KEY==   12
                                                                   SOR$GK_FLTH_KEY==   13
                                                                          .EXTRN  SOR$PASS_FILES, SOR$BEGIN_SORT
                                                                          .EXTRN  SOR$BEGIN_MERGE
                                                                          .EXTRN  SOR$END_SORT

                                                              007C 00000 DSC_DTYPE:
                                                                          .WORD   Save R2,R3,R4,R5,R6                     ; 0196
                                                        56        B6    AF  9E 00002          MOVAB   DSC_LENGTH, R6
                                                     08 BC        04    BC  B0 00006          MOVW    @KEY_BUF1, @KEY_BUF2      ; 0286
                                                  00FF 8F         08    BC  B1 0000B          CMPW    @KEY_BUF2, #255           ; 0287
                                                                    69  1A 00011              BGTRU   3$
                                                        54        04    BC  3C 00013          MOVZWL  @KEY_BUF1, I              ; 0288
                                                                  00A0  31 00017              BRW     8$
                                                        52        04 BC44  7E 0001A 1$:       MOVAQ   @KEY_BUF1[I], KBF1        ; 0290
                                                        52           02  C0 0001F           ADDL2   #2, KBF1
                                                        51        08 BC44  7E 00022          MOVAQ   @KEY_BUF2[I], KBF2        ; 0291
                                                        51           02  C0 00027           ADDL2   #2, KBF2
                                                     02 A1        02    A2  B0 0002A          MOVW    2(KBF1), 2(KBF2)          ; 0292
                                                        01        02    A1  B1 0002F          CMPW    2(KBF2), #1               ; 0293
                                                                    47  1A 00033              BGTRU   3$
                                                        53           62  3C 00035           MOVZWL  (KBF1), R3                ; 0294
                                                        01           53  B1 00038           CMPW    R3, #1
                                                                     11  13 0003B           BEQL    2$
                                                        1F        06    A2  B1 0003D          CMPW    6(KBF1), #31              ; 0297
                                                                    39  1A 00041              BGTRU   3$
                                                        50        06    A2  3C 00043          MOVZWL  6(KBF1), R0               ; 0298
                                                                  6643  DF 00047              PUSHAL  DSC_LENGTH[R3]
                                                  2E              9E    50  E1 0004A          BBC     R0, @(SP)+, 3$
                                                     06 A1        06    A2  B0 0004E 2$:       MOVW    6(KBF1), 6(KBF2)          ; 0302
                                                                  04    A2  B5 00053          TSTW    4(KBF1)                   ; 0303
                                                                    24  13 00056              BEQL    3$
                                                  04 A1        04    A2  A3 00058           SUBW3   #1, 4(KBF1), 4(KBF2)      ; 0304
                                                        61        F0 A643  9B 0005E          MOVZBW  DSC_DTYPES[R3], (KBF2)    ; 0305
                                                                    30  12 00063              BNEQ    6$                        ; 0310
50                                                  10              00  EA 00065          FFS     #0, #16, 6(KBF1), BITNUM  ; 0319
                                                  06 A2           05    50  D1 0006B          CMPL    BITNUM, #5                ; 0320
```

```
                                              0C  18 0006E          BGEQ    3$                                    :
                      55          01          50  78 00070          ASHL    BITNUM, #1, R5                        : 0321
        55      06    A2          10          00  ED 00074          CMPZV   #0, #16, 6(KBF1), R5                  :
                                              08  13 0007A          BEQL    4$                                    :
                          50 001C8034         8F  D0 0007C 3$:      MOVL    #1867828, R0                          : 0323
                                              04     00083          RET                                           :
                                  02          53  B1 00084 4$:      CMPW    R3, #2                                : 0324
                                              07  12 00087          BNEQ    5$                                    :
                                  61     38 A640  9B 00089          MOVZBW  DSC_BIN[BITNUM], (KBF2)               : 0325
                                              05  11 0008E          BRB     6$                                    :
                                  61     40 A640  9B 00090 5$:      MOVZBW  DSC_USB[BITNUM], (KBF2)               : 0326
                      50 0000A000         8F  61  78 00095 6$:      ASHL    (KBF2), #40960, R0                    : 0335
                                              05  18 0009D          BGEQ    7$                                    :
                                         06   A1  B6 0009F          INCW    6(KBF2)                               : 0337
                                              16  11 000A2          BRB     8$                                    :
                      50 00300018         8F  61  78 000A4 7$:      ASHL    (KBF2), #3145752, R0                  : 0340
                                              0C  18 000AC          BGEQ    8$                                    :
                                            6643  DF 000AE          PUSHAL  DSC_LENGTH[R3]                        : 0346
        50            9E          1F          01  EA 000B1          FFC     #1, #31, @(SP)+, BITNUM               :
                             06   A1          50  B0 000B6          MOVW    BITNUM, 6(KBF2)                       : 0347
                                  02          54  F4 000BA 8$:      SOBGEQ  I, 9$                                 : 0288
                                              03  11 000BD          BRB     10$                                   :
                                            FF58  31 000BF 9$:      BRW     1$                                    :
                                  50          01  D0 000C2 10$:     MOVL    #1, R0                                : 0352
                                              04     000C5          RET                                           : 0354

; Routine Size:  198 bytes,    Routine Base:  SOR$RO_CODE + 0055
```

```
;  291          0355  1 GLOBAL ROUTINE SOR$INIT_SORT
;  292          0356  1         (
;  293          0357  1         KEY_BUFFER:     REF VECTOR[1,WORD],
;  294          0358  1         LRL:            REF VECTOR[1,WORD],
;  295          0359  1         FILE_ALLOC:     REF VECTOR[1,LONG],
;  296          0360  1         WORK_FILES:     REF VECTOR[1,BYTE],
;  297          0361  1         SORT_TYPE:      REF VECTOR[1,BYTE],
;  298          0362  1         TOT_KEY_SIZE:   REF VECTOR[1,BYTE],
;  299          0363  1         USER_COMPARE,
;  300          0364  1         OPTIONS:        REF BLOCK[1],
;  301          0365  1         EXTRA
;  302          0366  1         ) =
;  303          0367  1 !++
;  304          0368  1 !
;  305          0369  1 ! FUNCTIONAL DESCRIPTION:
;  306          0370  1 !
;  307          0371  1 !       This routine sets the COM_HACK_TKB bit, converts keys to DSC format,
;  308          0372  1 !       and calls SOR$BEGIN_SORT to initialize the sort.
;  309          0373  1 !
;  310          0374  1 ! FORMAL PARAMETERS:
;  311          0375  1 !
;  312          0376  1 !       KEY_BUFFER.raw          Key buffer address
;  313          0377  1 !       LRL.rwu.r               Longest record length
;  314          0378  1 !       FILE_ALLOC.rlu.r        Input file allocation
;  315          0379  1 !       WORK_FILES.rbu.r        Number of work files
;  316          0380  1 !       SORT_TYPE.rbu.r         Type of sort (record/tag/index/address)
;  317          0381  1 !       TOT_KEY_SIZE.rbu.r      Total key size
;  318          0382  1 !       USER_COMPARE.rzem.r     User-written comparison routine
;  319          0383  1 !       OPTIONS.rlu.r           Option bits
;  320          0384  1 !
;  321          0385  1 !       All parameters are optional.
;  322          0386  1 !
;  323          0387  1 ! IMPLICIT INPUTS:
;  324          0388  1 !
;  325          0389  1 !       NONE
;  326          0390  1 !
;  327          0391  1 ! IMPLICIT OUTPUTS:
;  328          0392  1 !
;  329          0393  1 !       NONE
;  330          0394  1 !
;  331          0395  1 ! ROUTINE VALUE:
;  332          0396  1 !
;  333          0397  1 !       Status code.
;  334          0398  1 !
;  335          0399  1 ! SIDE EFFECTS:
;  336          0400  1 !
;  337          0401  1 !       The working set is extended and the virtual memory is allocated.
;  338          0402  1 !
;  339          0403  1 !--
;  340          0404  2     BEGIN
;  341          0405  2     FIRSTPARAMETER_(KEY_BUFFER);        ! Required by PRESENT_ and NULL_ macros
;  342          0406  2
;  343          0407  2     LITERAL
;  344          0408  2         USED_OPTIONS =
;  345          0409  2             MASK_(OPT_EBCDIC, OPT_STABLE),
;  346          0410  2         DEF_OPTIONS =
;  347          0411  2             MASK_(OPT_NOSIGNAL);
```

```
;  348          0412  2
;  349          0413  2      LOCAL
;  350          0414  2          KEYS:   KEY_BLOCK,              ! DSC format keys
;  351          0415  2          CTX:    REF CTX_BLOCK,          ! Addr of context area
;  352          0416  2          KEY_PARAM: REF KEY_BLOCK,
;  353          0417  2          STATUS;
;  354          0418  2
;  355          0419  2      MACRO
;  356          0420  2          PARAM_(A) = (IF PRESENT_(A) THEN .A ELSE 0) %;
;  357          0421  2
;  358          0422  2
;  359          0423  2      ! If the user wants concurrent sorts, he must use the new interface to
;  360          0424  2      ! get them.
;  361          0425  2      !
;  362          0426  2      IF NOT NULL_(EXTRA) THEN RETURN SOR$_UNDOPTION;
;  363          0427  2
;  364          0428  2
;  365          0429  2      ! Get the context area
;  366          0430  2      !
;  367          0431  2      ! We know that SOR$PASS_FILES will allocate the context area,
;  368          0432  2      ! and that SOR$PASS_FILES may be called with no parameters.
;  369          0433  2      ! Finally, as a hack, SOR$PASS_FILES returns the address of the context
;  370          0434  2      ! area in R1.
;  371          0435  2      !
;  372          0436  2      STATUS = SOR$PASS_FILES(;CTX);
;  373          0437  2      IF NOT .STATUS THEN RETURN .STATUS;
;  374          0438  2
;  375          0439  2
;  376          0440  2      ! Check the options specified
;  377          0441  2      !
;  378          0442  3      IF NOT NULL_(OPTIONS)
;  379          0443  2      THEN
;  380          0444  2          IF (.OPTIONS[0,L_] AND NOT USED_OPTIONS) NEQ 0
;  381          0445  2          THEN
;  382          0446  2              RETURN SOR$_UNDOPTION;              ! Invalid options specified
;  383          0447  2
;  384          0448  2
;  385          0449  2      ! Get the total key size, for what it's worth.
;  386          0450  2      ! Note that this may conflict with information from the key buffer,
;  387          0451  2      ! or the specification file.
;  388          0452  2      !
;  389          0453  2      ! TKS stuff is damned stupid.
;  390          0454  2      !
;  391          0455  2      IF NOT NULL_(TOT_KEY_SIZE) THEN CTX[COM_TKS] = .TOT_KEY_SIZE[0];
;  392          0456  2      CTX[COM_HACK_STRIP] = TRUE;          ! Set this by default
;  393          0457  2
;  394          0458  2
;  395          0459  2      ! Set the bit indicating only 2 parameters are passed to callback routines
;  396          0460  2      !
;  397          0461  2      CTX[COM_HACK_2ARGS] = TRUE;
;  398          0462  2
;  399          0463  2
;  400          0464  2      ! Convert keys to the new format
;  401          0465  2      ! Note: "If you pass both key buffer address and comparison address,
;  402          0466  2      ! SORT ignores the comparison routine address".
;  403          0467  2      !
;  404          0468  2      KEY_PARAM = 0;
```

```
;  405          0469  2     IF NOT NULL_(KEY_BUFFER) THEN
;  406          0470  2     IF .KEY_BUFFER[0] NEQ 0
;  407          0471  2     THEN
;  408          0472  3         BEGIN                               ! Convert to DSC format
;  409          0473  3         STATUS = DSC_DTYPE(KEY_BUFFER[0], KEYS[BASE_]);
;  410          0474  3         IF NOT .STATUS THEN RETURN .STATUS;
;  411          0475  3         KEY_PARAM = KEYS[BASE_];
;  412          0476  2         END;
;  413          0477  2
;  414          0478  2
;  415          0479  2     ! Call SOR$BEGIN_SORT to do the rest of the processing
;  416          0480  2     !
;  417          0481  2     RETURN SOR$BEGIN_SORT(
;  418          0482  2         KEY_PARAM[BASE_],
;  419          0483  2         PARAM_(LRL),
;  420          0484  2         %REF(DEF_OPTIONS OR (IF NULL_(OPTIONS) THEN 0 ELSE .OPTIONS[0,L_])),
;  421          0485  2         PARAM_(FILE_ALLOC),
;  422          0486  3         (IF KEY_PARAM[BASE_] NEQ 0 THEN 0
;  423          0487  3         ELSE IF NULL_(USER_COMPARE) THEN RETURN SOR$_MISS_PARAM
;  424          0488  2         ELSE .USER_COMPARE),
;  425          0489  2         0,
;  426          0490  2         PARAM_(SORT_TYPE),
;  427          0491  2         PARAM_(WORK_FILES));
;  428          0492  1     END;


                                0004 00000          .ENTRY  SOR$INIT_SORT, Save R2                          ; 0355
                5E      F800    CE   9E 00002       MOVAB   -2048(SP), SP
                09              6C   91 00007       CMPB    (AP), #9                                        ; 0426
                                05   1F 0000A       BLSSU   1$
                          24    AC   D5 0000C       TSTL    36(AP)
                                1E   12 0000F       BNEQ    2$
00000000G       00              00   FB 00011 1$:   CALLS   #0, SOR$PASS_FILES                              ; 0436
                4B              50   E9 00018       BLBC    STATUS, 5$                                      ; 0437
                08              6C   91 0001B       CMPB    (AP), #8                                        ; 0442
                                17   1F 0001E       BLSSU   3$
                          20    AC   D5 00020       TSTL    32(AP)
                                12   13 00023       BEQL    3$
FFFFFFFC        8F        20    BC   D3 00025       BITL    @OPTIONS, #-4                                   ; 0444
                                08   13 0002D       BEQL    3$
                50 001C814C     8F   D0 0002F 2$:   MOVL    #1868108, R0                                    ; 0446
                                04   00036          RET
                06              6C   91 00037 3$:   CMPB    (AP), #6                                        ; 0455
                                0A   1F 0003A       BLSSU   4$
                          18    AC   D5 0003C       TSTL    24(AP)
                                05   13 0003F       BEQL    4$
             78 A1        18    BC   90 00041       MOVB    @TOT_KEY_SIZE, 120(CTX)
             5C A1        60    8F   88 00046 4$:   BISB2   #96, 92(CTX)                                    ; 0461
                                52   D4 0004B       CLRL    KEY_PARAM                                       ; 0468
                                6C   95 0004D       TSTB    (AP)                                            ; 0469
                                1C   13 0004F       BEQL    6$
                          04    AC   D5 00051       TSTL    4(AP)
                                17   13 00054       BEQL    6$
                          04    BC   B5 00056       TSTW    @KEY_BUFFER                                     ; 0470
```

```
                        12  13 00059          BEQL    6$                                      ;
                  04    AE  9F 0005B          PUSHAB  KEYS                                   ; 0473
                  04    AC  DD 0005E          PUSHL   KEY_BUFFER                              ;
            FED4  CF        02  FB 00061      CALLS   #2, DSC_DTYPE                           ;
                  79        50  E9 00066 5$:  BLBC    STATUS, 22$                             ; 0474
                  52  04    AE  9E 00069      MOVAB   KEYS, KEY_PARAM                         ; 0475
                  04        6C  91 0006D 6$:  CMPB    (AP), #4                                ; 0491
                            05  1F 00070      BLSSU   7$                                      ;
                  10    AC  DD 00072          PUSHL   WORK_FILES                              ;
                            02  11 00075      BRB     8$                                      ;
                            7E  D4 00077 7$:  CLRL    -(SP)                                   ;
                  05        6C  91 00079 8$:  CMPB    (AP), #5                                ; 0490
                            05  1F 0007C      BLSSU   9$                                      ;
                  14    AC  DD 0007E          PUSHL   SORT_TYPE                               ;
                            02  11 00081      BRB     10$                                     ;
                            7E  D4 00083 9$:  CLRL    -(SP)                                   ;
                            7E  D4 00085 10$: CLRL    -(SP)                                   ; 0482
                            52  D5 00087      TSTL    KEY_PARAM                               ; 0486
                            04  13 00089      BEQL    11$                                     ;
                            7E  D4 0008B      CLRL    -(SP)                                   ;
                            18  11 0008D      BRB     14$                                     ;
                  07        6C  91 0008F 11$: CMPB    (AP), #7                                ; 0487
                            05  1F 00092      BLSSU   12$                                     ;
                  1C    AC  D5 00094          TSTL    28(AP)                                  ;
                            08  12 00097      BNEQ    13$                                     ;
            50 001C80E4     8F  D0 00099 12$: MOVL    #1868004, R0                            ;
                                04  000A0     RET                                             ;
                  50    1C  AC  D0 000A1 13$: MOVL    USER_COMPARE, R0                        ; 0488
                            50  DD 000A5      PUSHL   R0                                      ; 0487
                  03        6C  91 000A7 14$: CMPB    (AP), #3                                ; 0485
                            05  1F 000AA      BLSSU   15$                                     ;
                  0C    AC  DD 000AC          PUSHL   FILE_ALLOC                              ;
                            02  11 000AF      BRB     16$                                     ;
                            7E  D4 000B1 15$: CLRL    -(SP)                                   ;
                  08        6C  91 000B3 16$: CMPB    (AP), #8                                ; 0484
                            05  1F 000B6      BLSSU   17$                                     ;
                  20    AC  D5 000B8          TSTL    32(AP)                                  ;
                            04  12 000BB      BNEQ    18$                                     ;
                            50  D4 000BD 17$: CLRL    R0                                      ;
                            04  11 000BF      BRB     19$                                     ;
                  50    20  BC  D0 000C1 18$: MOVL    @OPTIONS, R0                            ;
            14 AE       50      08  C9 000C5 19$: BISL3   #8, R0, 20(SP)                      ;
                  14    AE  9F 000CA          PUSHAB  20(SP)                                  ;
                  02        6C  91 000CD      CMPB    (AP), #2                                ; 0483
                            05  1F 000D0      BLSSU   20$                                     ;
                  08    AC  DD 000D2          PUSHL   LRL                                     ;
                            02  11 000D5      BRB     21$                                     ;
                            7E  D4 000D7 20$: CLRL    -(SP)                                   ;
                            52  DD 000D9 21$: PUSHL   KEY_PARAM                               ; 0482
            00000000G   00      08  FB 000DB      CALLS   #8, SOR$BEGIN_SORT                  ;
                                04  000E2 22$:   RET                                         ; 0492

; Routine Size:  227 bytes,    Routine Base:  SOR$RO_CODE + 011B
```

```
;  430          0493  1 GLOBAL ROUTINE SOR$INIT_MERGE
;  431          0494  1         (
;  432          0495  1         MERGE_ORDER:    REF VECTOR[,BYTE],
;  433          0496  1         KEY_BUFFER:     REF VECTOR[,WORD],
;  434          0497  1         LRL:            REF VECTOR[,WORD],
;  435          0498  1         OPTIONS:        REF BLOCK[1],
;  436          0499  1         USER_COMPARE,
;  437          0500  1         USER_INPUT,
;  438          0501  1         EXTRA
;  439          0502  1         ) =
;  440          0503  1 !++
;  441          0504  1 !
;  442          0505  1 ! FUNCTIONAL DESCRIPTION:
;  443          0506  1 !
;  444          0507  1 !       This routine converts keys to DSC format,
;  445          0508  1 !       and calls SOR$BEGIN_MERGE to initialize the merge.
;  446          0509  1 !
;  447          0510  1 ! FORMAL PARAMETERS:
;  448          0511  1 !
;  449          0512  1 !       MERGE_ORDER.rab         Order of the merge
;  450          0513  1 !       KEY_BUFFER.raw          Key buffer address
;  451          0514  1 !       LRL.rwu.r               Longest record length
;  452          0515  1 !       OPTIONS.rlu.r           Option bits
;  453          0516  1 !       USER_COMPARE,           User-written comparison routine
;  454          0517  1 !       USER_INPUT              User-written input routine
;  455          0518  1 !
;  456          0519  1 !       All parameters are optional.
;  457          0520  1 !
;  458          0521  1 ! IMPLICIT INPUTS:
;  459          0522  1 !
;  460          0523  1 !       NONE
;  461          0524  1 !
;  462          0525  1 ! IMPLICIT OUTPUTS:
;  463          0526  1 !
;  464          0527  1 !       NONE
;  465          0528  1 !
;  466          0529  1 ! ROUTINE VALUE:
;  467          0530  1 !
;  468          0531  1 !       Status code.
;  469          0532  1 !
;  470          0533  1 ! SIDE EFFECTS:
;  471          0534  1 !
;  472          0535  1 !       The work files are defined (not necessarily created), the working set
;  473          0536  1 !       is extended and the virtual memory is extended.
;  474          0537  1 !
;  475          0538  1 !--
;  476          0539  2     BEGIN
;  477          0540  2     FIRSTPARAMETER_(MERGE_ORDER);       ! Required by PRESENT_ and NULL_ macros
;  478          0541  2
;  479          0542  2     LITERAL
;  480          0543  2         USED_OPTIONS =
;  481          0544  2             MASK_(OPT_EBCDIC, OPT_SEQ_CHECK),
;  482          0545  2         DEF_OPTIONS =
;  483          0546  2             MASK_(OPT_NOSIGNAL);
;  484          0547  2
;  485          0548  2     LOCAL
;  486          0549  2         KEYS:   KEY_BLOCK,              ! DSC format keys
```

```
;  487          0550  2          CTX:     REF CTX_BLOCK,          . Addr of context area
;  488          0551  2          KEY_PARAM: REF KEY_BLOCK,
;  489          0552  2          STATUS;
;  490          0553  2
;  491          0554  2      MACRO
;  492          0555  2          PARAM_(A) = (IF PRESENT_(A) THEN .A ELSE 0) %;
;  493          0556  2
;  494          0557  2
;  495          0558  2      ! If the user wants concurrent sorts, he must use the new interface to
;  496          0559  2      ! get them.
;  497          0560  2      !
;  498          0561  2      IF NOT NULL_(EXTRA) THEN RETURN SOR$_UNDOPTION;
;  499          0562  2
;  500          0563  2
;  501          0564  2      ! Check the options specified
;  502          0565  2      !
;  503          0566  3      IF NOT NULL_(OPTIONS)
;  504          0567  2      THEN
;  505          0568  2          IF (.OPTIONS[0,L_] AND NOT USED_OPTIONS) NEQ 0
;  506          0569  2          THEN
;  507          0570  2              RETURN SOR$_UNDOPTION;              ! Invalid options specified
;  508          0571  2
;  509          0572  2
;  510          0573  2      ! Get the context area
;  511          0574  2      !
;  512          0575  2      ! We know that SOR$PASS_FILES will allocate the context area,
;  513          0576  2      ! and that SOR$PASS_FILES may be called with no parameters.
;  514          0577  2      ! Finally, as a hack, SOR$PASS_FILES returns the address of the context
;  515          0578  2      ! area in R1.
;  516          0579  2      !
;  517          0580  2      STATUS = SOR$PASS_FILES(;CTX);
;  518          0581  2      IF NOT .STATUS THEN RETURN .STATUS;
;  519          0582  2
;  520          0583  2
;  521          0584  2      !
;  522          0585  2      ! Do not set the COM_HACK_STRIP bit for merges
;  523          0586  2      !
;  524          0587  2
;  525          0588  2
;  526          0589  2      ! Set the bit indicating only 2 parameters are passed to callback routines
;  527          0590  2      !
;  528          0591  2      CTX[COM_HACK_2ARGS] = TRUE;
;  529          0592  2
;  530          0593  2
;  531          0594  2      ! Convert keys to the new format
;  532          0595  2      ! Note: "If you pass both key buffer address and comparison address,
;  533          0596  2      ! SORT ignores the comparison routine address".
;  534          0597  2      !
;  535          0598  2      KEY_PARAM = 0;
;  536          0599  2      IF NOT NULL_(KEY_BUFFER) THEN
;  537          0600  2      IF .KEY_BUFFER[0] NEQ 0
;  538          0601  2      THEN
;  539          0602  3          BEGIN                               ! Convert to DSC format
;  540          0603  3          STATUS = DSC_DTYPE(KEY_BUFFER[0], KEYS[BASE_]);
;  541          0604  3          IF NOT .STATUS THEN RETURN .STATUS;
;  542          0605  3          KEY_PARAM = KEYS[BASE_];
;  543          0606  2          END;
```

```
;  544          0607  2
;  545          0608  2
;  546          0609  2      ! Call SOR$BEGIN_MERGE to do the rest of the processing
;  547          0610  2
;  548          0611  2      RETURN SOR$BEGIN_MERGE(
;  549          0612  2          KEY_PARAM[BASE_],
;  550          0613  2          PARAM_(LRL),
;  551          0614  2          %REF(DEF_OPTIONS OR (IF NULL_(OPTIONS) THEN 0 ELSE .OPTIONS[0,L_])),
;  552          0615  2          PARAM_(MERGE_ORDER),
;  553          0616  3          (IF KEY_PARAM[BASE_] NEQ 0 THEN 0
;  554          0617  3          ELSE IF NULL_(USER_COMPARE) THEN RETURN SOR$_MISS_PARAM
;  555          0618  2          ELSE .USER_COMPARE),
;  556          0619  2          0,
;  557          0620  3          PARAM_(USER_INPUT)
;  558          0621  2          );
;  559          0622  1      END;


                                                0004 00000          .ENTRY  SOR$INIT_MERGE, Save R2                         ; 0493
                                5E      F800    CE   9E 00002       MOVAB   -2048(SP), SP                                   ;
                                07              6C   91 00007       CMPB    (AP), #7                                        ; 0561
                                                05   1F 0000A       BLSSU   1$                                              ;
                                        1C      AC   D5 0000C       TSTL    28(AP)                                          ;
                                                14   12 0000F       BNEQ    2$                                              ;
                                04              6C   91 00011 1$:   CMPB    (AP), #4                                        ; 0566
                                                17   1F 00014       BLSSU   3$                                              ;
                                        10      AC   D5 00016       TSTL    16(AP)                                          ;
                                                12   13 00019       BEQL    3$                                              ;
                    FFFFFFED    8F          10      BC   D3 0001B       BITL    @OPTIONS, #-19                              ; 0568
                                                08   13 00023       BEQL    3$                                              ;
                                50  001C814C     8F   D0 00025 2$:   MOVL    #1868108, R0                                ; 0570
                                                04   0002C           RET                                                     ;
                    00000000G   00              00   FB 0002D 3$:   CALLS   #0, SOR$PASS_FILES                           ; 0580
                                20              50   E9 00034       BLBC    STATUS, 4$                                      ; 0581
                          5C    A1              20   88 00037       BISB2   #32, 92(CTX)                                    ; 0591
                                                52   D4 0003B       CLRL    KEY_PARAM                                       ; 0598
                                02              6C   91 0003D       CMPB    (AP), #2                                        ; 0599
                                                1C   1F 00040       BLSSU   5$                                              ;
                                        08      AC   D5 00042       TSTL    8(AP)                                           ;
                                                17   13 00045       BEQL    5$                                              ;
                                        08      BC   B5 00047       TSTW    @KEY_BUFFER                                     ; 0600
                                                12   13 0004A       BEQL    5$                                              ;
                                        04      AE   9F 0004C       PUSHAB  KEYS                                            ; 0603
                                        08      AC   DD 0004F       PUSHL   KEY_BUFFER                                      ;
                        FE00    CF              02   FB 00052       CALLS   #2, DSC_DTYPE                                   ;
                                6C              50   E9 00057 4$:   BLBC    STATUS, 19$                                     ; 0604
                                52          04  AE   9E 0005A       MOVAB   KEYS, KEY_PARAM                                 ; 0605
                                06              6C   91 0005E 5$:   CMPB    (AP), #6                                        ; 0620
                                                05   1F 00061       BLSSU   6$                                              ;
                                        18      AC   DD 00063       PUSHL   USER_INPUT                                      ;
                                                02   11 00066       BRB     7$                                              ;
                                                7E   D4 00068 6$:   CLRL    -(SP)                                           ;
                                                7E   D4 0006A 7$:   CLRL    -(SP)                                           ; 0612
                                                52   D5 0006C       TSTL    KEY_PARAM                                       ; 0616
```

```
                                    04  13 0006E           BEQL    8$                                      ;
                                    7E  D4 00070           CLRL    -(SP)                                   ;
                                    18  11 00072           BRB     11$                                     ;
                              05        6C  91 00074 8$:     CMPB    (AP), #5                                ; 0617
                                        05  1F 00077           BLSSU   9$                                      ;
                                 14     AC  D5 00079           TSTL    20(AP)                                  ;
                                        08  12 0007C           BNEQ    10$                                     ;
                              50 001C80E4   8F  D0 0007E 9$:     MOVL    #1868004, R0                            ;
                                        04  00085              RET                                             ;
                              50     14     AC  D0 00086 10$:    MOVL    USER_COMPARE, R0                        ; 0618
                                        50  DD 0008A           PUSHL   R0                                      ; 0617
                                        6C  95 0008C 11$:     TSTB    (AP)                                    ; 0615
                                        05  13 0008E           BEQL    12$                                     ;
                                 04     AC  DD 00090           PUSHL   MERGE_ORDER                             ;
                                        02  11 00093           BRB     13$                                     ;
                                        7E  D4 00095 12$:     CLRL    -(SP)                                   ;
                              04        6C  91 00097 13$:     CMPB    (AP), #4                                ; 0614
                                        05  1F 0009A           BLSSU   14$                                     ;
                                 10     AC  D5 0009C           TSTL    16(AP)                                  ;
                                        04  12 0009F           BNEQ    15$                                     ;
                                        50  D4 000A1 14$:     CLRL    R0                                      ;
                                        04  11 000A3           BRB     16$                                     ;
                              50        10  BC  D0 000A5 15$:     MOVL    @OPTIONS, R0                            ;
                 10   AE      50        08  C9 000A9 16$:     BISL3   #8, R0, 16(SP)                          ;
                                 10     AE  9F 000AE           PUSHAB  16(SP)                                  ;
                              03        6C  91 000B1           CMPB    (AP), #3                                ; 0613
                                        05  1F 000B4           BLSSU   17$                                     ;
                                 0C     AC  DD 000B6           PUSHL   LRL                                     ;
                                        02  11 000B9           BRB     18$                                     ;
                                        7E  D4 000BB 17$:     CLRL    -(SP)                                   ;
                                        52  DD 000BD 18$:     PUSHL   KEY_PARAM                               ; 0612
                    00000000G   00      07  FB 000BF           CALLS   #7, SOR$BEGIN_MERGE                     ;
                                        04  000C6 19$:     RET                                             ; 0622

; Routine Size:  199 bytes,    Routine Base:  SOR$RO_CODE + 01FE
```

```
;  561          0623  1 GLOBAL ROUTINE SOR$DO_MERGE
;  562          0624  1         (
;  563          0625  1         EXTRA
;  564          0626  1         ) =
;  565          0627  1 !++
;  566          0628  1 !
;  567          0629  1 ! FUNCTIONAL DESCRIPTION:
;  568          0630  1 !
;  569          0631  1 !       "Perform the merge".
;  570          0632  1 !
;  571          0633  1 ! FORMAL PARAMETERS:
;  572          0634  1 !
;  573          0635  1 !       NONE
;  574          0636  1 !
;  575          0637  1 ! IMPLICIT INPUTS:
;  576          0638  1 !
;  577          0639  1 !       NONE
;  578          0640  1 !
;  579          0641  1 ! IMPLICIT OUTPUTS:
;  580          0642  1 !
;  581          0643  1 !       NONE
;  582          0644  1 !
;  583          0645  1 ! ROUTINE VALUE:
;  584          0646  1 !
;  585          0647  1 !       Status code.
;  586          0648  1 !
;  587          0649  1 ! SIDE EFFECTS:
;  588          0650  1 !
;  589          0651  1 !       NONE
;  590          0652  1 !
;  591          0653  1 !--
;  592          0654  2     BEGIN
;  593          0655  2     FIRSTPARAMETER_(EXTRA);     ! Required by PRESENT_ and NULL_ macros
;  594          0656  2
;  595          0657  2     BUILTIN
;  596          0658  2         AP,
;  597          0659  2         CALLG;
;  598          0660  2
;  599          0661  2     LOCAL
;  600          0662  2         CTX:    REF CTX_BLOCK,                  ! Addr of context area
;  601          0663  2         STATUS;
;  602          0664  2
;  603          0665  2
;  604          0666  2     ! If the user wants concurrent sorts, he must use the new interface to
;  605          0667  2     ! get them.
;  606          0668  2     !
;  607          0669  2     IF NOT NULL_(EXTRA) THEN RETURN SOR$_UNDOPTION;
;  608          0670  2
;  609          0671  2
;  610          0672  2     ! Get the context area
;  611          0673  2     !
;  612          0674  2     ! We know that SOR$PASS_FILES will allocate the context area,
;  613          0675  2     ! and that SOR$PASS_FILES may be called with no parameters.
;  614          0676  2     ! Finally, as a hack, SOR$PASS_FILES returns the address of the context
;  615          0677  2     ! area in R1.
;  616          0678  2     !
;  617          0679  2     STATUS = SOR$PASS_FILES(;CTX);
```

```
;  618          0680  2     IF NOT .STATUS THEN RETURN .STATUS;
;  619          0681  2
;  620          0682  2
;  621          0683  2     ! Check the flow control flags.
;  622          0684  2     !
;  623          0685  2     IF NOT .CTX[COM_FLO_DOMERGE] THEN RETURN SOR$_SORT_ON;
;  624          0686  2
;  625          0687  2
;  626          0688  2     ! Let SOR$END_SORT clean up
;  627          0689  2     !
;  628          0690  2     RETURN CALLG(.AP, SOR$END_SORT);
;  629          0691  1     END;


                                        0000 00000          .ENTRY  SOR$DO_MERGE, Save nothing                      ; 0623
                                    6C  95 00002            TSTB    (AP)                                            ; 0669
                                    0D  13 00004            BEQL    1$                                              ;
                                04  AC  D5 00006            TSTL    4(AP)                                           ;
                                    08  13 00009            BEQL    1$                                              ;
                          50 001C814C   8F  D0 0000B        MOVL    #1868108, R0                                    ;
                                        04 00012            RET                                                     ;
                00000000G   00          00  FB 00013 1$:    CALLS   #0, SOR$PASS_FILES                              ; 0679
                            14          50  E9 0001A        BLBC    STATUS, 3$                                      ; 0680
            08      5C      A1          03  E0 0001D        BBS     #3, 92(CTX), 2$                                 ; 0685
                          50 001C802C   8F  D0 00022        MOVL    #1867820, R0                                    ;
                                        04 00029            RET                                                     ;
                00000000G   00          6C  FA 0002A 2$:    CALLG   (AP), SOR$END_SORT                              ; 0690
                                        04 00031 3$:        RET                                                     ; 0691

; Routine Size:  50 bytes,    Routine Base:  SOR$RO_CODE + 02C5
```

```
;  631          0692  1 END
;  632          0693  0 ELUDOM
```

```
;                                PSECT SUMMARY
;
;      Name                      Bytes                          Attributes
;
;  SOR$RO_CODE                     759  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  .  ABS  .                         0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)
```

```
;                             Library Statistics
;
;                                      -------- Symbols --------      Pages       Processing
;      File                            Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1    9776        23         0         581        00:01.0
;  _$255$DUA28:[SORT32.SRC]SORLIB.L32;1  409       141        34          34        00:00.4
```

```
;                                    COMMAND QUALIFIERS

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

; Size:          674 code + 85 data bytes
; Run Time:        00:17.9
; Elapsed Time:    01:02.0
; Lines/CPU Min:    2325
; Lexemes/CPU-Min: 24701
; Memory Used:  150 pages
; Compilation Complete
```

SORCOMMAN
LIS

LIBFIXUPD
LIS

REQSYM
R32

CRETRANS
LIS

SFKEYWRD
LIS

SORCOLLAT
LIS

OPCODES
LIS

SORARCHAI
LIS

SORCOLUTI
LIS