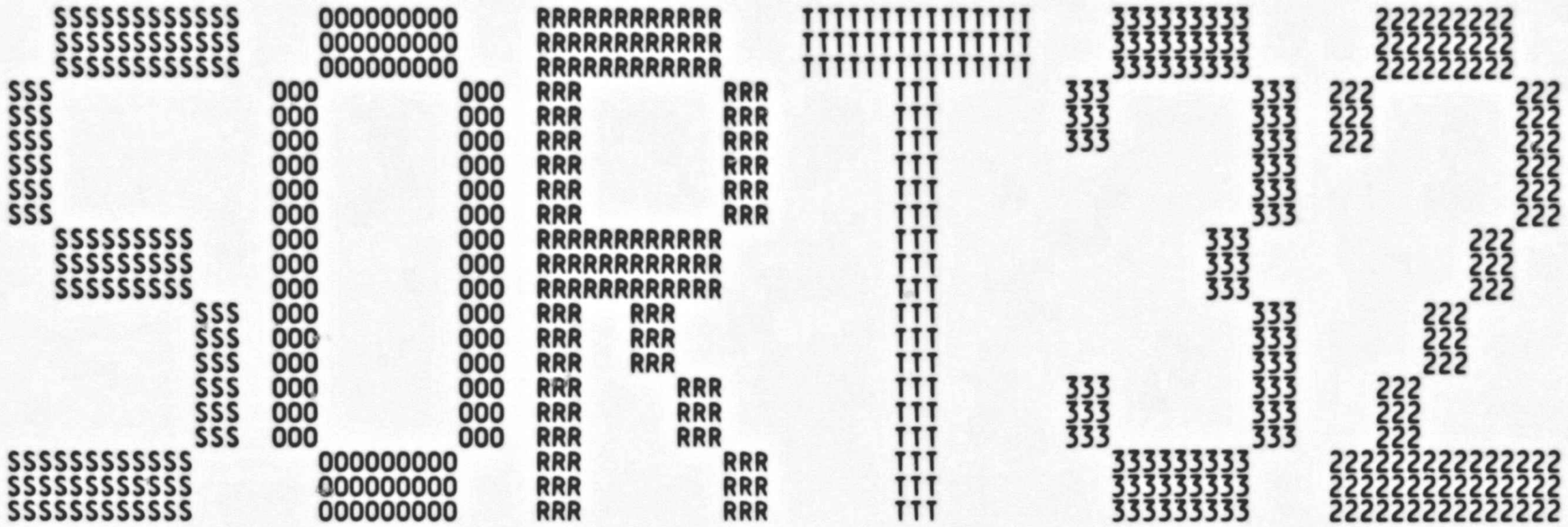
SSSSSSSSSSS 000000000 RRRRRRRRRRR TTTTTTTTTTTTTTT 333333333 222222222
SSSSSSSSSSS 000000000 RRRRRRRRRRR TTTTTTTTTTTTTTT 333333333 222222222
SSSSSSSSSSS 000000000 RRRRRRRRRRR TTTTTTTTTTTTTTT 333333333 222222222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000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

FILEID**SRTSPC

```
  SSSSSSSS  RRRRRRRR    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      CCCCCCCC
  SSSSSSSS  RRRRRRRR    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      CCCCCCCC
SS          RR      RR      TT      SS          PP      PP  CC
SS          RR      RR      TT      SS          PP      PP  CC
SS          RR      RR      TT      SS          PP      PP  CC
SS          RR      RR      TT      SS          PP      PP  CC
  SSSSSS    RRRRRRRR        TT        SSSSSS    PPPPPPPP    CC
  SSSSSS    RRRRRRRR        TT        SSSSSS    PPPPPPPP    CC
        SS  RR  RR          TT              SS  PP          CC
        SS  RR  RR          TT              SS  PP          CC
        SS  RR    RR        TT              SS  PP          CC                ....
        SS  RR    RR        TT              SS  PP          CC                ....
SSSSSSSS    RR      RR      TT      SSSSSSSS    PP            CCCCCCCC      ....
SSSSSSSS    RR      RR      TT      SSSSSSSS    PP            CCCCCCCC      ....

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

```
0001  0     ! File: SRTSPC.REQ IDENT = 'V04-000'    ! File: SRTSPC.REQ Edit: PDG3028
0002  0     !
0003  0     !
0004  0     !****************************************************************************
0005  0     !*                                                                          *
0006  0     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0007  0     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0008  0     !*  ALL RIGHTS RESERVED.                                                    *
0009  0     !*                                                                          *
0010  0     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0011  0     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0012  0     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0013  0     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0014  0     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0015  0     !*  TRANSFERRED.                                                            *
0016  0     !*                                                                          *
0017  0     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0018  0     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0019  0     !*  CORPORATION.                                                            *
0020  0     !*                                                                          *
0021  0     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0022  0     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0023  0     !*                                                                          *
0024  0     !*                                                                          *
0025  0     !****************************************************************************
0026  0     !
0027  0     !
0028  0     !++
0029  0     !
0030  0     ! FACILITY:     VAX-11 SORT/MERGE,  PDP-11 SORT/MERGE
0031  0     !
0032  0     ! ABSTRACT:
0033  0     !
0034  0     !       This require file is for data structures returned from specification
0035  0     !       file processing.
0036  0     !       This file is used as a library source.
0037  0     !
0038  0     ! ENVIRONMENT:  VAX/VMS user mode
0039  0     !
0040  0     ! AUTHOR: V. Bennison, CREATION DATE: 03-May-1982
0041  0     !
0042  0     ! MODIFIED BY:
0043  0     !
0044  0     !       31-Aug-1982     PDG
0045  0     !               Add definitions that are required for SRTSPC.BLI.
0046  0     !       T03-016 Rework TDT table to give precedence to AND/OR.  PDG 13-Dec-1982
0047  0     !       T03-017 Add WF_NAMES, CFT indices of work file names.  PDG 26-Dec-1982
0048  0     !       T03-018 Removed RDT_VAR.  PDG 3-Jan-1983
0049  0     !       T03-019 Removed PT/ST_ADR; added WRK_SIZ, BS_DECM.  PDG 26-Jan-1983
0050  0     !       T03-020 Add FDT_SCALE and CA_PAD.  PDG 8-Feb-1983
0051  0     !       T03-022 Fix computation for packed in KFT_UNITS_.  PDG 11-Feb-1983
0052  0     !       T03-022 Remove unreferenced fields.  PDG 16-Mar-1983
0053  0     !       T03-024 Work around Bliss bug with CA_LINKAGE_LB.  PDG 12-May-1983
0054  0     !       T03-025 Define KFT_NDE_SIZ_ for BLISST6.  PDG 26-Jul-1983
0055  0     !       T03-026 Put WHILE_FAIL_ here.  PDG 1-Aug-1983
0056  0     !       T03-027 Word-align elements in CON_SYM_TAB.  PDG 1-Aug-1983
0057  0     !       T03-028 Make sharing of code easier to maintain.  PDG 31-Jan-1984
```

```
;   0058  0     !--
```

```
;    0059  0     !----------------------------------------------------------
;    0060  0     !                FIELD DEFINITION TABLE (FDT)
;    0061  0     !----------------------------------------------------------
;    0062  0     LITERAL
;    0063  0         FDT_MAX = 255,              !maximum number of entries in FDT
;    0064  0         FDT_UNIT = 6;               ! Size in bytes, must be even for bliss16
;    0065  0                                     ! because of 16 bit field
;    0066  0     STRUCTURE
;    0067  0         FDT_TAB[ O,B,P,S,E; BS ] =
;    0068  0             [ BS*FDT_UNIT ]
;    0069  0             ( FDT_TAB + O*FDT_UNIT + B )<P,S,E>;
;    0070  0     MACRO
;    0071  0         FDT_TYPE    = 0, 0,  8, 0 %,            ! Data type
;    0072  0         FDT_SCALE   = 1, 0,  8, 1 %,            ! Scale factor
;    0073  0         FDT_FLD_POS = 2, 0, 16, 0 %,            ! Position of field
;    0074  0         FDT_FLD_SIZ = 4, 0, 16, 0 %;            ! Size of field
```

```
;  0075  0     !----------------------------------------------------------------
;  0076  0     !                   TEST DEFINITION TABLE (TDT)
;  0077  0     !----------------------------------------------------------------
;  0078  0     LITERAL
;  0079  0         TDT_MAX = 255,              !maximum number of entries in TDT
;  0080  0         TDT_UNIT = 4;                                ! Size in bytes
;  0081  0
;  0082  0     STRUCTURE
;  0083  0         TDT_TAB[ O,B,P,S,E; BS ] =
;  0084  0             [ BS*TDT_UNIT ]
;  0085  0             ( TDT_TAB + O*TDT_UNIT + B )<P,S,E>;
;  0086  0
;  0087  0     MACRO
;  0088  0         TDT_TRUE    =  0, 0, 1, 0 %,        ! Set to simply return TRUE
;  0089  0         TDT_CMP     =  0, 1, 3, 0 %,        ! The comparison flags
;  0090  0         TDT_EQL     =  0, 1, 1, 0 %,        ! True if "Equal to" succeeds
;  0091  0         TDT_LSS     =  0, 2, 1, 0 %,        ! True if "Less than" succeeds
;  0092  0         TDT_GTR     =  0, 3, 1, 0 %,        ! True if "Greater than" succeeds
;  0093  0         TDT_CONSTANT=  0, 4, 1, 0 %,        ! True iff FLD_TWO is to CFT
;  0094  0         TDT_FLD_ONE =  1, 0, 8, 0 %,        ! Index in FDT of 1st field
;  0095  0         TDT_FLD_TWO =  2, 0, 8, 0 %,        ! Index in FDT (or CFT) of 2nd field
;  0096  0         TDT_GOTO    =  3, 0, 8, 0 %;        ! TDT index adjustment
;  0097  0
;  0098  0     ! This structure should only be referenced by the routines that builds it,
;  0099  0     ! and the routine SOR$$TDT!
;  0100  0
;  0101  0     ! This table is used as follows:
;  0102  0     !
;  0103  0     !       Set IX to the index of the test description to test
;  0104  0     !Loop:  If TDT_TRUE is clear then return TRUE
;  0105  0     !       If the comparison between FLD_ONE and FLD_TWO is true
;  0106  0     !           (according to the EQL/LSS7GTR bits)
;  0107  0     !       then
;  0108  0     !           if TDT_GOTO is zero then return false else add TDT_GOTO to IX
;  0109  0     !       else
;  0110  0     !           add 1 to IX
;  0111  0     !       goto Loop
;  0112  0     !
```

```
;  0113  0     !----------------------------------------------------------------
;  0114  0     !                KEY/DATA FIELD TABLE (KFT)
;  0115  0     !----------------------------------------------------------------
;  0116  0     LITERAL
;  0117  0         KFT_MAX = 255,              !maximum number of entries in KFT
;  0118  0         KFT_UNIT = 8;               ! Size in bytes, must be even for bliss16
;  0119  0                                     ! because of 16 bit field
;  0120  0
;  0121  0     STRUCTURE
;  0122  0         KFT_TAB[ O,B,P,S,E; BS ] =
;  0123  0             [ BS*KFT_UNIT ]
;  0124  0             ( KFT_TAB + O*KFT_UNIT + B )<P,S,E>;
;  0125  0
;  0126  0     MACRO
;  0127  0         KFT_NDE_POS =  0, 0, 16, 0 %,           ! Starting position in node
;  0128  0
;  0129  0         KFT_CONTINUE =  3, 0, 1, 0 %,           ! Continue = 1
;  0130  0         KFT_CONSTANT =  3, 1, 1, 0 %,           ! True iff FDT_IDX is to CFT
;  0131  0         KFT_CONT_CDX =  3, 2, 1, 0 %,           ! Continued condition = 1
;  0132  0         KFT_CONDX    =  3, 3, 1, 0 %,           ! Conditional field = 1
;  0133  0         KFT_BUILD    =  3, 4, 1, 0 %,           ! Build the key = 1
;  0134  0         KFT_DESCEND =  3, 5, 1, 0 %,            ! Asc/desc, descend = 1
;  0135  0         KFT_DATA     =  3, 6, 1, 0 %,           ! Key or data, data = 1
;  0136  0
;  0137  0         KFT_FDT_IDX =  4, 0,  8, 0 %,           ! Index in FDT (or CFT)
;  0138  0         KFT_TDT_IDX =  5, 0,  8, 0 %,           ! TDT index for forces
;  0139  0
;  0140  0         KFT_NDE_SIZ =  6, 0, 16, 0 %;           ! Size (bytes) in internal node
```

```
;    0141  0      !----------------------------------------------------------
;    0142  0      !                RECORD DEFINITION TABLE (RDT)
;    0143  0      !----------------------------------------------------------
;    0144  0      LITERAL
;    0145  0          RDT_MAX = 64,                   !maximum number of entries in RDT
;    0146  0          RDT_UNIT = 6;                   ! Size in bytes
;    0147  0
;    0148  0      STRUCTURE
;    0149  0          RDT_TAB[ O,B,P,S,E; BS ] =
;    0150  0              [ BS*RDT_UNIT ]
;    0151  0              ( RDT_TAB + O*RDT_UNIT + B )<P,S,E>;
;    0152  0
;    0153  0      MACRO
;    0154  0          RDT_INCLUDE =  0, 0,  1, 0 %,               ! Include/omit, Include = 1
;    0155  0          RDT_CONDX   =  0, 1,  1, 0 %,               ! Conditional = 1
;    0156  0
;    0157  0          RDT_TDT_IDX =  1, 0,  8, 0 %,               ! Index into TDT
;    0158  0          RDT_KCT_ADR =  2, 0, 16, 0 %,               ! For Sort-11 only
;    0159  0          RDT_KFT_IDX =  4, 0,  8, 0 %;               ! Index into KFT
;    0160  0
;    0161  0      ! The RDT table is scanned sequentially until either an unconditional entry is
;    0162  0      ! found, or until a condition (via RDT_TDT_IDX) passes.  This matched entry
;    0163  0      ! describes whether to omit or include the record (RDT_INCLUDE).  If included,
;    0164  0      ! then RDT_KFT_IDX is used to index the KFT table, for record reformatting.
;    0165  0      !
```

```
;  0166  0     !----------------------------------------------------------
;  0167  0     !                CONSTANT FIELD TABLE (CFT)
;  0168  0     !----------------------------------------------------------
;  0169  0     LITERAL
;  0170  0         CFT_MAX = 255,                  !maximum number of entries in CFT
;  0171  0         CFT_UNIT = 2+%BPADDR/8;         ! Size in bytes
;  0172  0
;  0173  0     STRUCTURE
;  0174  0         CFT_TAB[ O,B,P,S,E; BS ] =
;  0175  0             [ BS*CFT_UNIT ]
;  0176  0             ( CFT_TAB + O*CFT_UNIT + B )<P,S,E>;
;  0177  0
;  0178  0     MACRO
;  0179  0         CFT_CON_LEN =  0, 0,  8, 0 %,                   ! Length of constant
;  0180  0         CFT_CON_ADR =  2, 0, %BPADDR, 0 %;  ! Address of constant
```

```
;    0181  0      !----------------------------------------------------------
;    0182  0      !                  COMMON DEFINITIONS
;    0183  0      !----------------------------------------------------------
;    0184  0
;  L 0185  0      %IF %BLISS(BLISS32)
;    0186  0      %THEN
;    0187  0              LIBRARY 'SYS$LIBRARY:STARLET';
;    0188  0              LIBRARY 'SRC$:SORLIB';
;  U 0189  0      %ELSE
;  U 0190  0              LIBRARY 'S11V3SRC:SMCOM';
;    0191  0      %FI
;    0192  0
;    0193  0
;    0194  0      ! Define the linkage to the common routines
;    0195  0      !
;    0196  0      LITERAL
;    0197  0          LB_REG = 4;
;    0198  0      LINKAGE
;    0199  0          CA_LINKAGE =
;    0200  0              %BLISS32( CALL:GLOBAL(CA=COM_REG_CTX) ) ! MUST BE SAME AS CAL_CTXREG!
;    0201  0              %BLISS16( JSR ),
;    0202  0          CA_LINKAGE_LB =              ! Same as CA_LINKAGE, with an extra register
;    0203  0              %BLISS32( CALL:GLOBAL(CA=COM_REG_CTX, LB=LB_REG) )
;    0204  0              %BLISS16( JSR :GLOBAL(                LB=LB_REG) ),
;    0205  0          CA_LINK_SEGMENT =
;    0206  0              %BLISS16( JSR )
;  P 0207  0              %BLISS32( JSB (
;  P 0208  0                      REGISTER=6,
;  P 0209  0                      REGISTER=COM_REG_SRC2):
;  P 0210  0                      GLOBAL(CA=COM_REG_CTX)
;  P 0211  0                      PRESERVE(COM_REG_SRC2)
;  P 0212  0                      NOTUSED(7,8,9)
;    0213  0                      NOPRESERVE(0,1,2,3,4,5));
;    0214  0
;    0215  0
;    0216  0      ! A macro to declare/get the address of the common area
;    0217  0      !
;    0218  0      MACRO
;  M 0219  0          CA_AREA_( X ) =
;  M 0220  0              %IF %BLISS(BLISS32)
;  M 0221  0              %THEN
;  M 0222  0                  EXTERNAL REGISTER
;  M 0223  0                      %IF %NULL(X) %THEN CA %ELSE X %FI
;  M 0224  0                          = COM_REG_CTX:       REF BLOCK[CTX_K_SIZE]
;  M 0225  0                                               FIELD(CTX_FIELDS);
;  M 0226  0              %ELSE
;  M 0227  0                  %IF NOT %NULL(X)
;  M 0228  0                  %THEN
;  M 0229  0                      LOCAL
;  M 0230  0                          X : REF BLOCK [, %UPVAL] FIELD (COM_FIELDS);
;  M 0231  0                      %QUOTE GET_IMPAREA_( X );
;  M 0232  0                  %FI
;    0233  0              %FI %;
;    0234  0
;    0235  0      ! Specification file error messages
;    0236  0      !
;    0237  0      !LITERAL
```

```
;    0238  0      !     SRTIWA = SOR$_SRTIWA,      !insufficient work area
;    0239  0      !     SPCOVR = SOR$_SPCOVR,      !warning: overriden specification
;    0240  0      !     SPCMIS = SOR$_SPCMIS,      !warning: invalid merge specification
;    0241  0      !     SPCSIS = SOR$_SPCSIS,      !warning: invalid sort specification
;    0242  0      !     SPCIVP = SOR$_SPCIVP,      !invalid sort process
;    0243  0      !     SPCIVS = SOR$_SPCIVS,      !invalid specification
;    0244  0      !     SPCIVC = SOR$_SPCIVC,      !invalid collating sequence specification
;    0245  0      !     SPCIVF = SOR$_SPCIVF,      !invalid field specification
;    0246  0      !     SPCIVD = SOR$_SPCIVD,      !invalid data type
;    0247  0      !     SPCIVX = SOR$_SPCIVX,      !invalid condition specification
;    0248  0      !     SPCIVK = SOR$_SPCIVK,      !invalid key or data specification
;    0249  0      !     SPCIVI = SOR$_SPCIVI;      !invalid include or omit specification
;    0250  0
;    0251  0
;    0252  0
;    0253  0      ! A macro to expand fields
;    0254  0      !
;  L 0255  0      %IF %BLISS(BLISS32)
;    0256  0      %THEN   MACRO _(X,Y) = %QUOTE %EXPAND %FIELDEXPAND(X) %;
;  U 0257  0      %ELSE   MACRO _(X,Y) = %QUOTE %EXPAND %FIELDEXPAND(Y) %;
;    0258  0      %FI
;    0259  0
;    0260  0      MACRO
;    0261  0              ! Sort/Merge process information
;    0262  0              !
;    0263  0              CA_PROCESS      = %EXPAND _(COM_SORT_TYPE,      COM_PROCESS_) %,
;    0264  0              CA_PROCESS_OVR  = %EXPAND _(COM_OVR_PROC,       COM_PROC_OVR_) %,
;    0265  0              CA_VAR_MERGE    = %EXPAND _(COM_MERGE,          COM_MERGE_) %,
;    0266  0
;    0267  0              ! Collating information
;    0268  0              !
;    0269  0              CA_TIE_BREAK    = %EXPAND _(COM_TIE_BREAK,      COM_TIE_BREAK_) %,
;    0270  0              CA_ST_ADR       = %EXPAND _(COM_COLLATE,        COM_CS_TAB_ADR_) %,
;    0271  0              CA_ST_SIZ       = %EXPAND _(COM_ST_SIZ,         COM_CS_TAB_SIZ_) %,
;    0272  0              CA_BS_DECM      = %EXPAND _(COM_BS_DECM,        COM_BS_DECM_) %,
;    0273  0              CA_PAD          = %EXPAND _(COM_PAD,            COM_PAD_CHAR_) %,
;    0274  0
;    0275  0              ! Keys and stable information
;    0276  0              !
;    0277  0              CA_KEY_OVR      = %EXPAND _(COM_OVR_KEY,        COM_KEY_OVR_) %,
;    0278  0              CA_CHKSEQ       = %EXPAND _(COM_SEQ_CHECK,      COM_CH_SEQ_) %,
;    0279  0              CA_CHKSEQ_OVR   = %EXPAND _(COM_SEQ_CHECK,      COM_CHSEQ_OVR_) %,
;    0280  0              CA_STABLE       = %EXPAND _(COM_STABLE,         COM_STABLE_) %,
;    0281  0              CA_STABLE_OVR   = %EXPAND _(COM_STABLE,         COM_STBL_OVR_) %,
;    0282  0              CA_COLSEQ_OVR   = %EXPAND _(COM_OVR_COLSEQ,     COM_CSEQ_OVR_) %,
;    0283  0
;    0284  0              ! Record reformatting, and other tables
;    0285  0              !
;    0286  0              CA_RDT_ADR      = %EXPAND _(COM_RDT_ADR,        COM_RDT_ADR_) %,
;    0287  0              CA_RDT_SIZ      = %EXPAND _(COM_RDT_SIZ,        COM_RDT_SIZ_) %,
;    0288  0              CA_KFT_ADR      = %EXPAND _(COM_KFT_ADR,        COM_KFT_ADR_) %,
;    0289  0              CA_KFT_SIZ      = %EXPAND _(COM_KFT_SIZ,        COM_KFT_SIZ_) %,
;    0290  0              CA_CFT_ADR      = %EXPAND _(COM_CFT_ADR,        COM_CFT_ADR_) %,
;    0291  0              CA_CFT_SIZ      = %EXPAND _(COM_CFT_SIZ,        COM_CFT_SIZ_) %,
;    0292  0              CA_FDT_ADR      = %EXPAND _(COM_FDT_ADR,        COM_FDT_ADR_) %,
;    0293  0              CA_FDT_SIZ      = %EXPAND _(COM_FDT_SIZ,        COM_FDT_SIZ_) %,
;    0294  0              CA_TDT_ADR      = %EXPAND _(COM_TDT_ADR,        COM_TDT_ADR_) %,
```

```
;    0295  0              CA_TDT_SIZ      = %EXPAND _(COM_TDT_SIZ,        COM_TDT_SIZ_) %,
;    0296  0
; U  0297  0      %IF %BLISS(BLISS16) %THEN
; U  0298  0              CA_STAT_ADR     = %EXPAND _(0,                  COM_STAT_ADR_) %,       ! user error buffer (address)
; U  0299  0              CA_USR_WRN      = %EXPAND _(0,                  COM_USR_WRN_) %,        ! address of user-written warning ro
; U  0300  0              CA_1ST_SPC_ERR  = %EXPAND _(0,                  COM_1ST_SPC_ERR_) %,    ! first spec fatal error code
; U  0301  0              CA_1ST_SPC_LIN  = %EXPAND _(0,                  COM_1ST_SPC_LIN_) %,    ! first spec error line number
;    0302  0      %FI
;    0303  0              CA_CONST_AREA   = %EXPAND _(COM_CONST_AREA,     COM_CONST_AREA_) %,     ! constant area (address)
;    0304  0              CA_WRK_ADR      = %EXPAND _(COM_WRK_ADR,        COM_WRK_ADR_) %,        ! address of work area
;    0305  0              CA_WRK_END      = %EXPAND _(COM_WRK_END,        COM_WRK_END_) %,        ! address past end of work area
;    0306  0              CA_WF_NAMES     = %EXPAND _(COM_WF_NAMES,       COM_WF_NAMES_) %;       ! counted list of indices into CFT o
;    0307  0
;    0308  0      UNDECLARE %QUOTE _;
;    0309  0
;    0310  0      ! A macro to expand fields
;    0311  0      !
; L  0312  0      %IF %BLISS(BLISS32)
;    0313  0      %THEN   MACRO _(X,Y) = X %;
; U  0314  0      %ELSE   MACRO _(X,Y) = Y %;
;    0315  0      %FI
;    0316  0
;    0317  0      ! Values for datatypes
;    0318  0      ! A negative value indicates that the datatype is not supported
;    0319  0      !
;    0320  0      LITERAL
;    0321  0              DT_T    = _(DSC$K_DTYPE_T,      C$$),   ! Character (text)
;    0322  0              DT_AF   = _(-1,                 A$$),   ! Ascii Floating
;    0323  0              DT_AZ   = _(-1,                 Z$$),   ! Ascii Zoned
;    0324  0              DT_DB   = _(-1,                 L$$),   ! Dibol
;    0325  0              DT_F    = _(DSC$K_DTYPE_F,      F$$),   ! F-floating
;    0326  0              DT_D    = _(DSC$K_DTYPE_D,      F$$),   ! D-floating
;    0327  0              DT_G    = _(DSC$K_DTYPE_G,      -1),    ! G-floating
;    0328  0              DT_H    = _(DSC$K_DTYPE_H,      -1),    ! H-floating
;    0329  0              DT_P    = _(DSC$K_DTYPE_P,      P$$),   ! Packed decimal
;    0330  0              DT_B    = _(DSC$K_DTYPE_B,      B$$),   ! Signed binary
;    0331  0              DT_U    = _(DSC$K_DTYPE_BU,     U$$),   ! Unsigned binary
;    0332  0              DT_NU   = _(DSC$K_DTYPE_NU,     D$$),   ! Decimal unsigned
;    0333  0              DT_NL   = _(DSC$K_DTYPE_NL,     I$$),   ! Decimal leading separate
;    0334  0              DT_NLO  = _(DSC$K_DTYPE_NLO,    K$$),   ! Decimal leading overpunch
;    0335  0              DT_NR   = _(DSC$K_DTYPE_NR,     J$$),   ! Decimal trailing separate
;    0336  0              DT_NRO  = _(DSC$K_DTYPE_NRO,    D$$),   ! Decimal trailing overpunch
;    0337  0              DT_NZ   = _(DSC$K_DTYPE_NZ,     -1);    ! Zoned decimal
;    0338  0
;    0339  0      UNDECLARE %QUOTE _;
;    0340  0
;    0341  0      MACRO
;    0342  0              ! Macro to determine the length in bytes, given a KFT pointer
;    0343  0              ! Note that this is not needed after the spec file parser is called,
;    0344  0              ! since KFT_NDE_SIZ_ gives the same information.
;    0345  0              !
; M  0346  0              KFT_UNITS_(KFT_PTR) =
; M  0347  0                  BEGIN
; M  0348  0                  LOCAL
; M  0349  0                      FDT_IX;
; M  0350  0                  FDT_IX = .KFT_PTR[0,KFT_FDT_IDX];
; M  0351  0                  IF .KFT_PTR[0,KFT_CONSTANT]
```

```
;  M 0352  0                    THEN
;  M 0353  0                        .CFT[.FDT_IX, CFT_CON_LEN]
;  M 0354  0                    ELSE
;  M 0355  0                        %IF %BLISS(BLISS32) %THEN
;  M 0356  0                        IF .FDT[.FDT_IX, FDT_TYPE] EQL DT_P
;  M 0357  0                        THEN
;  M 0358  0                            .FDT[.FDT_IX, FDT_FLD_SIZ]/2 + 1    ! Length in bytes
;  M 0359  0                        ELSE
;  M 0360  0                        %FI
;  M 0361  0                            .FDT[.FDT_IX, FDT_FLD_SIZ]
;    0362  0                    END %;
;    0363  0
;  L 0364  0        %IF %BLISS(BLISS32)
;    0365  0        %THEN
;    0366  0
;    0367  0        ! Character codes
;    0368  0        !
;    0369  0        LITERAL
;    0370  0            C_LBRACK    = %X'5B',       ! Character "["
;    0371  0            C_RBRACK    = %X'5D',       ! Character "]"
;    0372  0            C_SLASH     = %X'2F',       ! Character "/"
;    0373  0            C_EXCLAM    = %X'21',       ! Character "!"
;    0374  0            C_PERCENT   = %X'25',       ! Character "%"
;    0375  0            C_COMMA     = %X'2C',       ! Character ","
;    0376  0            C_NULL      = %X'00',       ! Character NUL
;    0377  0            C_QUOTE     = %X'22',       ! Character """
;    0378  0            C_L_PAREN   = %X'28',       ! Character "("
;    0379  0            C_R_PAREN   = %X'29',       ! Character ")"
;    0380  0            C_COLON     = %X'3A',       ! Character ":"
;    0381  0            C_EQUAL     = %X'3D',       ! Character "="
;    0382  0            C_LESS      = %X'3C',       ! Character "<"
;    0383  0            C_GREATER   = %X'3E',       ! Character ">"
;    0384  0            C_DASH      = %X'2D',       ! Character "-"
;    0385  0            C_SPACE     = %X'20',       ! Character " "
;    0386  0            C_TAB       = %X'09',       ! Character HT
;    0387  0            C_CR        = %X'0D',       ! Character CR
;    0388  0            C_LF        = %X'0A';       ! Character LF
;    0389  0        %FI
;    0390  0        LITERAL
;    0391  0            C_OCT       = %X'6F',       ! Lower case "o", for octal number base
;    0392  0            C_DEC       = %X'64',       ! Lower case "d", for decimal number base
;    0393  0            C_HEX       = %X'78';       ! Lower case "x", for hexadecimal number base
```

```
;  U 0394  0     %IF %BLISS(BLISS16) %THEN
;  U 0395  0     !----------------------------------------------------------
;  U 0396  0     !                KEY COMPARISON TABLE (KCT)
;  U 0397  0     !
;  U 0398  0     !       This table is used by Sort-11 for fast access to the
;  U 0399  0     !       key descriptions of keys that need to be compared.
;  U 0400  0     !
;  U 0401  0     !----------------------------------------------------------
;  U 0402  0     LITERAL
;  U 0403  0         KCT_MAX = 64,                   !maximum number of entries in KCT
;  U 0404  0         KCT_UNIT = 8;                   !size in bytes
;  U 0405  0
;  U 0406  0     STRUCTURE
;  U 0407  0         KCT_TAB[ O,B,P,S,E; BS ] =
;  U 0408  0             [ BS*KCT_UNIT ]
;  U 0409  0             ( KCT_TAB + O*KCT_UNIT + B )<P,S,E>;
;  U 0410  0
;  U 0411  0     MACRO
;  U 0412  0         KCT_CMP_ADR_    =  0, 0, 16, 0 %,       !address of comparison routine
;  U 0413  0         KCT_KEY_POS_    =  2, 0, 16, 0 %,       !starting position of key field
;  U 0414  0         KCT_KEY_LEN_    =  4, 0, 16, 0 %,       !length of key field
;  U 0415  0         KCT_CONTINUE_   =  6, 0,  1, 0 %,       !continue word
;  U 0416  0         KCT_DESCEND_    =  6, 1,  1, 0 %,       !descend = 1, ascend = 0
;  U 0417  0         KCT_TYPE_       =  7, 0,  8, 0 %;       !data type, used to reinitialize
;     0418  0     %FI
```

```
;    0419  0      ! WHILE_FAIL
;    0420  0      !       This macro produces code that advances a table pointer through
;    0421  0      !       successive entries until the entry is unconditional, or the
;    0422  0      !       entry is conditional and passes the condition.
;    0423  0      !       The parameter to this macro (X) is the identification of the table.
;    0424  0      !       The table pointer must be of the form (X)_PTR, and the table must
;    0425  0      !       have the following fields: (X)_CONDX and (X)_TDT_IDX.
;    0426  0      !
;    0427  0      MACRO
;M   0428  0          WHILE_FAIL_(X) =
;M   0429  0              BEGIN
;M   0430  0              MACRO
;M   0431  0                  X_PTR       = %NAME(X,'_PTR') %QUOTE %,
;M   0432  0                  X_CONDX     = %NAME(X,'_CONDX') %QUOTE %,
;M   0433  0                  X_TDT_IDX   = %NAME(X,'_TDT_IDX') %QUOTE %;
;M   0434  0              !
;M   0435  0              ! While we fail conditional tests
;M   0436  0              !
;M   0437  0              WHILE 1 DO
;M   0438  0                  BEGIN
;M   0439  0                  LOCAL
;M   0440  0                      PASS;
;M   0441  0
;M   0442  0                  ! Unconditional tests are easy
;M   0443  0                  !
;M   0444  0                  IF NOT .X_PTR[0, X_CONDX] THEN EXITLOOP;
;M   0445  0
;M   0446  0                  ! We have a condition
;M   0447  0                  !
;M   0448  0                  PASS = %IF %BLISS(BLISS32) %THEN SOR$$TDT %ELSE $TDT %FI (
;M   0449  0                      INPREC[0],                  ! Length/address of record
;M   0450  0                      TDT[.X_PTR[0,X_TDT_IDX],BASE_]  ! Address of TDT tests
;M   0451  0                      );
;M   0452  0                  IF .PASS GTRU 1 THEN RETURN .PASS;  ! Unexpected result
;M   0453  0                  IF .PASS EQLU 1 THEN EXITLOOP;      ! We passed the test!
;M   0454  0
;M   0455  0                  ! Advance to the next record definition
;M   0456  0                  !
;M   0457  0                  X_PTR = X_PTR[1,BASE_];
;M   0458  0                  END;
;    0459  0              END %;
```

```
;  U 0460  0     %IF %BLISS(BLISS16) %THEN
;  U 0461  0     !
;  U 0462  0     ! Other Sort-11 modules that use the fields defined herein
;  U 0463  0     ! like to see underscores at the ends of the names.
;  U 0464  0     !
;  U 0465  0     MACRO   _(X) = X = %quote %expand %REMAINING %QUOTE % %;
;  U 0466  0     MACRO
;  U 0467  0             _(FDT_TYPE_,     FDT_TYPE),
;  U 0468  0             _(FDT_FLD_POS_, FDT_FLD_POS),
;  U 0469  0             _(FDT_FLD_SIZ_, FDT_FLD_SIZ),
;  U 0470  0             _(KFT_NDE_POS_, KFT_NDE_POS),
;  U 0471  0             _(KFT_NDE_SIZ_, KFT_NDE_SIZ),
;  U 0472  0             _(KFT_CONTINUE_,KFT_CONTINUE),
;  U 0473  0             _(KFT_CONSTANT_,KFT_CONSTANT),
;  U 0474  0             _(KFT_CONT_CDX_,KFT_CONT_CDX),
;  U 0475  0             _(KFT_CONDX_,    KFT_CONDX),
;  U 0476  0             _(KFT_BUILD_,    KFT_BUILD),
;  U 0477  0             _(KFT_DESCEND_, KFT_DESCEND),
;  U 0478  0             _(KFT_DATA_,     KFT_DATA),
;  U 0479  0             _(KFT_FDT_IDX_, KFT_FDT_IDX),
;  U 0480  0             _(KFT_TDT_IDX_, KFT_TDT_IDX),
;  U 0481  0             _(RDT_INCLUDE_, RDT_INCLUDE),
;  U 0482  0             _(RDT_CONDX_,    RDT_CONDX),
;  U 0483  0             _(RDT_TDT_IDX_, RDT_TDT_IDX),
;  U 0484  0             _(RDT_KCT_ADR_, RDT_KCT_ADR),
;  U 0485  0             _(RDT_KFT_IDX_, RDT_KFT_IDX);
;  U 0486  0     UNDECLARE %QUOTE _;
;    0487  0     %FI
```

```
;    0488  0     !
;    0489  0     !       Check that the fields are large enough
;    0490  0     MACRO
;    0491  0         S_[O,P,S,E] = 1^S %,
;  M 0492  0         M_(V,O,P,S,E)[] = %IF V GTRU MINU(1^S-1,S_(%REMAINING)) %THEN
;    0493  0             %WARN(V,' is too large') %FI %;
;    0494  0
;    0495  0     M_(FDT_MAX, CA_FDT_SIZ, TDT_FLD_ONE, TDT_FLD_TWO, KFT_FDT_IDX)
;    0496  0     M_(TDT_MAX, CA_TDT_SIZ, KFT_TDT_IDX, RDT_TDT_IDX)
;    0497  0     M_(KFT_MAX, CA_KFT_SIZ, RDT_KFT_IDX)
;    0498  0     M_(RDT_MAX, CA_RDT_SIZ, 0,0,8,0)
;    0499  0     M_(CFT_MAX, CA_CFT_SIZ, KFT_FDT_IDX)
;    0500  0
;    0501  0     UNDECLARE %QUOTE S_, %QUOTE M_;
;    0502  0
;    0503  0     !----------------------------------------------------------
;    0504  0     ! End of SRTSPC.REQ


;                              Library Statistics
;                                        --------- Symbols ---------      Pages       Processing
;       File                             Total     Loaded   Percent      Mapped      Time
;  _$255$DUA28:[SYSLIB]STARLET.L32;1      9776         14         0         581        00:01.0
;  _$255$DUA28:[SORT32.SRC]SORLIB.L32;1    409        100        24          34        00:00.4



;                                        COMMAND QUALIFIERS

;       BLISS SRC$:SRTSPC/LIS=LIS$:SRTSPC/LIB=SRC$:SRTSPC

; Run Time:          00:06.5
; Elapsed Time:      00:26.3
; Lines/CPU Min:      4623
; Lexemes/CPU-Min:  30036
; Memory Used:  71 pages
; Library Precompilation Complete
```

SRTSPCBLI
LIS

SORSPEC
LIS

SORTRNVEC
LIS

SORSORT
LIS

SORSPCUTI
LIS

SRTSPC
LIS