

```

SSSSSSSSSSSS 00000000 RRRRRRRRRR TTTTTTTTTT 33333333 22222222
SSSSSSSSSSSS 00000000 RRRRRRRRRR TTTTTTTTTT 33333333 22222222
SSSSSSSSSSSS 00000000 RRRRRRRRRR TTTTTTTTTT 33333333 22222222
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      222
SSS          000      000  RRR      RRR  TTT          333      222
SSS          000      000  RRR      RRR  TTT          333      222
SSSSSSSSSS 000      000  RRRRRRRRRR TTT          333      222
SSSSSSSSSS 000      000  RRRRRRRRRR TTT          333      222
SSSSSSSSSS 000      000  RRRRRRRRRR 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      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
SSSSSSSSSS 00000000 RRR      RRR  TTT          33333333 22222222
SSSSSSSSSS 00000000 RRR      RRR  TTT          33333333 22222222
SSSSSSSSSS 00000000 RRR      RRR  TTT          33333333 22222222

```


File: SRTSPC.REQ IDENT = 'V04-000' ! File: SRTSPC.REQ Edit: PDG3028

```

*****
*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****

```

```

**
FACILITY:      VAX-11 SORT/MERGE, PDP-11 SORT/MERGE
ABSTRACT:
  This require file is for data structures returned from specification
  file processing.
  This file is used as a library source.
ENVIRONMENT:   VAX/VMS user mode
AUTHOR: V. Bennison, CREATION DATE: 03-May-1982
MODIFIED BY:
  31-Aug-1982   PDG
  Add definitions that are required for SRTSPC.BLI.
T03-016 Rework TDT table to give precedence to AND/OR. PDG 13-Dec-1982
T03-017 Add WF_NAMES, CFT indices of work file names. PDG 26-Dec-1982
T03-018 Removed RDT VAR. PDG 3-Jan-1983
T03-019 Removed PT/ST_ADR; added WRK_SIZ, BS_DECM. PDG 26-Jan-1983
T03-020 Add FDT_SCALE and CA_PAD. PDG 8-Feb-1983
T03-022 Fix computation for packed in KFT_UNITS. PDG 11-Feb-1983
T03-022 Remove unreferenced fields. PDG 16-Mar-1983
T03-024 Work around Bliss bug with CA_LINKAGE_LB. PDG 12-May-1983
T03-025 Define KFT_NDE_SIZ for BLISST6. PDG 26-Jul-1983
T03-026 Put WHILE_FAIL here. PDG 1-Aug-1983
T03-027 Word-align elements in CON_SYM_TAB. PDG 1-Aug-1983

```

SR

XIF

MAC
MAC

UNC
XF]

!-
!- T03-028 Make sharing of code easier to maintain. PDG 31-Jan-1984

SR
!
MA
M
M
M
M
UNC
!
!

FIELD DEFINITION TABLE (FDT)

LITERAL

```
FDT_MAX = 255,           !maximum number of entries in FDT
FDT_UNIT = 6;           ! Size in bytes, must be even for bliss16
                        ! because of 16 bit field
```

STRUCTURE

```
FDT_TAB[ O,B,P,S,E; BS ] =
  [ BS*FDT_UNIT ]
  ( FDT_TAB + O*FDT_UNIT + B )<P,S,E>;
```

MACRO

```
FDT_TYPE    = 0, 0, 8, 0 %;    ! Data type
FDT_SCALE   = 1, 0, 8, 1 %;    ! Scale factor
FDT_FLD_POS = 2, 0, 16, 0 %;   ! Position of field
FDT_FLD_SIZ = 4, 0, 16, 0 %;   ! Size of field
```

```

-----
TEST DEFINITION TABLE (TDT)
-----

```

LITERAL

```

TDT_MAX = 255,           !maximum number of entries in TDT
TDT_UNIT = 4;           ! Size in bytes

```

STRUCTURE

```

TDT_TAB( 0,B,P,S,E; BS ) =
  [ BS*TDT_UNIT ]
  ( TDT_TAB + 0*TDT_UNIT + B ) <P,S,E>;

```

MACRO

```

TDT_TRUE   = 0, 0, 1, 0 %;   ! Set to simply return TRUE
TDT_CMP    = 0, 1, 3, 0 %;   ! The comparison flags
TDT_EQL    = 0, 1, 1, 0 %;   ! True if "Equal to" succeeds
TDT_LSS    = 0, 2, 1, 0 %;   ! True if "Less than" succeeds
TDT_GTR    = 0, 3, 1, 0 %;   ! True if "Greater than" succeeds
TDT_CONSTANT = 0, 4, 1, 0 %; ! True iff FLD_TWO is to CFT
TDT_FLD_ONE = 1, 0, 8, 0 %;   ! Index in FDT of 1st field
TDT_FLD_TWO = 2, 0, 8, 0 %;   ! Index in FDT (or CFT) of 2nd field
TDT_GOTO   = 3, 0, 8, 0 %;   ! TDT index adjustment

```

```

! This structure should only be referenced by the routines that builds it,
! and the routine SOR$TDT!

```

```

! This table is used as follows:

```

```

      Set IX to the index of the test description to test
Loop:  If TDT_TRUE is clear then return TRUE
      If the comparison between FLD_ONE and FLD_TWO is true
      (according to the EQL/LSS/GTR bits)
      then
        if TDT_GOTO is zero then return false else add TDT_GOTO to IX
      else
        add 1 to IX
      goto Loop

```

```

-----
KEY/DATA FIELD TABLE (KFT)
-----

```

LITERAL

```

KFT_MAX = 255,           !maximum number of entries in KFT
KFT_UNIT = 8;           ! Size in bytes, must be even for bliss16
                        ! because of 16 bit field

```

STRUCTURE

```

KFT_TAB[ 0,B,P,S,E; BS ] =
  [ BS*KFT_UNIT ]
  ( KFT_TAB + 0*KFT_UNIT + B )<P,S,E>;

```

MACRO

```

KFT_NDE_POS = 0, 0, 16, 0 %;           ! Starting position in node

KFT_CONTINUE = 3, 0, 1, 0 %;           ! Continue = 1
KFT_CONSTANT = 3, 1, 1, 0 %;           ! True iff FDT_IDX is to CFT
KFT_CONT_CDX = 3, 2, 1, 0 %;           ! Continued condition = 1
KFT_CONDX = 3, 3, 1, 0 %;             ! Conditional field = 1
KFT_BUILD = 3, 4, 1, 0 %;             ! Build the key = 1
KFT_DESCEND = 3, 5, 1, 0 %;           ! Asc/desc, descend = 1
KFT_DATA = 3, 6, 1, 0 %;             ! Key or data, data = 1

KFT_FDT_IDX = 4, 0, 8, 0 %;           ! Index in FDT (or CFT)
KFT_TDT_IDX = 5, 0, 8, 0 %;           ! TDT index for forces

KFT_NDE_SIZ = 6, 0, 16, 0 %;         ! Size (bytes) in internal node

```

RECORD DEFINITION TABLE (RDT)

LITERAL

```
RDT_MAX = 64,           !maximum number of entries in RDT
RDT_UNIT = 6;          ! Size in bytes
```

STRUCTURE

```
RDT_TAB[ O,B,P,S,E; BS ] =
  [ BS*RDT_UNIT ]
  ( RDT_TAB + O*RDT_UNIT + B ) <P,S,E>;
```

MACRO

```
RDT_INCLUDE = 0, 0, 1, 0 %;    ! Include/omit, Include = 1
RDT_CONDX   = 0, 1, 1, 0 %;    ! Conditional = 1

RDT_TDT_IDX = 1, 0, 8, 0 %;    ! Index into TDT
RDT_KCT_ADR = 2, 0, 16, 0 %;   ! For Sort-11 only
RDT_KFT_IDX = 4, 0, 8, 0 %;   ! Index into KFT
```

```
! The RDT table is scanned sequentially until either an unconditional entry is
! found, or until a condition (via RDT_TDT_IDX) passes. This matched entry
! describes whether to omit or include the record (RDT_INCLUDE). If included,
! then RDT_KFT_IDX is used to index the KFT table, for record reformatting.
```

CONSTANT FIELD TABLE (CFT)

LITERAL

```
CFT_MAX = 255,           ! maximum number of entries in CFT
CFT_UNIT = 2+BPADDR/8;  ! Size in bytes
```

STRUCTURE

```
CFT_TAB[ 0, B, P, S, E; BS ] =
  [ BS*CFT_UNIT ]
  ( CFT_TAB + 0*CFT_UNIT + B )<P,S,E>;
```

MACRO

```
CFT_CON_LEN = 0, 0, 8, 0 %;           ! Length of constant
CFT_CON_ADR = 2, 0, BPADDR, 0 %;      ! Address of constant
```

```

-----
COMMON DEFINITIONS
-----

```

```

%IF %BLISS(BLISS32)
%THEN
    LIBRARY 'SYS$LIBRARY:STARLET';
    LIBRARY 'SRC$:SORLIB';
%ELSE
    LIBRARY 'S11V3SRC:SMCOM';
%FI

! Define the linkage to the common routines
!
LITERAL
    LB_REG = 4;
LINKAGE
    CA_LINKAGE =
        %BLISS32( CALL:GLOBAL(CA=COM_REG_CTX) ) ! MUST BE SAME AS CAL_CTXREG!
        %BLISS16( JSR ),
    CA_LINKAGE_LB =
        ! Same as CA_LINKAGE, with an extra register
        %BLISS32( CALL:GLOBAL(CA=COM_REG_CTX, LB=LB_REG) )
        %BLISS16( JSR :GLOBAL( LB=LB_REG) ),
    CA_LINK_SEGMENT =
        %BLISS16( JSR )
        %BLISS32( JSB (
            REGISTER=6,
            REGISTER=COM_REG_SRC2):
            GLOBAL(CA=COM_REG_CTX)
            PRESERVE(COM_REG_SRC2)
            NOTUSED(7,8,9)
            NOPRESERVE(0,1,2,3,4,5));

! A macro to declare/get the address of the common area
!
MACRO
    CA_AREA ( X ) =
        %IF %BLISS(BLISS32)
        %THEN
            EXTERNAL REGISTER
            %IF %NULL(X) %THEN CA %ELSE X %FI
            = COM_REG_CTX: REF BLOCK[CTX_K_SIZE]
            FIELD(CTX_FIELDS);
        %ELSE
            %IF NOT %NULL(X)
            %THEN
                LOCAL
                X : REF BLOCK [, %UPVAL] FIELD (COM_FIELDS);
                %QUOTE GET_IMPAREA_( X );
            %FI
        %FI X;

! Specification file error messages
!

```

```

!LITERAL
SRTIWA = SORS_SRTIWA,      !insufficient work area
SPCOVR = SORS_SPCOVR,     !warning: overridden specification
SPCMIS = SORS_SPCMIS,     !warning: invalid merge specification
SPCSIS = SORS_SPCISIS,    !warning: invalid sort specification
SPCIVP = SORS_SPCIVP,     !invalid sort process
SPCIVS = SORS_SPCIVS,     !invalid specification
SPCIVC = SORS_SPCIVC,     !invalid collating sequence specification
SPCIVF = SORS_SPCIVF,     !invalid field specification
SPCIVD = SORS_SPCIVD,     !invalid data type
SPCIVX = SORS_SPCIVX,     !invalid condition specification
SPCIVK = SORS_SPCIVK,     !invalid key or data specification
SPCIVI = SORS_SPCIVI;     !invalid include or omit specification

```

```
! A macro to expand fields
```

```

%IF %BLISS(BLISS32)
%THEN  MACRO  _(X,Y) = %QUOTE %EXPAND %FIELDEXPAND(X) %;
%ELSE  MACRO  _(X,Y) = %QUOTE %EXPAND %FIELDEXPAND(Y) %;
%FI

```

```
MACRO
```

```
! Sort/Merge process information
```

```

CA_PROCESS      = %EXPAND  (COM_SORT_TYPE,      COM_PROCESS ) %,
CA_PROCESS_OVR  = %EXPAND  (COM_OVR_PROC,      COM_PROC_OVR ) %,
CA_VAR_MERGE    = %EXPAND  (COM_MERGE,        COM_MERGE_) %,

```

```
! Collating information
```

```

CA_TIE_BREAK    = %EXPAND  (COM_TIE_BREAK,      COM_TIE_BREAK ) %,
CA_ST_ADR       = %EXPAND  (COM_COLLATE,        COM_CS_TAB_ADR ) %,
CA_ST_SIZ       = %EXPAND  (COM_ST_SIZ,        COM_CS_TAB_SIZ ) %,
CA_BS_DECM      = %EXPAND  (COM_BS_DECM,       COM_BS_DECM ) %,
CA_PAD          = %EXPAND  (COM_PAD,          COM_PAD_CHAR_) %,

```

```
! Keys and stable information
```

```

CA_KEY_OVR      = %EXPAND  (COM_OVR_KEY,        COM_KEY_OVR ) %,
CA_CHKSEQ       = %EXPAND  (COM_SEQ_CHECK,      COM_CH_SEQ ) %,
CA_CHKSEQ_OVR   = %EXPAND  (COM_SEQ_CHECK,      COM_CHSEQ_OVR ) %,
CA_STABLE       = %EXPAND  (COM_STABLE,        COM_STABLE ) %,
CA_STABLE_OVR   = %EXPAND  (COM_STABLE,        COM_STBL_OVR ) %,
CA_COLSEQ_OVR   = %EXPAND  (COM_OVR_COLSEQ,     COM_CSEQ_OVR_) %,

```

```
! Record reformatting, and other tables
```

```

CA_RDT_ADR      = %EXPAND  (COM_RDT_ADR,        COM_RDT_ADR ) %,
CA_RDT_SIZ      = %EXPAND  (COM_RDT_SIZ,        COM_RDT_SIZ_) %,
CA_KFT_ADR      = %EXPAND  (COM_KFT_ADR,        COM_KFT_ADR_) %,
CA_KFT_SIZ      = %EXPAND  (COM_KFT_SIZ,        COM_KFT_SIZ_) %,
CA_CFT_ADR      = %EXPAND  (COM_CFT_ADR,        COM_CFT_ADR_) %,
CA_CFT_SIZ      = %EXPAND  (COM_CFT_SIZ,        COM_CFT_SIZ_) %,
CA_FDT_ADR      = %EXPAND  (COM_FDT_ADR,        COM_FDT_ADR_) %,

```

```

CA_FDT_SIZ      = %EXPAND (COM_FDT_SIZ,      COM_FDT_SIZ ) %,
CA_TDT_ADR      = %EXPAND (COM_TDT_ADR,      COM_TDT_ADR ) %,
CA_TDT_SIZ      = %EXPAND (COM_TDT_SIZ,      COM_TDT_SIZ ) %,

%IF %BLISS(BLISS16) %THEN
CA_STAT_ADR     = %EXPAND (0,                COM_STAT_ADR ) %,      ! user error buffer (address)
CA_USR_WRN      = %EXPAND (0,                COM_USR_WRN ) %,      ! address of user-written warning routine
CA_1ST_SPC_ERR  = %EXPAND (0,                COM_1ST_SPC_ERR ) %,  ! first spec fatal error code
CA_1ST_SPC_LIN  = %EXPAND (0,                COM_1ST_SPC_LIN ) %,  ! first spec error line number

%FI
CA_CONST_AREA   = %EXPAND (COM_CONST_AREA,   COM_CONST_AREA ) %,    ! constant area (address)
CA_WRK_ADR      = %EXPAND (COM_WRK_ADR,      COM_WRK_ADR ) %,      ! address of work area
CA_WRK_END      = %EXPAND (COM_WRK_END,      COM_WRK_END ) %,      ! address past end of work area
CA_WF_NAMES     = %EXPAND (COM_WF_NAMES,     COM_WF_NAMES ) %,     ! counted list of indices into CFT of work file name

```

```
UNDECLARE %QUOTE _;
```

```
! A macro to expand fields
```

```

%IF %BLISS(BLISS32)
%THEN  MACRO (X,Y) = X %;
%ELSE  MACRO (X,Y) = Y %;
%FI

```

```
! Values for datatypes
```

```
! A negative value indicates that the datatype is not supported
```

```
LITERAL
```

```

DT_T      = (DSCSK_DTYPE_T,    $$$), ! Character (text)
DT_AF     = (-1,              ASS), ! Ascii Floating
DT_AZ     = (-1,              Z$$), ! Ascii Zoned
DT_DB     = (-1,              L$$), ! Dibol
DT_F      = (DSCSK_DTYPE_F,    F$$), ! F-floating
DT_D      = (DSCSK_DTYPE_D,    D$$), ! D-floating
DT_G      = (-1,              G$$), ! G-floating
DT_H      = (-1,              H$$), ! H-floating
DT_P      = (DSCSK_DTYPE_P,    P$$), ! Packed decimal
DT_B      = (DSCSK_DTYPE_B,    B$$), ! Signed binary
DT_U      = (DSCSK_DTYPE_BU,   U$$), ! Unsigned binary
DT_NU     = (DSCSK_DTYPE_NU,   D$$), ! Decimal unsigned
DT_NL     = (DSCSK_DTYPE_NL,   I$$), ! Decimal leading separate
DT_NLO    = (DSCSK_DTYPE_NLO,  K$$), ! Decimal leading overpunch
DT_NR     = (DSCSK_DTYPE_NR,   J$$), ! Decimal trailing separate
DT_NRO    = (DSCSK_DTYPE_NRO,  D$$), ! Decimal trailing overpunch
DT_NZ     = (DSCSK_DTYPE_NZ,   -1); ! Zoned decimal

```

```
UNDECLARE %QUOTE _;
```

```
MACRO
```

```

! Macro to determine the length in bytes, given a KFT pointer
! Note that this is not needed after the spec file parser is called,
! since KFT_NDE_SIZ_ gives the same information.

```

```

KFT_UNITS_(KFT_PTR) =
  BEGIN
  LOCAL

```

```

      FDT_IX;
      FDT_IX = .KFT_PTR[0,KFT_FDT_IDX];
      IF .KFT_PTR[0,KFT_CONSTANT]
      THEN
        .CFT[.FDT_IX, CFT_CON_LEN]
      ELSE
        %IF %BLISS(BLISS32) %THEN
          IF .FDT[.FDT_IX, FDT_TYPE] EQL DT_P
          THEN
            .FDT[.FDT_IX, FDT_FLD_SIZ]/2 + 1    ! Length in bytes
          ELSE
            %FI
            .FDT[.FDT_IX, FDT_FLD_SIZ]
          END %;

```

```

%IF %BLISS(BLISS32)
%THEN

```

```

! Character codes

```

```

LITERAL

```

```

C_LBRACK      = %X'5B',      ! Character '['
C_RBRACK      = %X'5D',      ! Character ']'
C_SLASH       = %X'2F',      ! Character '/'
C_EXCLAM     = %X'21',      ! Character '!'
C_PERCENT     = %X'25',      ! Character '%'
C_COMMA      = %X'2C',      ! Character ','
C_NULL       = %X'00',      ! Character ''
C_QUOTE      = %X'22',      ! Character '"'
C_L_PAREN    = %X'28',      ! Character '('
C_R_PAREN    = %X'29',      ! Character ')'
C_COLON      = %X'3A',      ! Character ':'
C_EQUAL      = %X'3D',      ! Character '='
C_LESS       = %X'3C',      ! Character '<'
C_GREATER    = %X'3E',      ! Character '>'
C_DASH       = %X'2D',      ! Character '-'
C_SPACE      = %X'20',      ! Character ' '
C_TAB        = %X'09',      ! Character HT
C_CR         = %X'0D',      ! Character CR
C_LF         = %X'0A',      ! Character LF

```

```

%FI

```

```

LITERAL

```

```

C_OCT        = %X'6F',      ! Lower case 'o', for octal number base
C_DEC        = %X'64',      ! Lower case 'd', for decimal number base
C_HEX        = %X'78',      ! Lower case 'x', for hexadecimal number base

```

```
%IF %BLISS(BLISS16) %THEN
```

```
-----  
KEY COMPARISON TABLE (KCT)
```

```
This table is used by Sort-11 for fast access to the  
key descriptions of keys that need to be compared.
```

```
-----  
LITERAL
```

```
KCT_MAX = 64,           !maximum number of entries in KCT  
KCT_UNIT = 8;          !size in bytes
```

```
STRUCTURE
```

```
KCT_TAB( 0,B,P,S,E; BS ) =  
  [ BS*KCT_UNIT ]  
  ( KCT_TAB + 0*KCT_UNIT + B ) <P,S,E>;
```

```
MACRO
```

```
KCT_CMP_ADR_ = 0, 0, 16, 0 %;   !address of comparison routine  
KCT_KEY_POS_ = 2, 0, 16, 0 %;   !starting position of key field  
KCT_KEY_LEN_ = 4, 0, 16, 0 %;   !length of key field  
KCT_CONTINUE_ = 6, 0, 1, 0 %;   !continue word  
KCT_DESCEND_ = 6, 1, 1, 0 %;    !descend = 1, ascend = 0  
KCT_TYPE_    = 7, 0, 8, 0 %;    !data type, used to reinitialize
```

```
%FI
```

```

! WHILE_FAIL
! This macro produces code that advances a table pointer through
! successive entries until the entry is unconditional, or the
! entry is conditional and passes the condition.
! The parameter to this macro (X) is the identification of the table.
! The table pointer must be of the form (X) PTR, and the table must
! have the following fields: (X)_CONDX and (X)_TDT_IDX.
MACRO
WHILE_FAIL_(X) =
  BEGIN
  MACRO
    X_PTR      = %NAME(X, '_PTR') %QUOTE %,
    X_CONDX    = %NAME(X, '_CONDX') %QUOTE %,
    X_TDT_IDX  = %NAME(X, '_TDT_IDX') %QUOTE %;
  ! While we fail conditional tests
  WHILE 1 DO
    BEGIN
    LOCAL
      PASS;
    ! Unconditional tests are easy
    IF NOT .X_PTR[0, X_CONDX] THEN EXITLOOP;
    ! We have a condition
    PASS = %IF %BLISS(BLISS32) %THEN SORS$TDT %ELSE $TDT %FI (
      INPREC[0],           ! Length/address of record
      TDT[X_PTR[0, X_TDT_IDX], BASE_] ! Address of TDT tests
    );
    IF .PASS GTRU 1 THEN RETURN .PASS; ! Unexpected result
    IF .PASS EQLU 1 THEN EXITLOOP;    ! We passed the test!
    ! Advance to the next record definition
    X_PTR = X_PTR[1, BASE_];
  END;
END %;

```

```
%IF %BLISS(BLISS16) %THEN
```

```
Other Sort-11 modules that use the fields defined herein  
like to see underscores at the ends of the names.
```

```
MACRO _ (X) = X = %quote %expand %REMAINING %QUOTE % %;
```

```
MACRO
```

```
(FDT_TYPE, FDT_TYPE),  
-(FDT_FLD_POS, FDT_FLD_POS),  
-(FDT_FLD_SIZ, FDT_FLD_SIZ),  
-(KFT_NDE_POS, KFT_NDE_POS),  
-(KFT_NDE_SIZ, KFT_NDE_SIZ),  
-(KFT_CONTINUE, KFT_CONTINUE),  
-(KFT_CONSTANT, KFT_CONSTANT),  
-(KFT_CONT_CDX, KFT_CONT_CDX),  
-(KFT_CONDX, KFT_CONDX),  
-(KFT_BUILD, KFT_BUILD),  
-(KFT_DESCEND, KFT_DESCEND),  
-(KFT_DATA, KFT_DATA),  
-(KFT_FDT_IDX, KFT_FDT_IDX),  
-(KFT_TDT_IDX, KFT_TDT_IDX),  
-(RDT_INCLUDE, RDT_INCLUDE),  
-(RDT_CONDX, RDT_CONDX),  
-(RDT_TDT_IDX, RDT_TDT_IDX),  
-(RDT_KCT_ADR, RDT_KCT_ADR),  
-(RDT_KFT_IDX, RDT_KFT_IDX);
```

```
UNDECLARE %QUOTE _;
```

```
%FI
```



```
!
! Check that the fields are large enough
MACRO
  S_[O,P,S,E] = 1^S %
  M_(V,O,P,S,E)[ ] = %IF V GTRU MINU(1^S-1,S_(%REMAINING)) %THEN
    %WARN(V,' is too large') %FI %;

M_(FDT_MAX, CA_FDT_SIZ, TDT_FLD_ONE, TDT_FLD_TWO, KFT_FDT_IDX)
M_(TDT_MAX, CA_TDT_SIZ, KFT_TDT_IDX, RDT_TDT_IDX)
M_(KFT_MAX, CA_KFT_SIZ, RDT_KFT_IDX)
M_(RDT_MAX, CA_RDT_SIZ, 0,0,8,0)
M_(CFT_MAX, CA_CFT_SIZ, KFT_FDT_IDX)

UNDECLARE %QUOTE S_, %QUOTE M_;

!-----
! End of SRTSPC.REQ
```

