1.2

Table of contents

```
                        .TITLE  SORTI   SORT INITIALIZATION
                        .ENABL  LC
                ;
                ;  Copyright (c) 1977,1978,1979,1980.
                ;  S&H Computer Systems, Inc.
                ;  Nashville, Tennessee
                ;
000000                  .PSECT  SORTID,D         ;DATA SECTION PSECT
                ;
                ;  Written by Phil Sherrod.
                ;
                ;
                ;  THIS MODULE OF SORT IS THE INTERFACE TO
                ;  THE CALLING PROGRAM OR USER.
                ;  IT ALLOWS SORT TO BE RUN IN ANY OF THREE MODES:
                ;   1) DIRECT RUN
                ;   2) CHAIN FROM ANOTHER PROGRAM
                ;   3) TSX MESSAGE COMMUNICATION
                ;
                ;
                ;###############################################################
                ;
                ; Modified in Feburary, 1980 by K.C. Lidster of DISC to also execute
                ; under the RT11 run-time for the RSTS operating system.  This mainly
                ; involved the handling of the extra file parameters pertaining to
                ; RSTS (PPN, clustersize, etc.).
                ;
                ;###############################################################
                ;
                ;
                ;  GLOBAL DEFINITIONS
                ;
                        .GLOBL  SRTINI,TERM,FT$D,SO$NOT
                        .GLOBL  S$LINK,S$OPR,S$TYP,S$BR,S$FLG
                        .GLOBL  S$POS1,S$SIZ1,S$POS2,S$$SZ,SO$CMP
                        .GLOBL  SO$AND,SO$OR,FT$T,SF$LIT,M$POS1
                        .GLOBL  M$POS2,M$SIZE,M$$SZ,BT$HI,BT$LO
                        .GLOBL  BT$REV
                ;
                ;  GLOBAL REFERENCES
                ;
                        .GLOBL  QBLK,MODE,TSXFL,FSTIME
                        .GLOBL  KTYPE,KAD,KSTRT,KLEN
                        .GLOBL  TOPMEM,NUMKY2,RECLEN,SKPCNT,MXREC1
                        .GLOBL  MXREC2,CINSPC,EOFCHR,EOFCOL
                        .GLOBL  T1SIZ,T2SIZ,ISMARS
                        .GLOBL  NUMCOL,NUMLEN,NUMBUF
                        .GLOBL  ITYPE,OTYPE,RT$CBL,RT$DBL
                        .GLOBL  RT$FTN,RT$TXT,RT$X,RT$F11
                        .GLOBL  RT$OS,RT$OR,RT$OX
                        .GLOBL  CURSAV,SKPVAL,RSPCS
                        .GLOBL  RT$RA,RT$RB,F$INPL,CISRTI
                        .GLOBL  RT$VLN,RT$CR,MULTPY,DIVIDE
                        .GLOBL  INFLEN,KYUCA,BLKFC
                        .GLOBL  OWCNT,OBUFSZ,START1,STACK
                        .GLOBL  RWDLEN,P2TOP,RCNT1,RCNT2
```

```
                                .GLOBL  KYCOMP,FLDMAP,MAPEND,INRLEN,SELADR
                                .GLOBL  KYDIS,EOFBUF,ANDFLG,ORFLG
                                .GLOBL  MSGNAM,COMPAR,REC1,REC2,POBASE
                                .GLOBL  POTOP,P2SIZ,LOWMEM,P1SIZ
                                .GLOBL  UEOF,V3FLAG,SRTCNT,TMPCNT,P1TOP
                                .GLOBL  KYBIN,KYTSS,KYLSS,KYDATE
                                .GLOBL  SFLAGS,F$BIN,F$CR,F$PAUS,F$VLEN
                                .GLOBL  CINNRT,SAVBUF
                                .GLOBL  SETNRT,RSTS,CHNPPN,FILPPN,SPCFLG
                                .GLOBL  FD1,FD2,FD3,FD4,SPC1,SPC2,SPC3,SPC4,SPC5,FILES
                        ;
                        ;  MISC. PARAMETERS
                        ;
000015                  CR      =       15              ;ASCII CARRIAGE RETURN
000012                  LF      =       12              ;ASCII LINE FEED
000032                  CTRLZ   =       32              ;CONTROL-Z
```

```
                        ;-----------------------------------------------------------------
                        ;  Format of record selection control blocks.
                        ;
000000                  S$LINK  =       0               ;W- LINK TO NEXT CONTROL BLOCK
000002                  S$OPR   =       2               ;B- TYPE OF OPERATION (SO$xxx)
000003                  S$TYP   =       3               ;B- TYPE OF DATA (FT$x)
000004                  S$BR    =       4               ;B- TRUTH TEST (EQ,NE,GT,LT,GE,LE)
000005                  S$FLG   =       5               ;B- CONTROL FLAGS (SF$xxx)
000006                  S$POS1  =       6               ;W- BYTE POSITION OF FIELD 1
000010                  S$SIZ1  =       10              ;W- # BYTES IN FIELD 1
000012                  S$POS2  =       12              ;W- BYTE POSITION OF FIELD 2
                        ;
000014                  S$$SZ   =       14              ;SIZE OF SELECT CONTROL BLOCK
                        ;
                        ;  Operation codes stored in S$OPR
                        ;
000000                  SO$CMP  =       0               ;FIELD COMPARISION
000002                  SO$AND  =       2               ;LOGICAL AND
000004                  SO$OR   =       4               ;LOGICAL OR
000006                  SO$NOT  =       6               ;LOGICAL NOT
                        ;
                        ;  Branch type codes stored in S$BR
                        ;
000001                  BT$HI   =       1               ;BRANCH IF A>B
000002                  BT$LO   =       2               ;BRANCH IF A<B
000004                  BT$REV  =       4               ;REVERSE SENSE OF BRANCH
                        ;
                        ;  Field type codes stored in S$TYP
                        ;
000000                  FT$C    =       0               ;C CHARACTER
000002                  FT$B    =       2               ;B SIGNED BINARY INTEGER
000004                  FT$U    =       4               ;U UNSIGNED BINARY INTEGER
000006                  FT$T    =       6               ;T DATE (MMDDYY)
000010                  FT$F    =       10              ;F FLOATING
000012                  FT$D    =       12              ;D DIBOL SIGNED NUMBER
000014                  FT$A    =       14              ;A UPPER CASE ASCII CHARACTERS
000016                  FT$I    =       16              ;I LEADING SEPARATE SIGN
000020                  FT$J    =       20              ;J TRAILING SEPARATE SIGN
                        ;
                        ;  Control flags stored in S$FLG
                        ;
000001                  SF$LIT  =       1               ;FIELD 2 IS A LITERAL
                        ;
                        ;-----------------------------------------------------------------
                        ;  Table of comparison operators.
                        ;
000000  105     121     CMPOPS: .ASCII  /EQ/
000002  116     105             .ASCII  /NE/
000004  114     124             .ASCII  /LT/
000006  114     105             .ASCII  /LE/
000010  107     124             .ASCII  /GT/
000012  107     105             .ASCII  /GE/
000014                  NUMCO   =       .-CMPOPS        ;2** OPERATORS
                        ;
                        ;  Parallel table of control flags.
                        ;  BT$HI==>True if A>B.
                        ;  BT$LO==>True if A<B.
```

```
                                ;  BT$REV==>Reverse sense of comparison.
                                ;
000014      007                 CMPCOD: .BYTE   BT$HI!BT$LO!BT$REV      ;EQ
000015      003                         .BYTE   BT$HI!BT$LO             ;NE
000016      002                         .BYTE   BT$LO                   ;LT
000017      005                         .BYTE   BT$HI!BT$REV            ;LE
000020      001                         .BYTE   BT$HI                   ;GT
000021      006                         .BYTE   BT$LO!BT$REV            ;GE
                                        .EVEN
            000003              BT$NE   =       BT$HI!BT$LO             ;BT CODE FOR NE OPERATION
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Operation codes used while parsing logical selection expression.
                                ;  Operation code value corresponds to precedence value for operator.
                                ;
            000001              OP$EOE  =       1               ;END OF EXPRESSION
            000002              OP$BOE  =       2               ;BEGINNING OF EXPRESSION
            000003              OP$RPN  =       3               ;RIGHT PAREN.
            000004              OP$LPN  =       4               ;LEFT PAREN.
            000005              OP$OR   =       5               ;"OR"
            000006              OP$AND  =       6               ;"AND"
            000007              OP$NOT  =       7               ;"NOT"
            000010              OP$COR  =       10              ;"," USED WITH EQ (SPECIAL "OR")
            000011              OP$CND  =       11              ;"," USED WITH NE (SPECIAL "AND")
                                ;
                                ;  Table to translate OP$xxx code values to SO$xxx codes.
                                ;
000022      000                 OPTRNS: .BYTE   0
000023      000                         .BYTE   0               ;OP$EOE
000024      000                         .BYTE   0               ;OP$BOE
000025      000                         .BYTE   0               ;OP$RPN
000026      000                         .BYTE   0               ;OP$LPN
000027      004                         .BYTE   SO$OR           ;OP$OR
000030      002                         .BYTE   SO$AND          ;OP$AND
000031      006                         .BYTE   SO$NOT          ;OP$NOT
000032      004                         .BYTE   SO$OR           ;OP$COR
000033      002                         .BYTE   SO$AND          ;OP$CND
                                        .EVEN
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Format of field mapping control blocks.
                                ;
            000000              M$POS1  =       0               ;W-- POSITION OF SOURCE FIELD
            000002              M$POS2  =       2               ;W-- POSITION OF DESTINATION FIELD
            000004              M$SIZE  =       4               ;W-- NUMBER OF BYTES TO MOVE
                                ;
            000006              M$$SZ   =       6               ;SIZE OF MAPPING CONTROL BLOCK
```

```
                                        ;-------------------------------------------------------------------------------
000034  015270                          R50DK:  .RAD50  /DK/
000036  074452                          R50TN0: .RAD50  /SOR/
000040  100760                          R50TN1: .RAD50  /T1X/
000042  101030                          R50TN2: .RAD50  /T2X/
000044  113050                          R50XX:  .RAD50  /XA /
000046  077430                          R50TMP: .RAD50  /TMP/
000050  014646                          R50DDF: .RAD50  /DDF/
000052  014474                          R50DAT: .RAD50  /DAT/
000054  011541                          R50CDA: .RAD50  /CDA/
000056  011542                          R50CDB: .RAD50  /CDB/
000060  012106                          R50CIO: .RAD50  /CIO/
000062  011576                          R50CDO: .RAD50  /CDO/
                                        ;
000064  014646                          DEFEXT: .RAD50  /DDF/           ;INPUT FILE DEFAULT EXTENSION
000066  000000  000000  000000                  .WORD   0,0,0           ;OUTPUT FILE DEFAULT EXTENSIONS
000074  074644                          INDIEX: .RAD50  /SRT/           ;INDIRECT PARAMETER FILE EXTENSION
000076  000000  000000  000000                  .WORD   0,0,0
                                        ;
000104  073376                          CHADEX: .RAD50  /SAV/           ;CHAIN FILE DEFAULT EXTENSION
000106  000000  000000  000000                  .WORD   0,0,0
                                        ;
000114     000     110                  GTLN:   .BYTE   0,110
000116     000     106                  MSGBLK: .BYTE   0,106
000120  011601'                                 .WORD   SRTNAM
000122  000644'                                 .WORD   MSGBUF
000124  000454                                  .WORD   300.
                                        ;
                                        ;  MESSAGE REPLY EMT BLOCK
000126     000     104                  SNDBLK: .BYTE   0,104
000130  000000G                                 .WORD   MSGNAM
000132  000136'                                 .WORD   ABTBUF
000134  000006                                  .WORD   6.
                                        ;  MESSAGE BUFFER
000136  000000                          ABTBUF: .WORD   0
000140  000000                          MRC2:   .WORD   0
000142  000000                          MRC1:   .WORD   0
                                        ;
                                        ;  DATA AREAS
                                        ;
000144                                  TTBUF:  .BLKW   160.
000644                                  MSGBUF: .BLKW   160.
001344                                  FILBUF: .BLKW   80.
001604                                  CINFIL: .BLKW   8.
001624                                  XBUF:   .BLKW   50.
001770                                  SLBUF:  .BLKW   25.
002052                                  SWBUF:  .BLKW   120.
002432                                  GENBUF: .BLKW   256.
003432                                  CBLK:   .BLKW   4
003442                                  CMPTMP: .BLKW   4
003452  000000                          KEYWD:  .WORD   0
003454  000000                          VALHI:  .WORD   0
003456                                  XAREA:  .BLKW   8.
003476  000000                          CURSEL: .WORD   0
003500  000000                          FLDPOS: .WORD   0
003502  000000                          FLDSIZ: .WORD   0
003504  000000                          FLDTYP: .WORD   0
```

```
     58 003506  000000                FLDORD: .WORD   0
     59 003510  000000                RETLOC: .WORD   0
     60 003512  000000                RSTAT:  .WORD   0
     61 003514  000000                INXDEV: .WORD   0
     62 003516                        INDNRT: .BLKW   4
     63                               ;
     64                               ;  Byte data
     65                               ;
     66 003526      000               QUOTE:  .BYTE   0
     67 003527      000               NEGVAL: .BYTE   0
     68 003530      000               DECPT:  .BYTE   0
     69                               ;  Operator stack.
     70 003531                               .BLKB   6.
     71 003537      002               OPRSTP: .BYTE   OP$BOE
     72                                       .EVEN
     73 003540  003537'               OPRSTK: .WORD   OPRSTP
```

```
                                ;
                                ;
                                ;  MACRO TO DEFINE SWITCH NAMES AND PROCESSING ROUTINES.
                                ;
000000                                  .CSECT  SWNAM
000000                          KEYTAB:
000000                                  .CSECT  SWBRV
000000                          KEYBR:
                                ;
                                        .MACRO  SW      NAME,RTN
                                        .NLIST  BIN
                                        .CSECT  SWNAM
                                        .ASCII  /'NAME'/
                                        .CSECT  SWBRV
                                        .WORD   RTN
                                        .PSECT  SORTID,D
                                        .LIST   BIN
                                        .ENDM   SW
                                ;
                                ;  DEFINE SWITCHES (2 CHARS) AND PROCESSING ROUTINES.
                                ;
000000                                  SW      SI,SWSIZE       ;/SIZE:NNN
003542                                  SW      SK,SWSKIP       ;/SKIP:NNN
003542                                  SW      RE,SWREC        ;/RECORDS:NNN
003542                                  SW      EO,SWEOF        ;/EOF:CHAR:COL
003542                                  SW      CH,SWCHAN       ;/CHAIN:NAME
003542                                  SW      KE,SWKEY        ;/KEY:TAS.L:TAS.L:...
003542                                  SW      PA,SWPAUS       ;/PAUSE
003542                                  SW      BL,SWBLOK       ;/BLOCK:NN
003542                                  SW      IN,SWINCL       ;/INCLUDE:(expression)
003542                                  SW      MA,SWMAP        ;/MAP:size:s.d.l:s.d.l:...
003542                                  SW      CR,SWCR         ;/CR
003542                                  SW      SE,SWSND        ;/SEND:TEXT
003542                                  SW      BI,SWBIN        ;/BIN
003542                                  SW      IT,SWITYP       ;/ITYPE:type
003542                                  SW      OT,SWOTYP       ;/OTYPE:type
003542                                  SW      NU,SWNMBR       ;/NUMBER:col.len
003542                                  SW      PP,SWPPN        ;/PPN:[chain ppn]:[file ppn]
                                ;
                                ;  DEFINE END OF SWITCH TABLES
000042                                  .CSECT  SWNAM
000042                          KEYEND:
003542                                  .PSECT  SORTID,D
                                ;
                                ;  KEY TYPE LETTERS AND TYPE CODES
                                ;
003542  103  102  125          KEYTC:  .ASCIZ  /CBUTFDAIJ/
003545  124  106  104
003550  101  111  112
003553  000
003554  000  002  004          KEYCOD: .BYTE   FT$C,FT$B,FT$U,FT$T,FT$F,FT$D,FT$A,FT$I,FT$J
003557  006  010  012
003562  014  016  020
                                        .EVEN
                                ;
                                ;  DEFINE # OF BYTES OF MEMORY NEEDED FOR THE CODE GENERATED
                                ;  TO COMPARE KEYS OF VARIOUS TYPES.
```

```
                                        ;
003566  000036                  CODSIZ: .WORD   30.              ;ASCII
003570  000030                          .WORD   24.              ;SIGNED INTEGER
003572  000022                          .WORD   18.              ;UNSIGNED INTEGER
003574  000030                          .WORD   24.              ;DATE
003576  000030                          .WORD   24.              ;FLOATING POINT
003600  000030                          .WORD   24.              ;DIBOL SIGNED NUMBER
003602  000030                          .WORD   24.              ;UPPERCASE ASCII STRING
003604  000030                          .WORD   24.              ;LEADING SEPARATE SIGN
003606  000030                          .WORD   24.              ;TRAILING SEPARATE SIGN
                                        ;
                                        ;   BRANCH VECTOR TO CODE GENERATOR ROUTINES
                                        ;
003610  011104'                  GENVEC: .WORD   GNASCI           ;ASCII
003612  011306'                          .WORD   GNCMP            ;SIGNED INTEGER
003614  011252'                          .WORD   GNUSI            ;UNSIGNED INTEGER
003616  011336'                          .WORD   GNDATE           ;DATE
003620  011344'                          .WORD   GNBIN            ;FLOATING POINT
003622  011360'                          .WORD   GNDIS            ;SIGNED DIBOL NUMBER
003624  011366'                          .WORD   GNUCA            ;UPPER-CASE ASCII COMPARE
003626  011322'                          .WORD   GNLSS            ;LEADING SEPARATE SIGN
003630  011330'                          .WORD   GNTSS            ;TRAILING SEPARATE SIGN
```

```
                                ;
                                ;  DEFINE ABORT MACRO
                                ;
                                        .MACRO  XXX     COD
                                XXX'COD =       .
                                        MOV     #'COD'.,R0
                                        JMP     TERM
                                        .ENDM   XXX
                                ;
                                ;  SORT ERROR MESSAGES
                                ;
003632  003750'                 ERRTAB: .WORD   EM0
003634  003767'                         .WORD   EM1
003636  004011'                         .WORD   EM2
003640  004035'                         .WORD   EM3
003642  004060'                         .WORD   EM4
003644  004105'                         .WORD   EM5
003646  004143'                         .WORD   EM6
003650  004174'                         .WORD   EM7
003652  004234'                         .WORD   EM8
003654  004263'                         .WORD   EM9
003656  004311'                         .WORD   EM10
003660  004351'                         .WORD   EM11
003662  004401'                         .WORD   EM12
003664  004434'                         .WORD   EM13
003666  004470'                         .WORD   EM14
003670  004527'                         .WORD   EM15
003672  004563'                         .WORD   EM16
003674  004401'                         .WORD   EM17
003676  004621'                         .WORD   EM18
003700  004676'                         .WORD   EM19
003702  004730'                         .WORD   EM20
003704  004766'                         .WORD   EM21
003706  005017'                         .WORD   EM22
003710  005065'                         .WORD   EM23
003712  005141'                         .WORD   EM24
003714  005206'                         .WORD   EM25
003716  005255'                         .WORD   EM26
003720  005311'                         .WORD   EM27
003722  005367'                         .WORD   EM28
003724  005441'                         .WORD   EM29
003726  005502'                         .WORD   EM30
003730  005544'                         .WORD   EM31
003732  005616'                         .WORD   EM32
003734  005672'                         .WORD   EM33
003736  005753'                         .WORD   EM34
003740  006012'                         .WORD   EM35
003742  006051'                         .WORD   EM36
003744  006077'                         .WORD   EM37
003746  006165'                         .WORD   EM38
                                ;
```

```
                        .NLIST  BIN
003750  EM0:    .ASCIZ  /Sort completed/
003767  EM1:    .ASCIZ  /Invalid # of keys/
004011  EM2:    .ASCIZ  /Invalid record size/
004035  EM3:    .ASCIZ  /Invalid field type/
004060  EM4:    .ASCIZ  /Invalid field length/
004105  EM5:    .ASCIZ  /Invalid field starting column/
004143  EM6:    .ASCIZ  /Not Ascending-Descending/
004174  EM7:    .ASCIZ  /Cannot load some device handler/
004234  EM8:    .ASCIZ  /Cannot open input file/
004263  EM9:    .ASCIZ  /Cannot open temp file/
004311  EM10:   .ASCIZ  /Not enough room for output file/
004351  EM11:   .ASCIZ  /Cannot open output file/
004401  EM12:   .ASCIZ  /Overflow of sort temp file/
004434  EM13:   .ASCIZ  /Invalid date in sort record/
004470  EM14:   .ASCIZ  /Error while reading input file/
004527  EM15:   .ASCIZ  /Error on write to temp file/
004563  EM16:   .ASCIZ  /Error on write to output file/
        EM17    =       EM12
004621  EM18:   .ASCIZ  /Temp file link failure (internal sort error)/
004676  EM19:   .ASCIZ  /Bad command string syntax/
004730  EM20:   .ASCIZ  /Unrecognizable command switch/
004766  EM21:   .ASCIZ  /Invalid switch parameter/
005017  EM22:   .ASCIZ  'Required switch missing (/SIZE, /KEY)'
005065  EM23:   .ASCIZ  /Unable to reopen sort temp file for phase 2/
005141  EM24:   .ASCIZ  /Insufficient memory space to do sort/
005206  EM25:   .ASCIZ  /Cannot open sort indirect command file/
005255  EM26:   .ASCIZ  /Error on SRTPLS message EMT/
005311  EM27:   .ASCIZ  /Invalid comparison operator in INCLUDE clause/
005367  EM28:   .ASCIZ  /Field types don't match in INCLUDE clause/
005441  EM29:   .ASCIZ  /Invalid syntax in INCLUDE clause/
005502  EM30:   .ASCIZ  /Invalid literal in INCLUDE clause/
005544  EM31:   .ASCIZ  /Invalid record size value with MAP switch/
005616  EM32:   .ASCIZ  /Invalid field specification with MAP switch/
005672  EM33:   .ASCIZ  /Record read is longer than specified record size/
005753  EM34:   .ASCIZ  /Invalid ITYPE switch parameter/
006012  EM35:   .ASCIZ  /Invalid OTYPE switch parameter/
006051  EM36:   .ASCIZ  /Invalid NUMBER switch/
006077  EM37:   .ASCIZ  /Inplace sort not legal with Indexed organization file/
006165  EM38:   .ASCIZ  /Only one input file is allowed if organization is Indexed/
```

```
                        ;
                        ;  Sort help message
                        ;
006257                  HLPMSG:
006257                  .ASCII  /SORT-Plus version 1.2/<CR><LF><LF>
006307                  .ASCII  /SORT command line:/<CR><LF><LF>
006334                  .ASCII  "[outspc][,tmp1spc[,tmp2spc]]=inspc1,inspc2,...,inspc6/sw1/sw2..."<CR><LF><LF>
006437                  .ASCII  /Valid switches:/<CR><LF>
006460                  .ASCII  "/SIZE:nnn  --  record size (# bytes)"<CR><LF>
006526                  .ASCII  "/REC:nnn   --  number of records to sort"<CR><LF>
006600                  .ASCII  "/SKIP:nnn  --  number of records to skip at start"<CR><LF>
006663                  .ASCII  "/EOF[:'char'[:col]]  --  end of file sentinel character"<CR><LF>
006754                  .ASCII  "/PAUSE     --  ask for output disk after reading all input"<CR><LF>
007050                  .ASCII  "/CHAIN:progspc  --  chain to specified program after sort"<CR><LF>
007143                  .ASCII  "/PPN:[chain ppn]:[file ppn] -- set default chain and file open"<CR><LF>
007243                  .ASCII  "                PPNs (brackets required).  Ignored if not under"<CR><LF>
007326                  .ASCII  "                RSTS"<CR><LF>
007336                  .ASCII  "/SEND:'text' -- pass message through chain area"<CR><LF>
007417                  .ASCII  "/ITYPE:lor -- Input file language, organization & recording mode"<CR><LF>
007521                  .ASCII  "/OTYPE:lor -- Output file language, org., & mode"<CR><LF>
007603                  .ASCII  "       Language types:"<CR><LF>
007625                  .ASCII  "                C = COBOL-Plus"<CR><LF>
007647                  .ASCII  "                D = DIBOL"<CR><LF>
007664                  .ASCII  "                F = FORTRAN"<CR><LF>
007703                  .ASCII  "                T = General variable length ascii text"<CR><LF>
007755                  .ASCII  "                X = General fixed length binary/ascii data"<CR><LF>
010033                  .ASCII  "       Organizations:"<CR><LF>
010054                  .ASCII  "                S = Sequential"<CR><LF>
010076                  .ASCII  "                R = Relative (Random)"<CR><LF>
010127                  .ASCII  "                I = Indexed"<CR><LF>
010146                  .ASCII  "       Recording mode:"<CR><LF>
010170                  .ASCII  "                A = Ascii"<CR><LF>
010205                  .ASCII  "                B = Binary"<CR><LF>
010223                  .ASCII  "/BLOCK:nn  --  use specified blocking factor"<CR><LF>
010301                  .ASCII  "/MESSAGE -- enter message mode"<CR><LF>
010341                  .ASCII  "/KEY:tas.l[:tas.l:...]  --  specify sort keys"<CR><LF>
010420                  .ASCII  "   t = type of key:  C=ASCII character string,"<CR><LF>
010477                  .ASCII  "                     A=Upper-case ASCII,"<CR><LF>
010550                  .ASCII  "                     B=signed binary (COBOL COMP),"<CR><LF>
010633                  .ASCII  "                     T=date (mmddyy),"<CR><LF>
010701                  .ASCII  "                     F=FORTRAN Real and Integer,"<CR><LF>
010762                  .ASCII  "                     D=COBOL signed numeric display and signed DIBOL number"<CR><LF>
011076                  .ASCII  "                     I=COBOL leading separate sign,"<CR><LF>
011162                  .ASCII  "                     J=COBOL trailing separate sign."<CR><LF>
011247                  .ASCII  /   a = "A" for ascending, "D" for descending./<CR><LF>
011326                  .ASCII  "   s = starting column #.  (First col. # = 1)"<CR><LF>
011404                  .ASCII  "   l = length of key (# of columns)."<CR><LF>
011452                  .ASCII  "/INCLUDE:(logical expression)"<CR><LF>
011511                  .ASCII  "/MAP:size:s.d.l:s.d.l:...  --  specify field mapping"<CR><LF>
011577                  .ASCIZ  <LF>
                        ;
011601  SRTNAM:  .ASCII  /RTSORT/          ;SORT INPUT MESSAGE CHANNEL
011607  SRTHD:   .ASCII  /?SRTPLS-F-/<200>
011622  ASTRTX:  .ASCIZ  /*/<200>
011625  TXNRR:   .ASCII  /Records read:  /<200>
011644  TXNRS:   .ASCII  /,  Records sorted:  /<200>
011670  CRLF:    .BYTE   0
```

```
     58 011671  CVTBUF: .ASCIZ  /012345678/
     59                 .EVEN
     60                                        .LIST   BIN
```

```
                                        .SBTTL  SORT INITIALIZATION
000000                                  .PSECT  SORTI           ;BEGIN PROCEDURE SECTION
                                ;
                                ;  INITIAL ENTRY TO SORT
                                ;
000000                          SRTINI:
000000                                  $SYINIT
                                ;
                                ;;;SORTR:       $GETCMD #TTBUF,#160.    ;MACRO TO GET A COMMAND LINE
                                ;  USE THE FOLLOWING CONDITIONAL ASSEMBLY INSTEAD OF
                                ;  A MACRO.  I WOULD HAVE PREFERED A MACRO, BUT IT TURNED
                                ;  OUT TO BE MUCH TOO MESSY.
                                .IF DF RT11
                                ;
                                ;  DETERMINE HOW WE WERE CALLED
                                ;
000170                          SORTR:
000170  105767  0000000G                TSTB    FSTIME          ;GETTING 1ST COMMAND LINE ?
000174  001405                         BEQ     1$              ;BRANCH IF SO
000176  105767  0000000G                TSTB    MODE            ;LOOK TO SEE HOW TO GET COMMAND
000202  002474                         BLT     MWAIT           ;-1 ==> GO WAIT FOR A MESSAGE
000204  001410                         BEQ     SORTD           ; 0 ==> GO GET A NORMAL COMMAND
000206                                  .EXIT                   ;EXIT SORT PROGRAM TO KMON
000210                          1$:
000210  152767  000001  0000000G        BISB    #1,FSTIME       ;INDICATE NO LONGER 1ST TIME THRU
000216  032737  000400  000044        BIT     #CHNBIT,@#JSW   ;WERE WE CHAINED TO?
000224  001055                         BNE     SORTC           ;BR IF CHAINED TO
                                ;
                                ;  We are in direct run mode.
                                ;
000226  105067  0000000G        SORTD:  CLRB    MODE            ;SAY DIRECT RUN MODE
                                ;  ACCEPT COMMAND LINE FROM USER
000232  012701  000144'                 MOV     #TTBUF,R1       ;STORE COMMAND IN TTBUF
000236  105767  0000000G                TSTB    V3FLAG          ;IS THIS RT-11 V2 OR V3?
000242  001416                         BEQ     V2GLIN          ;BR IF RT V2
                                ;  USE VERSION 3 .GTLIN EMT TO GET COMMAND LINE
000244                          V3GLIN: .GTLIN  R1,#ASTRTX      ;ACCEPT COMMAND LINE
                                ;  SEE IF LINE IS CONTINUED
000262  105721                  1$:     TSTB    (R1)+           ;SCAN FOR TRAILING NULL
000264  001376                         BNE     1$
000266  005301                         DEC     R1
000270  124127  000055                 CMPB    -(R1),#'-       ;DID LINE END WITH '-'?
000274  001763                         BEQ     V3GLIN          ;BR IF LINE WAS CONTINUED
000276  000425                         BR      GOTLIN          ;END OF COMMAND
                                ;
                                ;  GET LINE USING .TTYIN IF UNDER RT V2
                                ;
000300                          V2GLIN:
000300  112700  000052          5$:     MOVB    #'*,R0          ;PRINT PROMPT
000304                                  .TTYOUT
000310                          1$:     .TTYIN                  ;ACCEPT CHAR FROM USER
000314  120027  000012                 CMPB    R0,#LF          ;END OF LINE?
000320  001413                         BEQ     2$              ;BR IF END OF COMMAND
000322  120027  000055                 CMPB    R0,#'-          ;WANT TO CONTINUE LINE?
000326  001006                         BNE     3$              ;BR IF NOT
000330                          4$:     .TTYIN                  ;EAT REST OF THIS LINE
000334  120027  000012                 CMPB    R0,#LF
```

```
     58 000340  001373                        BNE     4$
     59 000342  000756                        BR      5$              ;NOW GO GET CONTINUATION LINE
     60 000344  110021              3$:       MOVB    R0,(R1)+        ;ACCRUE COMMAND LINE
     61 000346  000760                        BR      1$
     62 000350  105041              2$:       CLRB    -(R1)           ;PUT NULL AT END OF COMMAND
     63 000352  012701  000144'     GOTLIN:   MOV     #TTBUF,R1       ;SCAN COMMAND FROM TTBUF
     64 000356  000434                        BR      SCNCMD          ;GO PARSE COMMAND
```

```
                                        ;
                                        ;  WE WERE CHAINED TO -- COMMAND IS IN 510-777
                                        ;
000360  112767  000001  000000G SORTC:  MOVB    #1,MODE          ;SET CHAIN MODE
000366  012701  000510                  MOV     #510,R1          ;SCAN COMMAND FROM 510
000372  000426                         BR      SCNCMD           ;GO SCAN COMMAND
                                        ;
                                        ;  MWAIT IS JUMPED TO TO WAIT FOR ANOTHER MESSAGE COMMAND
                                        ;
000374  112767  177777  000000G MWAIT:  MOVB    #-1,MODE         ;REMEMBER WE'RE IN MESSAGE MODE
000402  012700  000116'                 MOV     #MSGBLK,R0       ;WAIT FOR A MESSAGE
000406  104375                         EMT     375
000410  103006                         BCC     1$               ;BR IF NO ERROR
000412  105067  000000G                CLRB    MODE             ;GET OUT OF MESSAGE MODE
000416                                  XXX     26
                                        ;  GOT A MESSAGE.
                                        ;  STORE A NULL AT END OF MESSAGE
000426  105060  000644' 1$:     CLRB    MSGBUF(R0)
000432  012701  000644'                 MOV     #MSGBUF,R1       ;POINT TO SEND MESSAGE BUFFER
000436  012702  000000G                 MOV     #MSGNAM,R2       ;SET RETURN CHANNEL NAME
000442  012122                         MOV     (R1)+,(R2)+      ;MOVE RETURN CHANNEL NAME
000444  012122                         MOV     (R1)+,(R2)+
000446  012112                         MOV     (R1)+,(R2)
                                        ;
                                        ;  SCAN MESSAGE, IGNORE SPACES, PUT FILSPC INTO FILBUF,
                                        ;  PUT SWITCHES INTO SWBUF.
                                        ;  R1 POINTS TO START OF COMMAND STRING.  NULL TERMINATES STRING.
                                        ;
000450  121127  000057          SCNCMD: CMPB    (R1),#'/         ;IS THERE A LEADING SWITCH
000454  001021                         BNE     2$               ;BR IF NOT
000456  126127  000001  000110         CMPB    1(R1),#'H        ;/HELP?
000464  001410                         BEQ     8$               ;BR IF HELP
000466  126127  000001  000115         CMPB    1(R1),#'M        ;/M ONLY LEGAL ONE
000474  001737                         BEQ     MWAIT            ;IT MEANS WAIT FOR A MESSAGE
000476                                  XXX     19
                                    ;  PRINT HELP MESSAGE
000506                              8$: .PRINT  #HLPMSG
000514  000167  177450                 JMP     SORTR            ;GO GET ANOTHER COMMAND LINE
```

```
                                ;
                                ; See if the command begins with a \SPEC\, where SPEC is an initial
                                ; specification of a /CHAIN file but turns on a special error
                                ; handling switch.  This switch specifies that if an error occurs,
                                ; the specified file is chained to with the 7 character low core
                                ; message of "ABORT x", where x is a single byte binary value that
                                ; contains the error code of the aborted sort.
                                ;
000520  121127  000134          2$:     CMPB    (R1),#'\        ;Special error mode?
000524  001015                          BNE     1$              ; => no
000526  005201                          INC     R1              ;Yes, get the specification
000530  012702  001344'                 MOV     #FILBUF,R2
000534  112112                  3$:     MOVB    (R1)+,(R2)
000536  001404                          BEQ     5$
000540  122227  000134                  CMPB    (R2)+,#'\
000544  001373                          BNE     3$
000546  105042                          CLRB    -(R2)
000550  004767  001510          5$:     CALL    SETCHN          ;Do the decode
000554  105267  0000000G                INCB    SPCFLG          ;And flag the special mode
                                ;
                                ;  SEE IF INDIRECT COMMAND LINE WAS SPECIFIED
                                ;  (@FILE SPEC)
                                ;
000560  121127  000100          1$:     CMPB    @R1,#'@         ;INDIRECT COMMAND LINE?
000564  001402                          BEQ     INDRCM          ;BR IF YES
000566  000167  000266                  JMP     @ETEND          ;BR IF NOT
                                ;
                                ;  GET INDIRECT COMMAND LINE
                                ;
000572  005201                  INDRCM: INC     R1              ;SKIP OVER "@"
000574  010605                          MOV     SP,R5           ;SAVE SP
000576  005037  000046                  CLR     @#USRLOC        ;PUT USR IN TOP OF MEMORY
000602                                  .SETTOP #SIEND+2000.    ;MAKE ROOM FOR HANDLERS
000610                                  $LOCK
000612                                  .CSISPC #XBUF,#INDDEX,R1;PARSE COMMAND LINE
000626  103004                          BCC     1$              ;BR IF NO ERROR
000630                          9$:     XXX     25              ;CAN'T OPEN INDIRECT FILE
                                ;  LOAD HANDLER AND OPEN INDIRECT FILE
000640  010506                  1$:     MOV     R5,SP           ;RESET SP
000642  105767  0000000G                TSTB    RSTS            ;Running under RSTS?
000646  001024                          BNE     2$              ; => yes, no fetches needed
000650                                  .FETCH  #SIEND,#XBUF+36 ;LOAD HANDLER
000662  103762                          BCS     9$              ;BR IF ERROR
000664                                  .LOOKUP #XAREA,#1,#XBUF+36
000704  103005                          BCC     2$
000706                          8$:     .RELEAS #XBUF+36        ;ERROR.  RELEASE HANDLER.
000716  000744                          BR      9$
                                ;  READ 1ST BLOCK OF FILE
000720                          2$:     .READW  #XAREA,#1,#GENBUF,#256.,#0
000756  103753                          BCS     8$
                                ;  CLOSE COMMAND FILE
000760                                  .CLOSE  #1
000766                                  .RELEAS #XBUF+36
000776                                  $UNLOCK
                                ;  SUPPRESS TRAILING ENDS OF CONTINUATION LINES
001000  012701  002432'                 MOV     #GENBUF,R1      ;START SCANNING HERE
001004  010102                          MOV     R1,R2           ;PUT RESULT HERE
```

SORT INITIALIZATION

```
     58 001006  112103                    4$:     MOVB    (R1)+,R3        ;GET NEXT CHAR FROM FILE BUFFER
     59 001010  001416                            BEQ     3$              ;BR IF HIT END
     60 001012  120327  000032                    CMPB    R3,#CTRLZ       ;CONTROL-Z CHARACTER?
     61 001016  001413                            BEQ     3$              ;IF YES THEN THIS IS END OF FILE
     62 001020  120327  000055                    CMPB    R3,#'-          ;SKIP CONTINUATION CHAR
     63 001024  001770                            BEQ     4$
     64 001026  120327  000015                    CMPB    R3,#CR          ;AND END OF LINE CHARS
     65 001032  001765                            BEQ     4$
     66 001034  120327  000012                    CMPB    R3,#LF
     67 001040  001762                            BEQ     4$
     68 001042  110322                            MOVB    R3,(R2)+        ;THIS CHAR IS NEEDED
     69 001044  000760                            BR      4$
     70 001046  105022                    3$:     CLRB    (R2)+           ;PUT NULL AT END OF COMMAND
     71 001050  012701  002432'                    MOV     #GENBUF,R1      ;SCAN COMMAND FROM HERE
     72 001054  000167  177370                    JMP     SCNCMD
     73 001060                            GETEND:
     74                                   .ENDC
```

SORT INITIALIZATION

```
                                        ;
                                        ;  WE NOW HAVE COMMAND LINE.
                                        ;
001060                                  SCNCM1:
                                        ;
                                        ;  Check in for debugging breakpoint.
                                        ;
001060  004767  000000G                         CALL    CISRTI          ;CHECK IN TO ROOT
                                        ;
                                        ;  Do initial memory allocation.
                                        ;
001064                                          $GETTOP
001072  010067  000000G                         MOV     R0,TOPMEM       ;SAVE TOP OF MEMORY ADDRESS
001076  012737  014574' 000046                  MOV     #SIEND,@#USRLOC ;SET USR SWAP LOCATION
001104  012704  000057                          MOV     #'/,R4          ;Enable the switch finder
001110  010103                                  MOV     R1,R3           ;Save off string pointer
001112  105767  000000G                         TSTB    RSTS            ;We running under RSTS?
001116  001420                                  BEQ     16$             ; => no
001120  112100                          13$:    MOVB    (R1)+,R0        ;Yes, if there is an equal sign, then
001122  001416                                  BEQ     16$             ;  we must ignore switches to the
001124  004767  011742                          CALL    QUOTCK          ;  left of it (/MODE:n, /SI:m, etc.)
001130  001006                                  BNE     15$
001132  112100                          17$:    MOVB    (R1)+,R0        ;Don't look within quoted strings
001134  001411                                  BEQ     16$
001136  120067  003526'                         CMPB    R0,QUOTE
001142  001373                                  BNE     17$
001144  000765                                  BR      13$
001146  020027  000075                  15$:    CMP     R0,#'=
001152  001362                                  BNE     13$
001154  012704  000040                          MOV     #' ,R4          ;We have a "=", disable switch finder
001160  010301                          16$:    MOV     R3,R1           ;Reset (possibly) the string ptr
001162  012702  001344'                         MOV     #FILBUF,R2      ;PUT FILE SPECS HERE
001166  105012                                  CLRB    (R2)
001170  105067  002052'                         CLRB    SWBUF
001174  005003                                  CLR     R3
001176  112100                          3$:     MOVB    (R1)+,R0        ;GET NEXT CHAR FROM COMMAND
001200  004767  011666                          CALL    QUOTCK          ;START OF QUOTED STRING?
001204  001006                                  BNE     10$             ;BR IF NOT
001206  110022                          11$:    MOVB    R0,(R2)+        ;ACCEPT ALL CHARS IN QUOTED STRING
001210  112100                                  MOVB    (R1)+,R0        ;GET NEXT CHAR IN STRING
001212  001426                                  BEQ     4$              ;BR IF END OF COMMAND HIT
001214  120067  003526'                         CMPB    R0,QUOTE        ;END OF QUOTED STRING?
001220  001372                                  BNE     11$             ;BR IF NOT
001222  120027  000040                  10$:    CMPB    R0,#'           ;IGNORE SPACES
001226  001763                                  BEQ     3$
001230  120027  000141                          CMPB    R0,#141         ;LOWER CASE 'a'?
001234  103405                                  BLO     12$             ;BR IF NOT LOWER CASE LETTER
001236  120027  000172                          CMPB    R0,#172         ;LOWER CASE 'z'?
001242  101002                                  BHI     12$             ;BR IF NOT LOWER CASE LETTER
001244  162700  000040                          SUB     #40,R0          ;CONVERT TO UPPER CASE LETTER
001250  120004                          12$:    CMPB    R0,R4           ;START OF SWITCHES?
001252  001006                                  BNE     4$              ;BR IF NOT
001254  005703                                  TST     R3              ;IS THIS THE 1ST SWITCH?
001256  001004                                  BNE     4$              ;BR IF NOT
001260  105012                                  CLRB    (R2)            ;END FILE SPEC
001262  012702  002052'                         MOV     #SWBUF,R2       ;GO TO SWITCH BUFFER
001266  005203                                  INC     R3
```

```
     58 001270  120027  000075          4$:     CMPB    R0,#'=
     59 001274  001002                          BNE     14$
     60 001276  012704  000057                  MOV     #'/,R4          ;On "=", enable (poss) switch finder
     61 001302  110022                  14$:     MOVB    R0,(R2)+        ;MOVE CHAR TO BUFFER
     62 001304  001334                          BNE     3$              ;CONTINUE UNTIL WE HIT A NULL
```

```
                        .SBTTL  PARSE SWITCHES
                ;
                ;  PARSE SWITCH STRING
                ;
PARSW:  CLR     NUMKY2          ;# OF KEYS
        CLR     INRLEN          ;INPUT RECORD LENGTH
        CLR     RECLEN
        CLR     SKPCNT          ;# OF RECORDS TO SKIP
        CLR     SRTCNT          ;# RECORDS PASSED TO SORT
        CLR     SRTCNT+2
        CLR     TMPCNT          ;# BLOCKS WRITTEN TO TEMP FILE
        CLR     TMPCNT+2
        CLR     SKPVAL
        CLR     RCNT1           ;ZERO RECORD COUNTERS
        CLR     RCNT2
        CLR     EOFBUF
        CLR     FLDMAP          ;NO FIELD MAPPING
        CLR     SELADR          ;NO RECORD SELECTION
        CLR     EOFCOL          ;NO USER EOF CHARACTER
        CLR     BLKFC           ;NO BLOCKING FACTOR YET
        CLRB    EOFCHR          ;END OF FILE CHARACTER
        CLRB    UEOF            ;HAVEN'T HIT USER'S EOF RECORD
        CLR     NUMLEN          ;NO RECORD NUMBERING WANTED
        CLR     SFLAGS          ;STATE FLAGS
        CLR     ITYPE           ;NO INPUT FILE TYPE
        CLR     OTYPE           ;NO OUTPUT FILE TYPE
.IF DF RT11
        CLR     CINSPC          ;CHAIN FILE NAME
        CLR     @#510           ;NO CHAIN MESSAGE
        CLR     CHNPPN          ;Default chain PPN = current PPN
        CLR     FILPPN          ;As well as the default file PPN
.ENDC
        MOV     #177777,MXREC1  ;SET TO SORT INFINITE # RECORDS
        MOV     #177777,MXREC2
;  Initialize record number counter to zero.
        MOV     #'0,R0
        MOV     #NUMBUF,R2      ;BCD RECORD COUNTER BUFFER
2$:     MOVB    R0,(R2)+        ;SET TO ALL ZERO
        CMP     R2,#NUMBUF+8.   ;DONE ALL?
        BLO     2$              ;BR IF NOT
;
```

```
      1 001512  012701  002053'                MOV     #SWBUF+1,R1     ;START SCANNING
      2                                ;  SCAN NEXT KEYWORD AND STORE 1ST TWO CHARS IN KEYWD.
      3 001516  012702  003452'        GETSW:  MOV     #KEYWD,R2       ;RECEIVING BUFFER
      4 001522  005012                  CLR     @R2
      5 001524  112100                3$:     MOVB    (R1)+,R0        ;GET NEXT CHAR
      6 001526  120027  000101                CMPB    R0,#'A          ;IS IT A LETTER?
      7 001532  103410                        BLO     1$              ;BR IF NOT
      8 001534  120027  000132                CMPB    R0,#'Z
      9 001540  101005                        BHI     1$              ;BR IF NOT
     10 001542  020227  003454'               CMP     R2,#KEYWD+2     ;GOTTEN 2 CHARS YET?
     11 001546  103366                        BHIS    3$              ;BR IF YES
     12 001550  110022                        MOVB    R0,(R2)+        ;STORE CHAR
     13 001552  000764                        BR      3$              ;PROCESS REST OF SWITCH KEYWORD
     14                               ;  KEYWORD (2 CHARS) IS NOW IN KEYWD.
     15                               ;  SEE IF WE RECOGNIZE IT.
     16 001554  005301                1$:     DEC     R1              ;POINT TO SWITCH DELIMITER
     17 001556  016702  003452'               MOV     KEYWD,R2        ;GET KEYWORD
     18 001562  012703  000000'               MOV     #KEYTAB,R3      ;POINT TO SWITCH NAME TABLE
     19 001566  020223                5$:     CMP     R2,(R3)+        ;LOOK UP KEYWORD
     20 001570  001407                        BEQ     4$              ;BR IF FOUND
     21 001572  020327  000042'               CMP     R3,#KEYEND      ;CHECKED ALL?
     22 001576  103773                        BLO     5$              ;BR IF MORE TO CHECK
     23 001600                                XXX     20              ;INVALID SWITCH
     24                               ;
     25                               ;  WE RECOGNIZE THE SWITCH -- BRANCH OFF TO PROCESSING ROUTINE.
     26                               ;  R1 POINTS TO DELIMITER WHICH FOLLOWED KEYWORD.
     27 001610  162703  000002'       4$:     SUB     #<KEYTAB+2>,R3
     28 001614  000173  000000'               JMP     @KEYBR(R3)      ;ENTER PROCESSING ROUTINE
```

```
                                        ;-------------------------------------------------------------
                                        ;  /SIZE:NNN   -- RECORD SIZE
                                        ;
001620  004767  011562          SWSIZE: CALL    CKCOL           ;CHECK FOR COLON
001624  004767  011262                  CALL    ACRDEC          ;ACCRUE SIZE
001630  005705                          TST     R5              ;MUST BE >0
001632  003004                          BGT     1$
001634                                  XXX     2               ;INVALID RECORD SIZE
001644  010567  000000G         1$:     MOV     R5,INRLEN       ;SAVE RECORD LENGTH
001650  111100                          MOVB    (R1),R0         ;GET VALUE DELIMITER
001652  001421                          BEQ     NXSJ            ;END OF SWITCHES
001654  120027  000057                  CMPB    R0,#'/          ;START OF NEXT SWITCH
001660  001416                          BEQ     NXSJ            ;BR IF YES
001662  120027  000072                  CMPB    R0,#':          ;ADDITIONAL PARAMETER?
001666  001404                          BEQ     2$              ;BR IF YES
001670                          3$:     XXX     19              ;INVALID SYNTAX
001700  005201                  2$:     INC     R1              ;NEXT PARAMETER SHOULD BE "V"
001702  122127  000126                  CMPB    (R1)+,#'V
001706  001370                          BNE     3$              ;BR IF BAD
001710  052767  000000G 000000G         BIS     #F$VLEN,SFLAGS  ;REMEMBER VARIABLE LENGTH
001716  000167  002544          NXSJ:   JMP     NXTSW
                                        ;
                                        ;-------------------------------------------------------------
                                        ;  /SKIP:NNN  -- SKIP # OF RECORDS
                                        ;
001722  004767  011460          SWSKIP: CALL    CKCOL           ;CHECK FOR COLON
001726  004767  011160                  CALL    ACRDEC          ;ACCRUE VALUE
001732  010567  000000G                 MOV     R5,SKPCNT       ;STORE SKIP COUNT
001736  010567  000000G                 MOV     R5,SKPVAL
001742  000765                          BR      NXSJ
                                        ;
                                        ;-------------------------------------------------------------
                                        ;  /REC:NNN  -- SET MAX # OF RECORDS TO SORT
                                        ;
001744  004767  011436          SWREC:  CALL    CKCOL           ;CHECK FOR COLON
001750  004767  011136                  CALL    ACRDEC          ;ACCRUE VALUE
001754  010567  000000G                 MOV     R5,MXREC1       ;STORE LOW ORDER PART
001760  016767  003454' 000000G         MOV     VALHI,MXREC2    ;STORE HIGH ORDER PART
001766  000753                          BR      NXSJ
                                        ;
                                        ;-------------------------------------------------------------
                                        ;  /BLOCK:NN  --  SET INTERNAL FILE BLOCKING FACTOR.
                                        ;
001770  004767  011412          SWBLOK: CALL    CKCOL           ;CHECK FOR COLON
001774  004767  011112                  CALL    ACRDEC          ;ACCRUE DECIMAL NUMBER
002000  010567  000000G                 MOV     R5,BLKFC        ;SET BLOCKING FACTOR
002004  000744                          BR      NXSJ
                                .IF DF RT11
                                        ;
                                        ;-------------------------------------------------------------
                                        ;  /SEND:'text'  --  SEND MESSAGE THROUGH CHAIN DATA AREA.
                                        ;
002006  004767  011374          SWSND:  CALL    CKCOL           ;CHECK FOR COLON
002012  012702  000510                  MOV     #510,R2         ;POINT TO CHAIN DATA AREA
002016  112100                          MOVB    (R1)+,R0        ;GET 1ST CHARACTER OF MESSAGE
002020  004767  011046                  CALL    QUOTCK          ;IS TEXT STRING QUOTED?
002024  001007                          BNE     2$              ;BR IF NOT
```

```
                                ;  TEXT STRING IS QUOTED
002026  112100                4$:     MOVB    (R1)+,R0        ;MOVE CHARS IN MESSAGE TILL QUOTE HIT
002030  120067  003526'              CMPB    R0,QUOTE        ;END OF MESSAGE?
002034  001412                       BEQ     1$              ;BR IF YES
002036  110022                       MOVB    R0,(R2)+        ;STORE TEXT CHAR IN SAVE AREA
002040  001372                       BNE     4$              ;BR IF NOT END OF COMMAND
002042  000406                       BR      5$              ;HIT END OF COMMAND
                                ;  TEXT STRING IS NOT QUOTED
002044  110022                2$:     MOVB    R0,(R2)+        ;MOVE CHAR TO SAVE AREA
002046  001404                       BEQ     5$              ;BR IF HIT END OF COMMAND
002050  112100                       MOVB    (R1)+,R0        ;GET NEXT TEXT CHAR
002052  120027  000057               CMPB    R0,#'/          ;START OF NEXT SWITCH?
002056  001372                       BNE     2$              ;BR IF NOT
002060  005301                5$:     DEC     R1              ;POINT BACK ONE
002062  105012                1$:     CLRB    (R2)            ;STORE NULL AT END OF STRING
002064  000403                       BR      NXSJ1           ;GO PROCESS NEXT SWITCH
                                .ENDC
                                ;
                                ;-------------------------------------------------------
                                ;  /BIN  --  BINARY RECORDS -- DON'T LOOK FOR CTRL-Z.
                                ;
002066  052767  0000006 0000006 SWBIN:  BIS     #F$BIN,SFLAGS   ;REMEMBER THIS IS A BINARY RECORD
002074  000167  002366          NXSJ1:  JMP     NXTSW
                                ;
                                ;-------------------------------------------------------
                                ;  /EOF:CHAR:COL  -- DECLARE EOF SENTINEL
                                ;
002100  012767  000001  0000006 SWEOF:  MOV     #1,EOFCOL       ;SET DEFAULT EOF COLUMN NUMBER
002106  112767  000032  0000006         MOVB    #CTRLZ,EOFCHR   ;AND EOF CHAR
002114  121127  000072                  CMPB    (R1),#':        ;DID USER SPECIFY VALUE?
002120  001026                          BNE     9$              ;BR IF NOT
002122  004767  011260                  CALL    CKCOL           ;CHECK FOR COLON
                                ;  SEE IF CHAR SPECIFIED IN QUOTES OR AS OCTAL VALUE
002126  111100                          MOVB    (R1),R0         ;GET NEXT CHARACTER
002130  004767  010736                  CALL    QUOTCK          ;IS STRING QUOTED
002134  001004                          BNE     1$              ;BR IF NOT
002136  005201                          INC     R1              ;SKIP OVER QUOTE
002140  112105                          MOVB    (R1)+,R5        ;GET EOF CHAR
002142  005201                          INC     R1              ;SKIP OVER TRAILING QUOTE
002144  000402                          BR      2$
002146  004767  011036          1$:     CALL    ACROCT          ;ACCRUE OCTAL CHAR VALUE
002152  110567  0000006         2$:     MOVB    R5,EOFCHR       ;STORE CHARACTER CODE
002156  121127  000072                  CMPB    (R1),#':        ;IS COLUMN NUMBER SPECIFIED?
002162  001005                          BNE     9$              ;BR IF NOT (USE 1)
002164  005201                          INC     R1              ;SKIP OVER COLON
002166  004767  010720                  CALL    ACRDEC          ;ACCRUE DEC VALUE OF COLUMN
002172  010567  0000006                 MOV     R5,EOFCOL       ;SAVE EOF COLUMN NUMBER
002176  000167  002264          9$:     JMP     NXTSW
```

```
                                .IF DF RT11
                                ;
                                ;-----------------------------------------------------------------
                                ;   /CHAIN:DEV:FILE.EXT  -- CHAIN TO NEXT PROGRAM
                                ;
                                ;  MOVE FILE NAME TO CINFIL AND INSERT TRAILING NULL
002202  004767  011200          SWCHAN: CALL    CKCOL           ;CHECK FOR COLON
002206  012702  001604'                  MOV     #CINFIL,R2      ;PUT NAME HERE
002212  112100                  2$:     MOVB    (R1)+,R0        ;PICK UP CHAR FROM NAME
002214  001410                          BEQ     1$              ;BR IF HIT END
002216  004767  010650                  CALL    QUOTCK          ;IGNORE QUOTES AROUND THE NAME
002222  001773                          BEQ     2$
002224  120027  000057                  CMPB    R0,#'/          ;HIT NEXT SWITCH?
002230  001402                          BEQ     1$              ;BR IF YES
002232  110022                          MOVB    R0,(R2)+        ;MOVE TO CINFIL
002234  000766                          BR      2$
002236  105012                  1$:     CLRB    (R2)            ;PUT NULL AT END OF NAME
002240  005301                          DEC     R1              ;POINT BACK TO DELIMITER
002242  004767  000016                  CALL    SETCHN          ;Set up the chain info
002246  103004                          BCC     10$
002250                                  XXX     21
002260  000167  002202          10$:    JMP     NXTSW

                                ;  LET .CSISPC PARSE FILE NAME
002264  010605                  SETCHN: MOV     SP,R5           ;SAVE SP
002266                                  .CSISPC #XBUF,#CHADEX,#CINFIL
002304  103426                          BCS     5$              ;BR IF BAD NAME
002306  010506                          MOV     R5,SP           ;RESTORE SP

                                ;  MOVE FILE NAME TO SAVE AREA IN ROOT
002310  012702  001662'                  MOV     #XBUF+36,R2
002314  012703  000000G                  MOV     #CINSPC,R3
002320  012223                  4$:     MOV     (R2)+,(R3)+
002322  020327  000010G                  CMP     R3,#CINSPC+10
002326  103774                          BLO     4$
002330  105767  000000G                  TSTB    RSTS            ;We under RSTS?
002334  001412                          BEQ     5$              ; => no
002336  013702  000054                  MOV     @#54,R2         ;Yes, save off the special info
002342  062702  000400                  ADD     #CSPCOF+2,R2    ;  (See the SVRSTS routine's comments
002346  012703  000000G                  MOV     #CINNRT,R3      ;  for what is happening here)
002352  012223                          MOV     (R2)+,(R3)+
002354  012223                          MOV     (R2)+,(R3)+
002356  012223                          MOV     (R2)+,(R3)+
002360  012223                          MOV     (R2)+,(R3)+
002362  000207                  5$:     RETURN
                                .ENDC
```

```
                                        ;
                                        ;-------------------------------------------------------
                                        ;  /KEY:TDS.L:TDS.L:...  -- DEFINE SORT KEYS
                                        ;
002364  016704  000000G        SWKEY:  MOV     NUMKY2,R4       ;GET # OF KEYS SO FAR
002370  062767  000002  000000G        ADD     #2,NUMKY2       ;INC #
002376  004767  011004                CALL    CKCOL           ;CHECK FOR COLON
                                        ;  GET 1ST CHAR WHICH DEFINES TYPE OF KEY
002402  112100                        MOVB    (R1)+,R0
002404  004767  007240                CALL    ACRFLD          ;PARSE FIELD SPEC
002410  016764  003506' 000000G        MOV     FLDORD,KAD(R4)  ;SET ASCENDING/DESCENDING FLAG
002416  016764  003500' 000000G        MOV     FLDPOS,KSTRT(R4);SET STARTING POSITION OF FIELD
002424  016764  003502' 000000G        MOV     FLDSIZ,KLEN(R4) ;SET FIELD LENGTH
002432  016764  003504' 000000G        MOV     FLDTYP,KTYPE(R4);SET FIELD DATA TYPE
002440  121127  000072                CMPB    (R1),#':        ;IS THERE ANOTHER KEY DESCRIPTOR?
002444  001747                        BEQ     SWKEY           ;YES -- GO GET IT
002446  000167  002014                JMP     NXTSW
                                .IF DF RT11
                                        ;
                                        ;-------------------------------------------------------
                                        ;  /PAUSE  --  PAUSE AFTER READING INPUT
                                        ;
002452  052767  000000G 000000G SWPAUS: BIS     #F$PAUS,SFLAGS  ;REMEMBER TO PAUSE
002460  000167  002002                JMP     NXTSW
                                .ENDC
                                        ;-------------------------------------------------------
                                        ;  /NUMBER:col,len -- Insert record number.
                                        ;
002464  004767  010716        SWNMBR: CALL    CKCOL           ;CHECK FOR COLON
002470  004767  010416                CALL    ACRDEC          ;GET STARTING COLUMN NUMBER
002474  005305                        DEC     R5              ;CONVERT TO BYTE INDEX INTO RECORD
002476  010567  000000G               MOV     R5,NUMCOL
002502  122127  000056                CMPB    (R1)+,#'.       ;PERIOD SHOULD BE DELIMITER
002506  001404                        BEQ     1$              ;BR IF OK
002510                                XXX     36              ;INVALID SWITCH
002520  004767  010366        1$:     CALL    ACRDEC          ;GET LENGTH OF NUMBER FIELD
002524  010567  000000G               MOV     R5,NUMLEN
002530  000167  001732                JMP     NXTSW
                                .IF DF RT11
                                        ;
                                        ;-------------------------------------------------------
                                        ;  /PPN:[chain ppn]:[file ppn]  --  Define default chain and file
                                        ;       open request PPNs for RSTS
                                        ;
002534  004067  000022        SWPPN:  JSR     R0,5$           ;Process the first switch
002540  000000G                       .WORD   CHNPPN
002542  121127  000072                CMPB    (R1),#':        ;Is there a second part?
002546  001003                        BNE     1$              ; => no
002550  004067  000006                JSR     R0,5$           ;Yes, process it, also
002554  000000G                       .WORD   FILPPN
002556  000167  001704        1$:     JMP     NXTSW

002562  004767  010620        5$:     CALL    CKCOL           ;Insure colon present
002566  121127  000072                CMPB    (R1),#':        ;Null entry?
002572  001432                        BEQ     10$             ; => yes
002574  122127  000133                CMPB    (R1)+,#'[       ;Check for a left bracket
```

```
     58 002600  001031                        BNE     15$
     59 002602  004767  010304                CALL    ACRDEC          ;Get the project number
     60 002606  032705  177400                BIT     #177400,R5      ;Insure its valid
     61 002612  001024                        BNE     15$
     62 002614  010504                        MOV     R5,R4           ;Load it in high order of R4
     63 002616  000304                        SWAB    R4
     64 002620  122127  000054                CMPB    (R1)+,#',       ;Insure comma is next
     65 002624  001017                        BNE     15$
     66 002626  004767  010260                CALL    ACRDEC          ;Get the programmer number
     67 002632  032705  177400                BIT     #177400,R5      ;Insure its valid
     68 002636  001012                        BNE     15$
     69 002640  122127  000135                CMPB    (R1)+,#']       ;And followed by right bracket
     70 002644  001007                        BNE     15$
     71 002646  050504                        BIS     R5,R4           ;Complete the RSTS PPN word
     72 002650  011005                        MOV     (R0),R5         ;Get the load address
     73 002652  005715                        TST     (R5)            ;Only use the first one seen
     74 002654  001001                        BNE     10$
     75 002656  010415                        MOV     R4,(R5)
     76 002660  005720              10$:      TST     (R0)+           ;Skip the argument
     77 002662  000200                        RTS     R0
     78 002664                      15$:      XXX     21
     79                             .ENDC
     80                             ;
     81                             ;-----------------------------------------------------------------------
     82                             ;  /ITYPE:lor -- Specify input file type.
     83                             ;
     84 002674  004767  000056      SWITYP:   CALL    GETTYP          ;PARSE FILE TYPE
     85 002700  103004                        BCC     1$              ;BR IF NO ERROR
     86 002702                                XXX     34              ;INVALID ITYPE PARAMETER
     87 002712  010267  000000G     1$:       MOV     R2,ITYPE        ;SAVE INPUT FILE TYPE
     88 002716  000167  001544                JMP     NXTSW
     89
     90                             ;-----------------------------------------------------------------------
     91                             ;  /OTYPE:lor -- Specify output file type.
     92                             ;
     93 002722  004767  000030      SWOTYP:   CALL    GETTYP          ;PAUSE FILE TYPE
     94 002726  103407                        BCS     1$              ;BR IF ERROR
     95 002730  032702  000000G               BIT     #RT$OX,R2       ;OUTPUT ORGANIZATION CANNOT BE INDEXED
     96 002734  001004                        BNE     1$              ;BR IF IT IS
     97 002736  010267  000000G               MOV     R2,OTYPE        ;SAVE OUTPUT FILE TYPE FLAGS
     98 002742  000167  001520                JMP     NXTSW
     99 002746                      1$:       XXX     35              ;INVALID OTYPE PARAMETER
```

```
                        ;-------------------------------------------------------------------------------
                        ;  Get file type codes (RT$xxx) specified with the /ITYPE & /OTYPE switches.
                        ;  On return the flags are in R2.
                        ;
002756  004767  010424  GETTYP: CALL    CKCOL           ;CHECK FOR COLON
002762  005002                 CLR     R2              ;BUILD FLAGS IN R2
                        ;
                        ;  Get language type.
                        ;
002764  112100                 MOVB    (R1)+,R0        ;GET LANGUAGE CODE LETTER
002766  120027  000103         CMPB    R0,#'C          ;COBOL?
002772  001003                 BNE     1$              ;BR IF NOT
002774  052702  0000006        BIS     #RT$CBL,R2      ;COBOL FLAG
003000  000435                 BR      GTORG
003002  120027  000104  1$:     CMPB    R0,#'D          ;DIBOL?
003006  001003                 BNE     2$              ;BR IF NOT
003010  052702  0000006        BIS     #RT$DBL,R2      ;DIBOL FLAG
003014  000427                 BR      GTORG
003016  120027  000106  2$:     CMPB    R0,#'F          ;FORTRAN?
003022  001003                 BNE     3$              ;BR IF NOT
003024  052702  0000006        BIS     #RT$FTN,R2      ;FORTRAN FLAG
003030  000421                 BR      GTORG
003032  120027  000130  3$:     CMPB    R0,#'X          ;ARBITRARY DATA?
003036  001003                 BNE     4$              ;BR IF NOT
003040  052702  0000006        BIS     #RT$X,R2        ;ARBITRARY DATA TYPE
003044  000413                 BR      GTORG
003046  120027  000124  4$:     CMPB    R0,#'T          ;TEXT FILE?
003052  001003                 BNE     5$              ;BR IF NOT
003054  052702  0000006        BIS     #RT$TXT,R2      ;TEXT FLAG
003060  000405                 BR      GTORG
003062  120027  000122  5$:     CMPB    R0,#'R          ;FILES-11 FORMAT?
003066  001066                 BNE     GTERR           ;NOPE, AND NONE OF ABOVE
003070  052702  0000006        BIS     #RT$F11,R2
                        ;
                        ;  Get file organization type.
                        ;
003074  111100         GTORG:  MOVB    (R1),R0         ;GET ORGANIZATION CODE LETTER
003076  001434                 BEQ     5$              ;BR IF HIT END OF COMMAND
003100  120027  000057         CMPB    R0,#'/          ;HIT NEXT SWITCH?
003104  001431                 BEQ     5$              ;BR IF YES
003106  005201                 INC     R1              ;ADVANCE CHAR POINTER
003110  120027  000123         CMPB    R0,#'S          ;SEQUENTIAL ORG?
003114  001003                 BNE     1$              ;BR IF NOT
003116  052702  0000006        BIS     #RT$OS,R2       ;ORG=SEQ
003122  000425                 BR      GTRMD
003124  120027  000122  1$:     CMPB    R0,#'R          ;RELATIVE ORGANIZATION?
003130  001003                 BNE     2$              ;BR IF NOT
003132  052702  0000006        BIS     #RT$OR,R2       ;ORG=RELATIVE
003136  000417                 BR      GTRMD
003140  120027  000111  2$:     CMPB    R0,#'I          ;ISAM ORG?
003144  001403                 BEQ     3$              ;BR IF YES
003146  120027  000130         CMPB    R0,#'X          ;ISAM ORG?
003152  001034                 BNE     GTERR           ;ERROR IF NONE OF ABOVE
003154  032702  0000006        BIT     #RT$CBL,R2      ;ISAM ONLY LEGAL FOR COBOL
003160  001431                 BEQ     GTERR           ;BR IF NOT COBOL
003162  052702  0000006        BIS     #RT$OX,R2       ;ORG=ISAM
003166  000403                 BR      GTRMD           ;GO GET RECORDING MODE
```

```
                        ;  Set default organization and recording mode.
003170  052702  000000C 5$:     BIS     #RT$OS+RT$RA,R2 ;ORG=SEQUENTIAL, RECORDING MODE = ASCII
003174  000421                 BR      GTOK
                        ;
                        ;  Get recording mode.
                        ;
003176  111100          GTRMD:  MOVB    (R1),R0         ;GET RECORDING MODE CODE LETTER
003200  001407                 BEQ     1$              ;BR IF HIT END OF COMMAND
003202  120027  000057         CMPB    R0,#'/          ;HIT NEXT SWITCH?
003206  001404                 BEQ     1$              ;BR IF YES
003210  005201                 INC     R1              ;ADVANCE CHAR POINTER
003212  120027  000101         CMPB    R0,#'A          ;ASCII?
003216  001003                 BNE     2$              ;BR IF NOT
003220  052702  000000G 1$:     BIS     #RT$RA,R2       ;MODE=ASCII
003224  000405                 BR      GTOK
003226  120027  000102 2$:     CMPB    R0,#'B          ;BINARY?
003232  001004                 BNE     GTERR           ;ERROR IF NEITHER
003234  052702  000000G         BIS     #RT$RB,R2       ;MODE=BINARY
                        ;
                        ;  Finished.
                        ;
003240  000241          GTOK:   CLC                     ;SIGNAL GOOD RETURN
003242  000207                 RETURN
                        ;
                        ;  Error.
                        ;
003244  000261          GTERR:  SEC                     ;SIGNAL ERROR ON RETURN
003246  000207                 RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  /INCLUDE:(expression) -- Perform record selection.
                                ;
003250  004767  010132          SWINCL: CALL    CKCOL           ;CHECK FOR COLON
003254  012767  003537' 003540'        MOV     #OPRSTP,OPRSTK  ;INIT OPERATOR STACK
003262  005002                        CLR     R2              ;NO SELECT BLOCKS YET
003264  105067  000000G               CLRB    ANDFLG          ;NO "AND" OPERATORS YET
003270  105067  000000G               CLRB    ORFLG           ;NO "OR" OPERATORS YET
                                ;
                                ;  Get next comparison specification.
                                ;
003274  112100                  SLCSPC: MOVB    (R1)+,R0        ;GET NEXT CHARACTER
003276  120027  000050                CMPB    R0,#'(          ;OPEN PAREN?
003302  001006                        BNE     4$              ;BR IF NOT
                                ;  Got open paren.  Push operator on stack.
003304  005367  003540'                DEC     OPRSTK          ;MAKE ROOM ON STACK
003310  112777  000004  003540'        MOVB    #OP$LPN,@OPRSTK ;PUT LEFT PAREN OPERATOR ON OPERATOR STACK
003316  000766                        BR      SLCSPC
                                ;
                                ;  Check for "NOT" operator
                                ;
003320  120027  000116          4$:     CMPB    R0,#'N          ;START OF "NOT"?
003324  001022                        BNE     1$              ;BR IF NOT
003326  122127  000117                CMPB    (R1)+,#'O       ;CHECK SPELLING
003332  001013                        BNE     9$
003334  122127  000124                CMPB    (R1)+,#'T
003340  001010                        BNE     9$
003342  012703  000007                MOV     #OP$NOT,R3      ;SAY OPERATOR IS "NOT"
003346  105267  000000G               INCB    ANDFLG          ;FORCE FULL EXPRESSION EVALUATION
003352  105267  000000G               INCB    ORFLG
003356  000167  000436                JMP     SLOPX           ;DO EXPRESSION GENERATION
003362                          9$:     XXX     29              ;INVALID SELECT SYNTAX
                                ;
                                ;  This should be the start of a field specification.
                                ;
003372  004767  006252          1$:     CALL    ACRFLD          ;ACCRUE FIELD SPECIFICATION
                                ;  Build a skeleton selection control block.
003376  162767  000014  0000006       SUB     #S$$SZ,TOPMEM   ;RESERVE ROOM FOR BLOCK
003404  016703  0000006               MOV     TOPMEM,R3       ;BUILD BLOCK HERE
003410  005702                        TST     R2              ;IS THIS THE FIRST BLOCK?
003412  001403                        BEQ     2$              ;BR IF YES
003414  010362  000000                MOV     R3,S$LINK(R2)   ;CHAIN LAST BLOCK FORWARD TO NEW BLOCK
003420  000402                        BR      3$
003422  010367  0000006         2$:     MOV     R3,SELADR       ;REMEMBER ADDRESS OF FIRST CONTROL BLOCK
003426  010302                  3$:     MOV     R3,R2           ;ADDR OF NEW BLOCK IS CARRIED IN R2
003430  010267  003476'                MOV     R2,CURSEL       ;REMEMBER ADDR OF CURRENT SELECT BLOCK
003434  005062  000000                CLR     S$LINK(R2)      ;CLEAR OUR FORWARD LINK
003440  105062  000005                CLRB    S$FLG(R2)       ;CLEAR CONTROL FLAG BYTE
003444  112762  000000  000002        MOVB    #SO$CMP,S$OPR(R2);SAY OPERATION IS FIELD COMPARISON
                                ;  Store field specification information in select block.
003452  016762  003500' 000006        MOV     FLDPOS,S$POS1(R2);STARTING BYTE POSITION
003460  016762  003502' 000010        MOV     FLDSIZ,S$SIZ1(R2);FIELD SIZE
003466  116762  003504' 000003        MOVB    FLDTYP,S$TYP(R2);DATA TYPE
                                ;  Now get comparison operator.
003474  112103                  SLCMOP: MOVB    (R1)+,R3        ;GET 1ST CHARACTER
003476  112100                        MOVB    (R1)+,R0        ;GET 2ND CHARACTER
003500  000300                        SWAB    R0              ;PUT TOGETHER THE 2 CHARACTERS
```

```
                        BIS     R3,R0
                        MOV     #NUMCO,R3       ;GET 2* NUMBER OF OPERATORS
2$:                     CMP     R0,CMPOPS-2(R3) ;LOOK UP OPERATOR NAME
                        BEQ     1$              ;BR IF FOUND
                        SUB     #2,R3           ;CHECK NEXT
                        BNE     2$
                        XXX     27              ;INVALID COMPARISON OPERATOR
                ;  Convert operator to 2-bit type code.
1$:                     ASR     R3              ;CONVERT TO BYTE TABLE INDEX
                        MOVB    CMPCOD-1(R3),S$BR(R2) ;SET COMPARISON CODE
                ;
                ;  Get operand which follows comparison operator.
                ;
SLOPND:         MOVB    (R1)+,R0        ;GET 1ST LETTER OF OPERAND
                        CMPB    R0,#'A          ;SEE IF IT IS A LETTER
                        BLO     2$              ;BR IF NOT
                        CMPB    R0,#'Z
                        BLE     1$              ;BR IF LETTER
                ;  Operand must be a literal.
                ;  Get it and store in comparison block.
2$:             CALL    SLLIT           ;ACCRUE LITERAL VALUE
                        BISB    #SF$LIT,S$FLG(R2);SAY OPERAND 2 IS A LITERAL
                        BR      SLOPR           ;GO CHECK NEXT OPERATOR
                ;  Operand must be another field descriptor.
                ;  Get it and store info in select control block.
1$:             CALL    ACRFLD          ;ACCRUE FIELD DESCRIPTOR
                        MOV     FLDPOS,S$POS2(R2);STORE FIELD POSITION
                        CMPB    FLDTYP,S$TYP(R2);COMPARE FIELD DATA TYPES
                        BEQ     SLOPR           ;BR IF THEY MATCH
                        XXX     28              ;FIELD TYPES DON'T MATCH
                ;
                ;  We have finished building a comparison control block.
                ;  See what follows.
                ;
SLOPR:          MOVB    (R1)+,R0        ;GET NEXT CHARACTER
                        BEQ     1$              ;BR IF HIT END OF COMMAND LINE
                        CMPB    R0,#'/          ;START OF NEXT SWITCH?
                        BNE     2$              ;BR IF NOT
1$:             DEC     R1              ;POINT BACK TO DELIMITER
                        MOVB    #OP$EOE,R3      ;SET END-OF-EXPRESSION OPERATOR
                        BR      SLOPX           ;FORCE EXPRESSION EVALUATION
2$:             CMPB    R0,#')          ;RIGHT PAREN?
                        BNE     3$              ;BR IF NOT
                        MOVB    #OP$RPN,R3      ;SET RIGHT PAREN OPERATOR CODE
                        BR      SLOPX           ;EVALUEATE EXPRESSION
                ;  Check for "AND" and "OR" logical operators.
3$:             CMPB    R0,#'A          ;START OF "AND"?
                        BNE     4$              ;BR IF NOT
                        CMPB    (R1)+,#'N       ;CHECK REST OF SPELLING
                        BNE     9$
                        CMPB    (R1)+,#'D
                        BNE     9$
                        MOVB    #OP$AND,R3      ;SET AND OPERATOR
                        INCB    ANDFLG          ;REMEMBER WE'VE HAD AN AND OPERATOR
                        BR      SLOPX           ;EVALUATE EXPRESSION
4$:             CMPB    R0,#'O          ;START OF "OR"?
                        BNE     5$              ;BR IF NOT
```

```
    115 003730  122127  000122                  CMPB    (R1)+,#'R       ;CHECK REST OF SPELLING
    116 003734  001404                          BEQ     6$
    117 003736                          9$:     XXX     29              ;INVALID SELECT SYNTAX
    118 003746  112703  000005          6$:     MOVB    #OP$OR,R3       ;GET OR OPERATION CODE
    119 003752  105267  000000G                 INCB    ORFLG           ;REMEMBER WE'VE HAD AN OR OPERATOR
    120 003756  000420                          BR      SLOPX
    121                                 ;  Check for comma as field separator.
    122 003760  120027  000054          5$:     CMPB    R0,#',          ;IS COMMA THE FIELD SEPARATOR
    123 003764  001364                          BNE     9$              ;INVALID SYNTAX IF ANYTHING ELSE
    124                                 ;  Comma is the operator.
    125                                 ;  Translate that into special AND or OR operation.
    126 003766  126227  000004  000003          CMPB    S$BR(R2),#BT$NE ;IS COMPARISON OPERATION "NE"?
    127 003774  001405                          BEQ     7$              ;BR IF YES
    128 003776  012703  000010                  MOV     #OP$COR,R3      ;SET COMMA-OR OPERATOR
    129 004002  105267  000000G                 INCB    ORFLG           ;REMEMBER WE'VE HAD AN OR OPERATION
    130 004006  000404                          BR      SLOPX
    131 004010  012703  000011          7$:     MOV     #OP$CND,R3      ;SET OPERATOR TO "AND"
    132 004014  105267  000000G                 INCB    ANDFLG          ;REMEMBER WE'VE HAD AN AND OPERATION
    133                                 ;
    134                                 ;  New operator code is in R3.
    135                                 ;  Compare new operator to operator on top of operator stack to
    136                                 ;  determine which operator should be applied first.
    137                                 ;
    138 004020  120377  003540'         SLOPX:  CMPB    R3,@OPRSTK      ;WHICH OPERATOR HAS HIGHER PRECEDENCE?
    139 004024  101046                          BHI     SLPSOP          ;BR IF NEW OPERATOR IS TO BE DONE FIRST
    140                                 ;  Operator on top of stack is to be processed next.
    141 004026  117704  003540'                 MOVB    @OPRSTK,R4      ;GET OPERATOR
    142 004032  005267  003540'                 INC     OPRSTK          ;POP STACK
    143 004036  120427  000004                  CMPB    R4,#OP$LPN      ;LEFT PAREN OPERATOR?
    144 004042  001011                          BNE     1$              ;BR IF NOT
    145 004044  120327  000003                  CMPB    R3,#OP$RPN      ;IS NEW OPERATOR RIGHT PAREN?
    146 004050  001404                          BEQ     2$              ;OK IF YES
    147 004052                          9$:     XXX     29              ;INVALID SYNTAX IF NOT
    148 004062  000167  177542          2$:     JMP     SLOPR           ;GO GET NEXT OPERATOR
    149 004066  120427  000002          1$:     CMPB    R4,#OP$BOE      ;REACHED BEGINNING OF EXPRESSION OPERATOR?
    150 004072  001002                          BNE     3$              ;BR IF NOT
    151 004074  000167  000366                  JMP     NXTSW           ;WE ARE FINISHED IF YES
    152 004100  120427  000003          3$:     CMPB    R4,#OP$RPN      ;RIGHT PAREN OPERATOR?
    153 004104  001762                          BEQ     9$              ;BR IF YES
    154                                 ;  Build an operator control block.
    155 004106  162767  000014  000000G         SUB     #S$$SZ,TOPMEM   ;RESERVE ROOM FOR BLOCK
    156 004114  016762  000000G 000000          MOV     TOPMEM,S$LINK(R2);CHAIN PREVIOUS BLOCK TO US
    157 004122  016702  000000G                 MOV     TOPMEM,R2       ;GET ADDRESS OF NEW BLOCK TO R2
    158 004126  005062  000000                  CLR     S$LINK(R2)      ;CLEARE OUR FORWARD LINK
    159                                 ;  Translate OP$xxx codes to SO$xxx codes.
    160 004132  116462  000022' 000002          MOVB    OPTRNS(R4),S$OPR(R2);SET TRANSLATED CODE
    161                                 ;  Finished with this operator.
    162                                 ;  See if we should process more now.
    163 004140  000727                          BR      SLOPX
    164                                 ;
    165                                 ;  It is not time to generate a block for current operator so
    166                                 ;  push it on top of operator stack.
    167                                 ;
    168 004142  005367  003540'         SLPSOP: DEC     OPRSTK          ;MAKE ROOM ON STACK
    169 004146  110377  003540'                 MOVB    R3,@OPRSTK      ;PUT OPERATOR ON STACK
    170                                 ;
    171                                 ;  If this operator represents a comma separating operands,
```

```
                                ;  generate a comparison block to hold info about next operand
                                ;  in list.
                                ;
004152  120327  000010                  CMPB    R3,#OP$COR      ;COMMA-OR?
004156  001405                          BEQ     1$              ;BR IF YES
004160  120327  000011                  CMPB    R3,#OP$CND      ;COMMA-AND?
004164  001402                          BEQ     1$              ;BR IF YES
004166  000167  177102                  JMP     SLCSPC          ;GO GET NEXT OPERAND
                                ;  Build comparison control block for next operand in list.
004172  162767  000014  000000G 1$:     SUB     #S$$SZ,TOPMEM   ;RESERVE ROOM FOR NEXT BLOCK
004200  016762  000000G 000000          MOV     TOPMEM,S$LINK(R2);CHAIN LAST BLOCK FORWARD TO NEW BLOCK
004206  016702  000000G                 MOV     TOPMEM,R2       ;GET ADDRESS OF NEW BLOCK TO R2
004212  005062  000000                  CLR     S$LINK(R2)      ;CLEARE OUR FORWARD LINK
004216  016703  003476'                 MOV     CURSEL,R3       ;GET ADDRESS OF LAST COMPARISON BLOCK
004222  016362  000006  000006          MOV     S$POS1(R3),S$POS1(R2) ;COPY INFO FROM LAST BLOCK
004230  016362  000010  000010          MOV     S$SIZ1(R3),S$SIZ1(R2)
004236  116362  000003  000003          MOVB    S$TYP(R3),S$TYP(R2)
004244  116362  000002  000002          MOVB    S$OPR(R3),S$OPR(R2)
004252  116362  000004  000004          MOVB    S$BR(R3),S$BR(R2)
004260  105062  000005                  CLRB    S$FLG(R2)
004264  000167  177254                  JMP     SLOPND          ;GO GET NEXT OPERAND IN LIST
```

```
                                ;-----------------------------------------------------------------------
                                ;  /MAP:size:s.d.l:s.d.l:...   --  specify field mapping
                                ;
004270  004767  007112  SWMAP:  CALL    CKCOL           ;CHECK FOR COLON
004274  004767  006612          CALL    ACRDEC          ;GET OUTPUT RECORD SIZE
004300  010567  000000G         MOV     R5,RECLEN       ;SAVE OUTPUT RECORD SIZE
004304  122127  000072          CMPB    (R1)+,#':       ;SIZE SHOULD BE FOLLOWED BY COLON
004310  001404                  BEQ     1$              ;BR IF OK
004312                          XXX     31              ;INVALID RECORD SIZE SPEC WITH /MAP
004322  016702  000000G 1$:     MOV     TOPMEM,R2       ;GET ADDR OF FREE MEMORY AREA
004326  162702  000006          SUB     #M$$SZ,R2       ;RESERVE ROOM FOR 1ST MAP BLOCK
004332  010267  000000G         MOV     R2,FLDMAP       ;REMEMBER ADDR OF 1ST BLOCK
                                ;  Parse next field descriptor.
004336  004767  006550  4$:     CALL    ACRDEC          ;ACCRUE STARTING COLUMN NUMBER
004342  005305                  DEC     R5              ;CONVERT TO BYTE NUMBER
004344  002405                  BLT     9$              ;BR IF INVALID
004346  010562  000000          MOV     R5,M$POS1(R2)   ;SAVE POS OF SOURCE FIELD
004352  122127  000056          CMPB    (R1)+,#'.       ;PERIOD SHOULD BE DELIMITER
004356  001404                  BEQ     2$              ;BR IF OK
004360                  9$:     XXX     32              ;INVALID /MAP FIELD SPECIFICATION
004370  004767  006516  2$:     CALL    ACRDEC          ;ACCRUE DESTINATION FIELD COLUMN NUMBER
004374  005305                  DEC     R5              ;CONVERT TO BYTE NUMBER
004376  002770                  BLT     9$              ;BR IF INVALID
004400  010562  000002          MOV     R5,M$POS2(R2)   ;SAVE DESTINATION FIELD POSITION
004404  122127  000056          CMPB    (R1)+,#'.       ;PERIOD SHOULD BE DELIMITER
004410  001363                  BNE     9$              ;BR IF ERROR
004412  004767  006474          CALL    ACRDEC          ;ACCURE LENGTH SPECIFICATION
004416  010562  000004          MOV     R5,M$SIZE(R2)   ;SAVE FIELD LENGTH
004422  122127  000072          CMPB    (R1)+,#':       ;IS THERE ANOTHER FIELD SPEC?
004426  001003                  BNE     5$              ;BR IF NOT
004430  162702  000006          SUB     #M$$SZ,R2       ;MAKE ROOM FOR ANOTHER MAP BLOCK
004434  000740                  BR      4$              ;GO ACCRUE NEXT FIELD DESCRIPTOR
                                ;  Hit end of /MAP switch specification
004436  005301          5$:     DEC     R1              ;POINT BACK TO DELIMITER
004440  010267  000000G         MOV     R2,MAPEND       ;SAVE ADDRESS OF LAST MAP BLOCK
004444  010267  000000G         MOV     R2,TOPMEM       ;SAVE NEW TOP-OF-MEMORY ADDRESS
004450  000167  000012          JMP     NXTSW           ;GO GET NEXT SWITCH
                                .IF DF RT11
                                ;
                                ;-----------------------------------------------------------------------
                                ;  /CR  --  Append cr-lf to end of records.
                                ;
004454  052767  000000G 000000G SWCR:   BIS     #F$CR,SFLAGS    ;REMEMBER FLAG OCCURED
004462  000167  000000          JMP     NXTSW           ;PROCESS NEXT SWITCH
                                .ENDC
                                ;
                                ;-----------------------------------------------------------------
                                ;  MOVE ON TO NEXT SWITCH
                                ;
004466  112100  NXTSW:  MOVB    (R1)+,R0        ;GET SWITCH DELIMITER
004470  001411          BEQ     ENDSW           ;BR IF HIT END OF SWITCHES
004472  120027  000057          CMPB    R0,#'/          ;START OF NEW SWITCH?
004476  001404                  BEQ     1$              ;BR IF YES
004500                          XXX     19
004510  000167  175002  1$:     JMP     GETSW           ;GO PROCESS NEXT SWITCH
```

```
                                        ;
                                        ;  GOT ALL SWITCHES.  CHECK THEM.
                                        ;
004514  005767  000000G        ENDSW:  TST     INRLEN          ;DID WE GET RECORD LENGTH?
004520  001010                        BNE     3$              ;BR IF YES
004522  032767  000000G 000000G       BIT     #RT$OX,ITYPE    ;IS THIS AN ISAM FILE?
004530  001004                        BNE     3$              ;DON'T HAVE TO SPECIFY RECORD LENGTH FOR ISAM FILE
004532                          2$:     XXX     22
004542  116701  000000G        3$:     MOVB    NUMKY2,R1       ;GET # OF KEYS SPECIFIED
004546  001771                        BEQ     2$              ;ERROR IF NONE
004550  005002                        CLR     R2
004552  026227  000000G 000006 5$:     CMP     KTYPE(R2),#FT$T ;DATE TYPE KEY?
004560  001003                        BNE     4$              ;BR IF NOT
004562  012762  000006  000000G       MOV     #6.,KLEN(R2)    ;SET DATE KEY LENGTH TO 6 CHARSS
004570  062702  000002        4$:     ADD     #2,R2           ;CHECK NEXT KEY
004574  020201                        CMP     R2,R1           ;CHECKED ALL?
004576  103765                        BLO     5$              ;BR IF NOT
                                        ;
                                        ;  Check input and output file types.
                                        ;
004600  016702  000000G        CKFTYP: MOV     ITYPE,R2        ;GET INPUT FILE TYPE FLAGS
004604  001002                        BNE     1$              ;BR IF TYPE SPECIFIED
004606  012702  000000C                MOV     #RT$DBL+RT$OR+RT$RB,R2 ;DEFAULT TO DIBOL, RELATIVE, BINARY
004612  016703  000000G        1$:     MOV     OTYPE,R3        ;GET OUTPUT FILE TYPE FLAGS
004616  001010                        BNE     2$              ;BR IF SOME SPECIFIED
004620  010203                        MOV     R2,R3           ;DEFAULT TO SAME TYPE AS INPUT
004622  032702  000000G                BIT     #RT$OX,R2       ;IS INPUT FILE ORG = INDEXED?
004626  001404                        BEQ     2$              ;BR IF NOT
004630  042703  000000G                BIC     #RT$OX,R3       ;DO NOT DEFAULT OUTPUT ORG TO INDEXED
004634  052703  000000G                BIS     #RT$OS,R3       ;DEFAULT OUTPUT ORG TO SEQUENTIAL
                                    ;  Check obsolete /BIN and /SIZE:V switch settings.
004640  032767  000000G 000000G 2$:     BIT     #F$VLEN,SFLAGS  ;VARIABLE LENGTH?
004646  001404                        BEQ     3$              ;BR IF NOT
004650  052702  000000G                BIS     #RT$VLN,R2      ;SAY BOTH INPUT & OUTPUT TYPES ARE VAR LEN
004654  052703  000000G                BIS     #RT$VLN,R3
004660                              3$:
                                    ;  Determine if input and/or output records are to be blank stripped.
004660  032702  000000G        4$:     BIT     #RT$RB,R2       ;INPUT MODE BINARY?
004664  001002                        BNE     5$              ;BR IF YES
004666  052702  000000G                BIS     #RT$VLN,R2      ;IS ASCII MODE THEN SET VARIABLE LENGTH FLAG
004672  032703  000000C        5$:     BIT     #RT$RB+RT$OR+RT$OX+RT$X,R3 ;BINARY, RELATIVE, ISAM OR "X" TYPE?
004676  001002                        BNE     6$              ;IF ANY OF ABOVE THEN FIXED LENGTH OUTPUT
004700  052703  000000G                BIS     #RT$VLN,R3      ;ALLOW VARIABLE LENGTH OUTPUT RECORDS
                                    ;  See if we need to manage CR-LF record terminator characters.
004704  032702  000000C        6$:     BIT     #RT$RB+RT$X,R2  ;INPUT BINARY OR FIXED-LENGTH?
004710  001010                        BNE     7$              ;BR IF EITHER
004712  032702  000000G                BIT     #RT$FTN,R2      ;INPUT FROM FORTRAN FILE?
004716  001403                        BEQ     8$              ;BR IF NOT
004720  032702  000000G                BIT     #RT$OR,R2       ;FORTRAN RELATIVE FILE?
004724  001002                        BNE     7$              ;NO CR-LF ON FORTRAN RELATIVE FILES
004726  052702  000000G        8$:     BIS     #RT$CR,R2       ;CR-LF IS RECORD TERMINATOR
004732  032703  000000C        7$:     BIT     #RT$RB+RT$X,R3  ;OUTPUT FILE BINARY OR FIXED-LENGTH?
004736  001010                        BNE     9$              ;BR IF EITHER
004740  032703  000000G                BIT     #RT$FTN,R3      ;OUTPUT TO FORTRAN DATA FILE?
004744  001403                        BEQ     10$             ;BR IF NOT
004746  032703  000000G                BIT     #RT$OR,R3       ;FORTRAN RELATIVE FILE?
004752  001002                        BNE     9$              ;BR IF YES (NO CR-LF)
```

```
     58 004754  052703  0000006          10$:    BIS     #RT$CR,R3       ;ADD CR-LF TO RECORD END
     59 004760  010267  0000006          9$:     MOV     R2,ITYPE        ;SAVE INPUT FILE TYPE FLAGS
     60 004764  010367  0000006                  MOV     R3,OTYPE        ;SAVE OUTPUT FILE TYPE FLAGS
```

```
                        .SBTTL  Initialize files
                ;
                ;  Do file initialization
                ;
004770          SETFIL: $LOCK                   ;LOCK USR IN MEMORY FOR SPEED
004772  004767  002144          CALL    INIFIL  ;DO FILE INITIALIZATION
                ;  Note: At this point R4 contains the sum of the input file sizes.
                ;
                ;  Delete any temp files that were left around from aborted runs.
                ;
004776              $ERASE  #FD1,#SPC2          ;TEMP FILE 1
005114              $ERASE  #FD1,#SPC3          ;TEMP FILE 2
                ;
                ;  Reopen 1st input file through FD1.
                ;
005232  012702  000006G         MOV     #FILES+6,R2     ;POINT TO SPC FOR 1ST INPUT FILE
005236              $REOPEN #FD1,(R2)           ;REOPEN 1ST INPUT FILE
005322  005722          TST     (R2)+           ;POINT TO ENTRY FOR NEXT FILE TO OPEN
005324  010267  000000G         MOV     R2,CURSAV       ;SET CURSAV TO POINT TO SPC FOR NEXT FILE
                ;
                ;  Set up record length information.
                ;
005330  005767  000000G         TST     RECLEN          ;WAS MAPPED RECORD LENGTH SPECIFIED?
005334  001003          BNE     2$              ;BR IF YES
005336  016767  000000G 000000G         MOV     INRLEN,RECLEN   ;DEFAULT TO INPUT RECORD LENGTH IF NOT
005344  016701  000000G 2$:     MOV     RECLEN,R1       ;GET SORT RECORD LENGTH
005350  005201          INC     R1              ;ROUND UP TO WORD BOUNDARY
005352  042701  000001          BIC     #1,R1
005356  010167  000000G         MOV     R1,RWDLEN       ;SAVE THIS AS SORT RECORD BUFFER SIZE
                ;
                ;  Determine how much space to allocate for temp files.
                ;  (Note: R4 now contains sum of # blocks in input files)
                ;
005362  032767  000000G 000000G         BIT     #RT$OX,ITYPE    ;IS INPUT FILE ORGANIZATION INDEXED?
005370  001430          BEQ     1$              ;BR IF NOT
                ;
                ;  Indexed organization file.
                ;  Use the following formula to determine temp file size:
                ;  Space = 1.2*(#-input-records*output-record-length)/512.+30.
                ;
005372  016704  000000G         MOV     MXREC1,R4       ;GET # INPUT RECORDS
005376  016705  000000G         MOV     MXREC2,R5
005402  016700  000000G         MOV     RECLEN,R0       ;GET OUTPUT RECORD LENGTH
005406  004767  000000G         CALL    MULTPY          ;MULTIPLY
005412  012700  000014          MOV     #12.,R0         ;NOW MULTIPLY BY 12 (1.2*10)
005416  004767  000000G         CALL    MULTPY
005422  012700  012000          MOV     #5120.,R0       ;DIVIDE BY 512.*10.
005426  004767  000000G         CALL    DIVIDE
005432  010402          MOV     R4,R2           ;GET LOW-ORDER RESULT VALUE
005434  062702  000036          ADD     #30.,R2         ;ADD 30 BLOCKS
005440  005505          ADC     R5              ;PROPOGATE CARRY AND CHECK FOR HIGH-ORDER NON-ZERO
005442  001433          BEQ     OPNTMP          ;BR IF NEED LESS THAN 65K BLOCKS
005444  012702  177777          MOV     #-1,R2          ;ASK FOR 65K BLOCKS
005450  000430          BR      OPNTMP
                ;
                ;  File organization is not Indexed.
                ;  Use the following formula to calculate how much temp file space to request:
```

Initialize files

```
                                        ;  Space = 1.2*(input-file-size)*(output-rec-len/input-rec-len)+30.
                                        ;
005452  005005                  1$:     CLR     R5              ;CLEAR HIGH-ORDER WORD
005454  012700  000014                  MOV     #12.,R0         ;MULTIPLY FILE SIZE BY 12.
005460  004767  000000G                 CALL    MULTPY
005464  016700  000000G                 MOV     RECLEN,R0       ;NOW MULTIPLY BY OUTPUT RECORD LENGTH
005470  004767  000000G                 CALL    MULTPY
005474  012700  000012                  MOV     #10.,R0         ;NOW DIVIDE BY 10.
005500  004767  000000G                 CALL    DIVIDE
005504  016700  000000G                 MOV     INRLEN,R0       ;NOW DIVIDE BY INPUT RECORD LENGTH
005510  004767  000000G                 CALL    DIVIDE
005514  010402                          MOV     R4,R2           ;GET LOW-ORDER PART OF RESULT
005516  062702  000050                  ADD     #40.,R2         ;NEED 30 EXTRA BLOCKS
005522  005505                          ADC     R5              ;PROPOGATE CARRY & TEST HIGH-ORDER PART
005524  001402                          BEQ     OPNTMP          ;BR IF AMT NEEDED FITS IN 16-BITS
005526  012702  177777                  MOV     #-1,R2          ;ASK FOR LARGEST POSSIBLE SPACE
                                        ;
                                        ;  Open temp files.
                                        ;  At this point, R2 contains total # blocks we want for temp file space.
                                        ;  Try to get entire temp file space in temp file # 1.
                                        ;
005532  005067  000000G         OPNTMP: CLR     T2SIZ           ;SAY 2ND TEMP FILE IS NOT USED YET
005536                                  $NEW    #FD2,#SPC2,R2   ;CREATE 1ST TEMP FILE
005672  103064                          BCC     5$              ;BRANCH IF OPEN WAS OK
                                        ;  Error occured on the open attempt.
                                        ;  Try to open to largest available space.
005674                                  $NEW    #FD2,#SPC2,#-1
006032  103004                          BCC     5$              ;BR IF OPEN SUCCESSFUL
006034                          9$:     XXX     9               ;COULD NOT OPEN FILE
006044  010067  000000G         5$:     MOV     R0,T1SIZ        ;SAVE SIZE OF TEMP FILE 1
006050  001771                          BEQ     9$              ;ERROR IF NO SPACE IN FILE 1
006052  020002                  TRYT2:  CMP     R0,R2           ;DID WE GET ENOUGH SPACE?
006054  103402                          BLO     2$              ;BR IF NOT
006056  000167  000442                  JMP     6$              ;BR IF GOT ENOUGH SPACE
006062  160002                  2$:     SUB     R0,R2           ;FIND OUT HOW MUCH WE LACK
                                        ;
                                        ;  REQUEST REMAINING SPACE NEEDED FOR TEMP FILE # 2
                                        ;
006064                                  $NEW    #FD4,#SPC3,R2   ;TRY TO GET NEEDED SPACE
006220  103137                          BCC     7$              ;BR IF GOT IT
006222                                  $NEW    #FD4,#SPC3,#-1  ;GET LARGEST AVAILABLE SPACE
006360  103402                          BCS     1$              ;BR IF COULDN'T GET ANY
006362  005700                          TST     R0              ;DID WE REALLY GET SOME SPACE?
006364  001055                          BNE     7$              ;BR IF YES
006366                          1$:     $PURGE  #FD4            ;RELEASE 2ND TEMP FILE
006516  005000                          CLR     R0              ;SAY ITS SIZE IS ZERO
006520  010067  000000G         7$:     MOV     R0,T2SIZ        ;SAVE SIZE OF TEMP FILE 2
006524                          6$:     $UNLOCK                 ;UNLOCK USR
```

```
                                        .SBTTL  Generate code to do key comparisons
                                ;-------------------------------------------------------------------------
                                ;
                                ;  GENERATE CODE TO DO KEY COMPARISONS
                                ;
                                ;  1ST STEAL ENOUGH CORE FROM TOP OF MEM FOR COMPARE ROUTINE.
006526  005002                  CODGEN: CLR     R2              ;INDEX TO 1ST KEY
006530  016701  000000G                MOV     TOPMEM,R1       ;POINT TO CURRENT TOP OF MEMORY
006534  016203  000000G          1$:   MOV     KTYPE(R2),R3    ;GET TYPE OF NEXT KEY
006540  166301  003566'                SUB     CODSIZ(R3),R1   ;LEAVE ROOM FOR GENERATED CODE
006544  062702  000002                ADD     #2,R2           ;GO CHECK NEXT KEY
006550  120267  000000G                CMPB    R2,NUMKY2       ;DONE ALL?
006554  103767                       BLO     1$              ;BR IF NOT
006556  162701  000036                SUB     #30.,R1         ;LEAVE ROOM FOR EXTRA CODE
                                ;
                                ;  STORE CODE FOR WINNER ROUTINE.
                                ;
006562  010167  000000G                MOV     R1,TOPMEM       ;SAVE NEW POINTER TO TOP OF MEMORY
006566  005721                       TST     (R1)+           ;POINT TO START OF CODE AREA
006570  004767  002634                CALL    GENWIN          ;GENERATE WINNER CODE
006574  010167  000000G                MOV     R1,COMPAR       ;START OF COMPARISON CODE
006600  012767  000001  003512'       MOV     #1,RSTAT        ;REGS R1&R2 ARE LOADED FOR ASCENDING KEYS
                                ;  GENERATE CODE TO SAVE INITIAL RECORD POINTERS
006606  012721  010137                MOV     #010137,(R1)+   ;[MOV R1,@#REC1]
006612  012721  000000G                MOV     #REC1,(R1)+     ;[REC1]
006616  012721  010237                MOV     #010237,(R1)+   ;[MOV R2,@#REC2]
006622  012721  000000G                MOV     #REC2,(R1)+     ;[REC2]
                                ;  NOW GENERATE COMPARE CODE FOR EACH KEY.
006626  005002                  4$:   CLR     R2              ;INIT KEY INDEX
006630  016203  000000G          3$:   MOV     KTYPE(R2),R3    ;GET KEY TYPE
006634  004773  003610'                CALL    @GENVEC(R3)     ;GENERATE CODE FOR THIS TYPE OF KEY
006640  062702  000002                ADD     #2,R2           ;MOVE ON TO NEXT KEY
006644  120267  000000G                CMPB    R2,NUMKY2       ;DONE ALL?
006650  103011                       BHIS    2$              ;BR IF FINISHED
                                ;  SEE IF WE HAVE GOTTEN TOO FAR FOR A BR INSTRUCTION TO WORK
006652  010103                       MOV     R1,R3           ;FIND DISTANCE FROM RETURN CODE
006654  166703  003510'                SUB     RETLOC,R3
006660  020327  000346                CMP     R3,#230.        ;GOTTEN TOO FAR?
006664  101761                       BLOS    3$              ;BR IF OK
006666  004767  002536                CALL    GENWIN          ;GENERATE NEW WINNER ROUTINES
006672  000756                       BR      3$
                                ;  GENERATE [RTS  PC] FOR EQUAL KEY COMPARISONS
006674  012721  000207          2$:   MOV     #207,(R1)+      ;[RTS PC]
                                ;
                                ;  SEE IF WE NEED TO RESERVE ROOM FOR AN EOF RECORD.
                                ;
006700  005767  000000G          SEOFR:  TST     EOFCOL          ;IS EOF RECORD EXPECTED?
006704  001423                       BEQ     SETBLK          ;BR IF NOT
006706  016704  000000G                MOV     TOPMEM,R4       ;GET CURRENT TOP OF MEMROY
006712  166704  000000G                SUB     RECLEN,R4       ;RESERVE ROOM FOR EOF RECORD
006716  042704  000001                BIC     #1,R4           ;MAKE SURE EVEN WORD
006722  010467  000000G                MOV     R4,EOFBUF       ;PLACE EOF RECORD HERE
006726  005744                       TST     -(R4)
006730  010467  000000G                MOV     R4,TOPMEM
                                ;  INITIALIZE EOF RECORD TO ALL EOF CHARS
006734  005724                       TST     (R4)+
006736  116703  000000G                MOVB    EOFCHR,R3
```

Generate code to do key comparisons

```
     58 006742  016702  0000006                 MOV     RECLEN,R2       ;LENGTH OF RECORD
     59 006746  110324                  1$:     MOVB    R3,(R4)+        ;FORM DUMMY EOF RECORD
     60 006750  005302                          DEC     R2
     61 006752  001375                          BNE     1$
```

```
                                        .SBTTL  Determine buffer sizes
                                ;-------------------------------------------------------------------------------
                                ;
                                ;  DETERMINE BUFFER SIZES
                                ;
006754  016704  000000G        SETBLK: MOV     BLKFC,R4        ;GET FILE BLOCKING FACTOR
006760  001037                        BNE     8$              ;BR IF USER SPECIFIED A VALUE
006762  016703  000000G                MOV     TOPMEM,R3       ;TOP OF FREE MEMORY
006766  012702  000000G                MOV     #P2TOP,R2       ;R2 -> TOP OF PASS 2
006772  022702  000000G                CMP     #P1TOP,R2       ;IS TOP OF PASS 1 HIGHER ??
006776  101402                        BLOS    9$              ;BRANCH IF NOT
007000  012702  000000G                MOV     #P1TOP,R2       ;USE HIGHER OF THE TWO
007004                          9$:
007004  160203                        SUB     R2,R3           ;GET SPACE AVAILABLE TO PASS 2
                                ;  MAKE SURE WE HAVE MIN SIZE
007006  020327  005000                CMP     R3,#<5*BUFSIZ>  ;GOT ENOUGH FOR SORT?
007012  103004                        BHIS    10$             ;BR IF YES
007014                                XXX     24              ;NOT ENOUGH MEMORY
                                ;  SET BLOCKING FACTOR SO THAT THERE IS ROOM FOR AT LEAST
                                ;  5 BUFFERS.
007024  012702  012000          10$:    MOV     #<2*5*BUFSIZ>,R2;TRY BLOCKING FACTOR OF 2
007030  012704  000001                MOV     #1,R4
007034  020203                  7$:     CMP     R2,R3           ;THIS SIZE OK
007036  101006                        BHI     6$              ;BR IF TOO BIG
007040  062702  005000                ADD     #<5*BUFSIZ>,R2  ;INCREASE BLOCKING FACTOR
007044  005204                        INC     R4
007046  020427  000005                CMP     R4,#5           ;GO TO MAX BLOCKING FACTOR OF 5
007052  103770                        BLO     7$
                                ;  NOW GET BUFFER SIZE.
007054  010467  000000G         6$:    MOV     R4,BLKFC        ;SAVE BLOCKING FACTOR
007060  005001                  8$:    CLR     R1
007062  062701  001000          1$:    ADD     #BUFSIZ,R1      ;CALCULATE BUFFER SIZE
007066  005304                        DEC     R4
007070  001374                        BNE     1$
007072  010167  000000G                MOV     R1,OBUFSZ       ;SAVE BUFFER SIZE
007076  006201                        ASR     R1              ;SAVE WORD SIZE OF BUFFER
007100  010167  000000G                MOV     R1,OWCNT
                                ;
                                ;  ENTER PHASE 1 OF SORT
                                ;
007104  012767  000000G 000000G        MOV     #P2TOP,LOWMEM   ;ASSUME TOP OF PASS 2 IS LOWEST
007112                                $GETBOT R2,#P1TOP       ;SEE IF WE CAN HAVE TOP OF PASS 1
007132  010267  000000G                MOV     R2,LOWMEM       ;LOWMEM = P2TOP OR P1TOP
007136  000167  000000G                JMP     START1
```

Determine buffer sizes

```
                                .IF     DF RT11
                                        .SBTTL  Set up information about file specs
                                ;-------------------------------------------------------------------------------
                                ;  INIFIL is called to process the file name portion of the command line.
                                ;  It parses the file specs and creates file blocks containing Rad50
                                ;  file names, sizes, etc.
                                ;
                                ;  Inputs:
                                ;   FILBUF = Asciz CSI file spec (i.e., out,temp=in1,in2,...)
                                ;
                                ;  Outputs:
                                ;   FILES = Vector of pointers to file specs. (last pointer is zero).
                                ;   R4 = Sum of input file sizes.
                                ;
007142  010146                  INIFIL: MOV     R1,-(SP)
007144  010246                          MOV     R2,-(SP)
007146  010346                          MOV     R3,-(SP)
007150  010546                          MOV     R5,-(SP)
                                ;
                                ;  Determine proper default file extensions.
                                ;
007152  016702  000000G                 MOV     ITYPE,R2        ;GET INPUT FILE TYPE FLAGS
007156  004767  004244                  CALL    GETDFX          ;GET DEFAULT INPUT FILE EXTENSION
007162  010267  000064'                 MOV     R2,DEFEXT       ;SET DEFAULT INPUT FILE EXTENSION
007166  016702  000000G                 MOV     OTYPE,R2        ;GET OUTPUT FILE TYPE FLAGS
007172  004767  004230                  CALL    GETDFX          ;GET DEFAULT EXTENSION
007176  010267  000066'                 MOV     R2,DEFEXT+2     ;SET DEFAULT OUTPUT FILE EXTENSION
                                ;
                                ;  Use .CSISPC to parse command line.
                                ;
007202  010605                          MOV     SP,R5           ;PRESERVE STACK POINTER
007204                                  .CSISPC #RSPCS,#DEFEXT,#FILBUF  ;PARSE COMMAND LINE
007222  103004                          BCC     2$              ;BR IF NO ERROR
007224                                  XXX     19              ;INVALID COMMAND LINE
007234  010506                  2$:     MOV     R5,SP           ;RESET STACK POINTER
007236  004767  001536                  CALL    SVRSTS          ;SAVE (POSSIBLY) ANY RSTS INFORMATION
                                ;
                                ;  Set addresses in FILES vector to point to file specifications.
                                ;
007242  012701  000000G                 MOV     #FILES,R1       ;POINT TO FIRST CELL IN POINTER VECTOR
007246  012731  000000G                 MOV     #RSPCS,@(R1)+   ;OUTPUT FILE
007252  012731  000012G                 MOV     #RSPCS+10.,@(R1)+;1ST TEMP FILE
007256  012731  000024G                 MOV     #RSPCS+20.,@(R1)+;2ND TEMP FILE
007262  012731  000036G                 MOV     #RSPCS+30.,@(R1)+;1ST INPUT FILE
007266  012731  000046G                 MOV     #RSPCS+38.,@(R1)+;2ND INPUT FILE
007272  012731  000056G                 MOV     #RSPCS+46.,@(R1)+;3RD INPUT FILE
007276  012731  000066G                 MOV     #RSPCS+54.,@(R1)+;4TH INPUT FILE
007302  012731  000076G                 MOV     #RSPCS+62.,@(R1)+;5TH INPUT FILE
007306  012731  000106G                 MOV     #RSPCS+70.,@(R1)+;6TH INPUT FILE
                                ;
                                ;  See if this is an in-place sort.
                                ;  Specified by omitting the output file spec.
                                ;
007312  005777  000000G                 TST     @SPC1           ;OUTPUT FILE SPECIFIED?
007316  001013                          BNE     5$              ;BR IF YES
007320  032767  000000G 000000G         BIT     #RT$OX,ITYPE    ;IS INPUT FILE ORGANIZATION INDEXED?
007326  001404                          BEQ     9$              ;BR IF NOT
```

Set up information about file specs

```
                                        XXX     37              ;ERROR -- INPLACE SORT NOT LEGAL WITH INDEXED FILE
007340  052767  000000G 000000G 9$:     BIS     #F$INPL,SFLAGS  ;REMEMBER THIS IS AN IN-PLACE SORT
                                ;
                                ;  Set up temp file names.
                                ;
007346  016702  000000G         5$:     MOV     SPC2,R2         ;GET ADDRESS OF TEMP FILE BLOCK
007352  005712                          TST     (R2)            ;DID USER SPECIFY THE TEMP FILE DEVICE?
007354  001002                          BNE     8$              ;BR IF YES
007356  016712  000034'                 MOV     R50DK,(R2)      ;DEFAULT TO DK:
007362  016703  000000G         8$:     MOV     SPC3,R3         ;POINT TO 2ND TEMP FILE SPEC AREA
007366  005713                          TST     (R3)            ;WAS DEVICE SPECIFIED FOR 2ND TEMP FILE?
007370  001001                          BNE     1$              ;BR IF YES
007372  011213                          MOV     (R2),(R3)       ;USE SAME DEVICE AS FOR 1ST TEMP FILE
007374  016762  000036' 000002  1$:     MOV     R50TN0,2(R2)    ;SET DEFAULT TEMP FILE NAME
007402  016762  000040' 000004          MOV     R50TN1,4(R2)
007410  016763  000036' 000002          MOV     R50TN0,2(R3)    ;SET NAME FOR 2ND TEMP FILE
007416  016763  000042' 000004          MOV     R50TN2,4(R3)
                                ;
                                ;  If running under TSX, generate extension of the form XXz where "z"
                                ;  is a function of the line number.
                                ;
                                ;  If running under RSTS, generate extension of the form XXz where "z"
                                ;  if a function of the job number.
                                ;
007424  105767  000000G                 TSTB    RSTS            ;RUNNING UNDER RSTS?
007430  001404                          BEQ     6$              ;BR IF NOT
007432  113700  000404                  MOVB    @#RSTSJB,R0     ;GET RSTS JOB # TIMES 2
007436  006200                          ASR     R0              ; TIMES 1
007440  000403                          BR      7$
007442  116700  000000G         6$:     MOVB    TSXFL,R0        ;GET TSX LINE NUMBER
007446  001407                          BEQ     3$              ;BR IF RUNNING DIRECTLY UNDER RT11
007450  066700  000044'         7$:     ADD     R50XX,R0        ;FORM EXTENSION NAME
007454  010062  000006                  MOV     R0,6(R2)        ;SET AS EXTENSION OF TEMP FILE 1
007460  010063  000006                  MOV     R0,6(R3)        ;AND TEMP FILE 2
007464  000406                          BR      4$
                                ;  Running directly under RT-11 -- Use "TMP" as temp file extension
007466  016762  000046' 000006  3$:     MOV     R50TMP,6(R2)    ;SET TMP AS TEMP FILE EXTENSIONS
007474  016763  000046' 000006          MOV     R50TMP,6(R3)
                                ;
                                ;  Load device handlers.
                                ;
007502  016705  000000G         4$:     MOV     TOPMEM,R5       ;GET ADDRESS OF TOP OF FREE MEMORY
007506  012702  000000G                 MOV     #FILES,R2       ;POINT TO FILE DESCRIPTOR POINTER VECTOR
007512  013201                  10$:    MOV     @(R2)+,R1       ;GET ADDRESS OF RT-11 FILE SPEC
007514  004767  003574                  CALL    LOADIT          ;LOAD DEVICE HANDLER
007520  020227  000022G                 CMP     R2,#FILES+22    ;HAVE WE DONE ALL?
007524  103772                          BLO     10$             ;BR IF NOT
007526  005745                          TST     -(R5)
007530  010567  000000G                 MOV     R5,TOPMEM       ;SAVE NEW TOP OF MEMORY ADDRESS
                                ;
                                ;  If this is a COBOL-Plus Indexed organization file, read in home block
                                ;  and set up information about data files.
                                ;
007534  032767  000000G 000000G         BIT     #RT$OX,ITYPE    ;IS THIS A COBOL ISAM FILE?
007542  001402                          BEQ     OPNDAT          ;BR IF NOT
007544  004767  000252                  CALL    INIX            ;DO INITIALIZATION FOR ISAM FILE
                                ;
```

Set up information about file specs

```
                                        ;  Open each file, Add up number of blocks and do .SAVESTATUS
                                        ;
007550  012702  000006G      OPNDAT: MOV     #FILES+6,R2     ;POINT TO SPEC FOR 1ST INPUT FILE
007554  012705  000000G              MOV     #SAVBUF,R5      ;DO SAVESTATUS INTO THIS AREA
007560  005004                      CLR     R4              ;ACCUMULATE FILE SIZES IN R4
007562  011201              3$:     MOV     @R2,R1          ;GET POINTER TO SPC FOR FILE
007564  001511                      BEQ     1$              ;BR IF REACHED END
007566  005731                      TST     @(R1)+          ;IS FILE SPEC NON-ZERO?
007570  001507                      BEQ     1$              ;IF ZERO THEN FILE WAS NOT SPECIFIED
007572                              $OLD    #FD1,(R2)       ;TRY TO OPEN INPUT FILE
007714  103004                      BCC     2$              ;BR IF OPEN WAS SUCCESSFUL
007716                              XXX     8               ;CANNOT OPEN INPUT FILE
007726  060004              2$:     ADD     R0,R4           ;ACCUMULATE FILE SIZES
007730  010532                      MOV     R5,@(R2)+       ;MAKE SPC POINT TO SAVESTATUS AREA
007732                              $SAVEST #FD1            ;SAVE FILE STATUS
010002  062705  000012              ADD     #12,R5          ;INCREMENT TO THE NEXT SAVESTATUS AREA
010006  000665                      BR      3$              ;GO OPEN REST OF INPUT FILES
                                        ;
                                        ;  Finished:  Return accumulated file size in R4.
                                        ;
010010  012605              1$:     MOV     (SP)+,R5
010012  012603                      MOV     (SP)+,R3
010014  012602                      MOV     (SP)+,R2
010016  012601                      MOV     (SP)+,R1
010020  000207                      RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  INIX is called to do initialization for COBOL-Plus Indexed organization
                                ;  files.  It reads the file home block and sets up the RAD50 file specs
                                ;  for the data files in the RSPCS area.
                                ;
                                ;  Inputs:
                                ;   TOPMEM = Top of free memory address.
                                ;
                                ;  Outputs:
                                ;   TOPMEM = Set to new top of free memory address.
                                ;   MXREC1-MXREC2 = Max # of records to sort.
                                ;
010022  010146                  INIX:   MOV     R1,-(SP)
010024  010246                          MOV     R2,-(SP)
010026  010346                          MOV     R3,-(SP)
010030  010446                          MOV     R4,-(SP)
010032  010546                          MOV     R5,-(SP)
                                ;
                                ;  Make sure he only specified one input file.
                                ;
010034  005777  000000G                 TST     @SPC5           ;IS THERE A SECOND INPUT FILE?
010040  001404                          BEQ     10$             ;BR IF NOT
010042                                  XXX     38              ;ERROR IF YES
                                ;
                                ;  Open ISAM index file and read in home block.
                                ;
010052                          10$:    $OLD    #FD1,#SPC4      ;OPEN FIRST INPUT FILE
010176  103004                          BCC     1$              ;BR IF OPEN OK
010200                                  XXX     8               ;CAN'T OPEN INPUT FILE
010210                          1$:     $READ   #FD1,#GENBUF,#256.,#0 ;READ HOME BLOCK
010370  103447                          BCS     2$              ;BR IF READ ERROR
010372                                  $WAIT   #FD1            ;WAIT FOR READ TO FINISH
010506  103004                          BCC     3$              ;BR IF READ OK
010510                          2$:     XXX     14              ;READ ERROR
010520                          3$:     $KEEP   #FD1            ;CLOSE INDEX FILE
                                ;
                                ;  Extract information from home block.
                                ;
                                ;  Set number of records in file.
                                ;
010566  012701  002432'                 MOV     #GENBUF,R1      ;POINT TO HOME BLOCK BUFFER
010572  016102  000032                  MOV     HB$RIU(R1),R2   ;GET LOW-ORDER NUMBER OF RECORDS IN FILE
010576  016103  000034                  MOV     HB$RIU+2(R1),R3 ;GET HIGH-ORDER NUMBER OF RECORDS IN FILE
010602  020367  000000G                 CMP     R3,MXREC2       ;COMPARE HIGH-ORDER WITH PREVIOUSLY SPECIFIED MAX
010606  103404                          BLO     4$              ;BR IF WE SHOULD USE ISAM # RECORDS
010610  101007                          BHI     5$              ;BR IF PREVIOUS MAX IS SMALLER
010612  020267  000000G                 CMP     R2,MXREC1       ;COMPARE LOW-ORDER
010616  103004                          BHIS    5$              ;BR IF PREVIOUS VALUE IS ONE TO USE
010620  010267  000000G                 MOV     R2,MXREC1       ;SET # RECORDS TO BE READ
010624  010367  000000G                 MOV     R3,MXREC2
                                ;
                                ;  Set up information about record length.
                                ;
010630  016103  000004          5$:     MOV     HB$RSZ(R1),R3   ;GET THE LOGICAL RECORD LENGTH (BYTES)
010634  010367  000000G                 MOV     R3,ISMARS       ;THIS IS FULL RECORD LENGTH LESS LEADING CONTROL WORD
010640  116100  000001                  MOVB    HB$DUP(R1),R0   ;GET # OF KEYS THAT ALLOW DUPLICATES
010644  006300                          ASL     R0              ;4 BYTES FOR EACH ACCESSION # FOR THESE KEYS
```

Set up information about file specs

```
     58 010646  006300                        ASL     R0                      ;
     59 010650  160003                        SUB     R0,R3                   ;GET LENGTH OF ACTUAL DATA PORTION OF RECORD
     60 010652  010367  000000G               MOV     R3,INRLEN               ;THIS IS THE LOGICAL RECORD LENGTH
     61                                ;
     62                                ;  Set up names of data files and load device handlers.
     63                                ;
     64 010656  016104  000002                MOV     HB$DFS(R1),R4           ;GET # DATA FILES
     65 010662  016103  000030                MOV     HB$DIP(R1),R3           ;OFFSET WITHIN HOME BLOCK OF DATA FILE INFO
     66 010666  060103                        ADD     R1,R3                   ;GET ADDRESS OF DATA FILE INFO BLOCKS
     67 010670  016705  000000G               MOV     TOPMEM,R5               ;GET ADDRESS OF TOP OF FREE MEMORY
     68 010674  012701  000036G               MOV     #RSPCS+30.,R1           ;POINT TO 1ST INPUT FILE SPEC
     69 010700  011167  003514'               MOV     (R1),INXDEV             ;SAVE NAME OF DEVICE WITH INDEX FILE # 0
     70 010704  016702  000062'               MOV     R50CDO,R2               ;GET EXTENSION FOR 1ST DATA FILE (CDO)
     71 010710  016300  000000        6$:     MOV     HD$DEV(R3),R0           ;GET NAME OF DEVICE WITH DATA FILE
     72 010714  001003                        BNE     7$                      ;BR IF EXPLICIT DEVICE SPECIFIED FOR FILE
     73 010716  016711  003514'               MOV     INXDEV,(R1)             ;DEFAULT TO SAME DEVICE AS INDEX FILE
     74 010722  000403                        BR      8$
     75 010724  010011                7$:     MOV     R0,(R1)                 ;SET DEVICE FOR DATA FILE
     76 010726  004767  002362                CALL    LOADIT                  ;LOAD DEVICE HANDLER
     77 010732  005721                8$:     TST     (R1)+                   ;POINT TO FILE NAME AREA
     78 010734  012700  000040G               MOV     #RSPCS+32.,R0           ;POINT TO ORIGINAL FILE NAME
     79 010740  012021                        MOV     (R0)+,(R1)+             ;COPY FILE NAME TO DATA FILE SPEC
     80 010742  012021                        MOV     (R0)+,(R1)+
     81 010744  010221                        MOV     R2,(R1)+                ;SET FILE EXTENSION
     82 010746  005202                        INC     R2                      ;INCREMENT EXTENSION
     83 010750  062703  000004                ADD     #HD$$SZ,R3              ;POINT TO NEXT DATA FILE INFO BLOCK
     84 010754  005304                        DEC     R4                      ;ARE THERE MORE DATA FILES?
     85 010756  001354                        BNE     6$                      ;BR IF YES
     86 010760  010567  000000G               MOV     R5,TOPMEM               ;SAVE NEW TOP OF MEMORY ADDRESS
     87                                ;
     88                                ;  Finished
     89                                ;
     90 010764  012605                        MOV     (SP)+,R5
     91 010766  012604                        MOV     (SP)+,R4
     92 010770  012603                        MOV     (SP)+,R3
     93 010772  012602                        MOV     (SP)+,R2
     94 010774  012601                        MOV     (SP)+,R1
     95 010776  000207                        RETURN
```

```
                        .SBTTL  RSTS oriented subroutines
                ;
                ; SVRSTS
                ;
                ;       Save the RSTS oriented file parameters after a .CSISPC file
                ;       decode.
                ;
                ;       When running under the RT11 run-time of the RSTS operating
                ;       system, there are a number of RSTS oriented file parameters
                ;       that can be present in a file specification which have no
                ;       meaning for RT11.  These include the Project-Programmer Number
                ;       (PPN), Protection Code, Mode, and Clustersize.  Fortunately
                ;       (and not documented outside of DEC), the RT11 run-time saves
                ;       these parameters in a job's "Read/Write Stack" after a .CSISPC
                ;       decode is performed.  This read/write stack is the job-dependent
                ;       data area that the RT11 run-time maintains for each job.  It
                ;       is created in the high part of the user memory allocated to
                ;       the running job, is about 3600 bytes long (for RSTS V06C),
                ;       and is at all times pointed at by the low core RMON word
                ;       (location 54).  The beginning of the saved parameters is
                ;       CSPSOF bytes into the R/W stack with each channel's information
                ;       occupying 20 bytes.
                ;
                ;       Also present in the R/W stack are four "one shot" cells
                ;       which are active for one file oriented operation (.LOOKUP,
                ;       .ENTER, etc.).  These four words are for the PPN, protection
                ;       code, mode, and clustersize, respectively, and reside at
                ;       the immediate beginning of the R/W stack.  Since the second
                ;       through fifth words of the saved .CSISPC parameters (see
                ;       above) are also these four items, these items can be saved
                ;       off after a .CSISPC call and then loaded into the one-shot
                ;       area immediately preceeding a file operation.
                ;
                ;       This routine does the saving of the RSTS-oriented paramters,
                ;       while the SETNRT routine processes the loading of the one-
                ;       shots.
                ;
011000  105767  000000G SVRSTS: TSTB    RSTS            ;We running under RSTS?
011004  001436                  BEQ     10$             ; => no
011006  010046                  MOV     R0,-(SP)        ;Get some work registers
011010  010146                  MOV     R1,-(SP)
011012  010246                  MOV     R2,-(SP)
011014  010346                  MOV     R3,-(SP)
011016  010446                  MOV     R4,-(SP)

011020  013700  000054          MOV     @#RMON,R0       ;Point to the initial block
011024  062700  000400          ADD     #CSPCOF+2,R0
011030  012704  000000G         MOV     #FILES,R4       ;Point to vector of file descriptor addresses
011034  012702  000011          MOV     #9.,R2          ;And set a counter

011040  012401          5$:     MOV     (R4)+,R1        ;Get address of next file descriptor block
011042  062701  000002          ADD     #2,R1           ;Point to area where we save RSTS data
011046  010003                  MOV     R0,R3           ;Get a move from pointer
011050  012321                  MOV     (R3)+,(R1)+     ;Save PPN
011052  012321                  MOV     (R3)+,(R1)+     ;       Protection code
011054  012321                  MOV     (R3)+,(R1)+     ;       Mode
011056  012321                  MOV     (R3)+,(R1)+     ; and Clustersize
```

```
011060  062700  000020                ADD     #20,R0          ;Skip to the next block
011064  005302                        DEC     R2              ;And continue until done
011066  001364                        BNE     5$              ;Loop for all files

011070  012604                        MOV     (SP)+,R4
011072  012603                        MOV     (SP)+,R3        ;Restore the work registers
011074  012602                        MOV     (SP)+,R2
011076  012601                        MOV     (SP)+,R1
011100  012600                        MOV     (SP)+,R0

011102  000207                10$:    RETURN                  ;Return to the caller
                               .ENDC   ;RT-11
```

```
                                        .SBTTL  CODE GENERATOR SUBROUTINES
                                ;
                                ;-----------------------------------------------------------
                                ;  ASCII KEY
                                ;
011104  016203  000000G        GNASCI: MOV     KLEN(R2),R3     ;GET KEY LENGTH
011110  020327  000002                CMP     R3,#2           ;SEE IF SHORT KEY
011114  101003                        BHI     2$              ;BR IF LONG
011116  103420                        BLO     ONEASC          ;BR IF SINGLE ASCII CHARACTER
011120  000167  000062                JMP     TWOASC          ;BR IF 2-CHAR KEY
                                ;  GENERATE CODE FOR GENERAL ASCII STRING COMPARISON
011124  004767  000424        2$:     CALL    GENKPT          ;GEN CODE TO GET KEY POINTERS
011130  012721  012703                MOV     #012703,(R1)+   ;[MOV #LEN,R3]
011134  010321                        MOV     R3,(R1)+        ;[#LEN]
011136  012721  122122                MOV     #122122,(R1)+   ;[CMPB (R1)+,(R2)+]
011142  004767  000446        4$:     CALL    GENBRU          ;GEN [BNE RETURN]
011146  012721  005303                MOV     #5303,(R1)+     ;[DEC R3]
011152  012721  001374                MOV     #1374,(R1)+     ;[BNE .-8.]
011156  000207                        RETURN
                                ;
                                ;-----------------------------------------------------------
                                ;  GENERATE CODE FOR SINGLE ASCII CHARACTER
                                ;
011160  004767  000262        ONEASC: CALL    GENRPT          ;GET RECORD POINTERS
011164  012721  126162                MOV     #126162,(R1)+   ;[CMPB POS(R1),POS(R2)]
011170  016203  000000G               MOV     KSTRT(R2),R3    ;GET POS+1
011174  010321                        MOV     R3,(R1)+        ;[POS]
011176  010321                        MOV     R3,(R1)+        ;[POS]
011200  004767  000410                CALL    GENBRU          ;GEN BNE
011204  000207                        RETURN
                                ;
                                ;-----------------------------------------------------------
                                ;  GEN CODE FOR TWO CHARACTER ASCII KEYS
                                ;
011206  004767  000234        TWOASC: CALL    GENRPT          ;GET RECORD POINTERS
011212  012721  126162                MOV     #126162,(R1)+   ;[CMPB POS(R1),POS(R2)]
011216  016203  000000G               MOV     KSTRT(R2),R3    ;GET COLUMN #
011222  010321                        MOV     R3,(R1)+        ;[POS]
011224  010321                        MOV     R3,(R1)+        ;[POS]
011226  004767  000362                CALL    GENBRU          ;GEN [BNE RETURN]
011232  012721  126162                MOV     #126162,(R1)+   ;[CMPB POS+1(R1),POS+1(R2)]
011236  005203                        INC     R3              ;GET POS+1
011240  010321                        MOV     R3,(R1)+        ;[POS+1]
011242  010321                        MOV     R3,(R1)+        ;[POS+1]
011244  004767  000344                CALL    GENBRU          ;GEN [BNE RETURN]
011250  000207                        RETURN
                                ;
                                ;-----------------------------------------------------------
                                ;  GENERATE CODE FOR UNSIGNED 16-BIT BINARY INTEGER.
                                ;
011252  004767  000170        GNUSI:  CALL    GENRPT          ;GET RECORD POINTERS
011256  012721  026162                MOV     #026162,(R1)+   ;[CMP POS(R1),POS(R2)]
011262  016203  000000G               MOV     KSTRT(R2),R3    ;GET STARTING COL #
011266  010321                        MOV     R3,(R1)+        ;[POS]
011270  010321                        MOV     R3,(R1)+        ;[POS]
011272  004767  000316                CALL    GENBRU          ;GEN [BNE RETURN]
011276  052767  000000G 000000G       BIS     #F$BIN,SFLAGS   ;REMEMBER THIS IS A BINARY RECORD
```

```
011304  000207                          RETURN
                        ;-----------------------------------------------------------------------
                        ;
                        ;  Comp item stored with least significant word first.
                        ;
011306  012704  000000G        GNCMP:  MOV     #KYCOMP,R4      ;COMPARISON ROUTINE
011312  052767  000000G 000000G        BIS     #F$BIN,SFLAGS   ;REMEMBER RECORD HAS COMP FIELD
011320  000425                        BR      GNCALL
                        ;
                        ;  Leading separate sign.
                        ;
011322  012704  000000G        GNLSS:  MOV     #KYLSS,R4       ;COMPARISON ROUTINE
011326  000422                        BR      GNCALL
                        ;
                        ;  Trailing separate sign.
                        ;
011330  012704  000000G        GNTSS:  MOV     #KYTSS,R4       ;COMPARISON ROUTINE
011334  000417                        BR      GNCALL
                        ;
                        ;  Date fields in the form (mmddyy).
                        ;
011336  012704  000000G        GNDATE: MOV     #KYDATE,R4      ;COMPARISON ROUTINE
011342  000414                        BR      GNCALL
                        ;
                        ;  Binary values stored with the most significant word first.
                        ;
011344  012704  000000G        GNBIN:  MOV     #KYBIN,R4       ;COMPARISON ROUTINE
011350  052767  000000G 000000G        BIS     #F$BIN,SFLAGS   ;REMEMBER BINARY FIELD IN RECORD
011356  000406                        BR      GNCALL
                        ;
                        ;  Signed numeric display (trailing non-separate sign).
                        ;
011360  012704  000000G        GNDIS:  MOV     #KYDIS,R4       ;COMPARISON ROUTINE
011364  000403                        BR      GNCALL
                        ;
                        ;  Ascii characters with lower case translated to upper case.
                        ;
011366  012704  000000G        GNUCA:  MOV     #KYUCA,R4       ;COMPARISON ROUTINE
011372  000400                        BR      GNCALL

                        ;-----------------------------------------------------------------------
                        ;  Gen code to call a subroutine (whose address is in R4) to do a key
                        ;  comparison.
                        ;  On entry to the subroutine the following registers will be set up:
                        ;    R1 = Address of key field 1.
                        ;    R2 = Address of key field 2.
                        ;    R3 = # of bytes in key fields.
                        ;
011374  004767  000154  GNCALL: CALL    GENKPT          ;GEN CODE TO LOAD R1 & R2 WITH KEY POINTERS
011400  012721  012703          MOV     #012703,(R1)+   ;[MOV # bytes,R3]
011404  016221  000000G         MOV     KLEN(R2),(R1)+  ;[# bytes]
011410  012721  004737          MOV     #4737,(R1)+     ;[CALL @#routine]
011414  010421                  MOV     R4,(R1)+        ;[Routine]
011416  004767  000172          CALL    GENBRU          ;[BNE return]
011422  105067  003512'         CLRB    RSTAT           ;REGISTERS ARE MESSED UP
011426  000207                  RETURN
                        ;
```

```
                                ;-------------------------------------------------------------------
                                ;  GEN CODE TO SIGNAL WHICH RECORD WAS THE WINNER.
                                ;
011430  012721  000401          GENWIN: MOV     #401,(R1)+      ;[BR .+2]
011434  010167  003510'                  MOV     R1,RETLOC       ;SAVE LOCATION OF RTS PC INSTRUCTION
011440  012721  000207                   MOV     #207,(R1)+      ;[RTS PC]
011444  000207                          RETURN
                                ;
                                ;-------------------------------------------------------------------
                                ;  GEN CODE TO LOAD R1 & R2 WITH POINTERS TO START OF RECORDS
                                ;  IF THEY ARE NOT ALREADY LOADED CORRECTLY.
                                ;
011446  005762  000000G         GENRPT: TST     KAD(R2)         ;NEEDED FOR ASCENDING OR DESCENDING?
011452  001020                          BNE     1$              ;BR IF DESCENDING
                                ;  GEN OF ASCENDING KEY
011454  026727  003512' 000001          CMP     RSTAT,#1        ;ARE THEY LOADED CORRECTLY NOW?
011462  001413                          BEQ     2$              ;BR IF YES
011464  012767  000001  003512'         MOV     #1,RSTAT        ;SAY THEY ARE LOADED CORRECTLY
011472  012721  013701                   MOV     #013701,(R1)+   ;[MOV @#REC1,R1]
011476  012721  000000G                  MOV     #REC1,(R1)+     ;[REC1]
011502  012721  013702                   MOV     #013702,(R1)+   ;[MOV @#REC2,R2]
011506  012721  000000G                  MOV     #REC2,(R1)+     ;[REC2]
011512  000207                  2$:     RETURN
                                ;  GEN CODE FOR DESCENDING KEYS
                                ;  (REVERSE ORDER OF RECORDS)
011514  026727  003512' 000002  1$:     CMP     RSTAT,#2        ;ARE REGS ALREADY LOADED?
011522  001413                          BEQ     3$              ;BR IF YES
011524  012767  000002  003512'         MOV     #2,RSTAT        ;SAY THEY ARE LOADED
011532  012721  013701                   MOV     #013701,(R1)+   ;[MOV @#REC2,R1]
011536  012721  000000G                  MOV     #REC2,(R1)+     ;[REC2]
011542  012721  013702                   MOV     #013702,(R1)+   ;[MOV @#REC1,R2]
011546  012721  000000G                  MOV     #REC1,(R1)+     ;[REC1]
011552  000207                  3$:     RETURN
                                ;
                                ;-------------------------------------------------------------------
                                ;  GEN CODE TO LOAD R1 & R2 WITH POINTERS TO START OF KEYS
                                ;
011554  010546                  GENKPT: MOV     R5,-(SP)
011556  004767  177664                  CALL    GENRPT          ;LOAD R1 & R2 WITH RECORD POINTERS
011562  016205  000000G                 MOV     KSTRT(R2),R5    ;GET STARTING COL #
011566  001406                          BEQ     9$              ;BR IF AT FRONT OF RECORD
011570  012721  062701                   MOV     #062701,(R1)+   ;[ADD #POS,R1]
011574  010521                          MOV     R5,(R1)+        ;[POS]
011576  012721  062702                   MOV     #062702,(R1)+   ;[ADD #POS,R2]
011602  010521                          MOV     R5,(R1)+        ;[POS]
011604  005067  003512'         9$:     CLR     RSTAT           ;SAY R1-R2 ARE MESSED UP
011610  012605                          MOV     (SP)+,R5
011612  000207                          RETURN
                                ;
                                ;-------------------------------------------------------------------
                                ;  Generate code for BNE following a test which leaves
                                ;  the condition codes correctly set for BGT/BLT/BEQ.
                                ;
011614  010346                  GENBRU: MOV     R3,-(SP)
011616  012721  001000                  MOV     #001000,(R1)+   ;[BNE]
011622  010103                          MOV     R1,R3
011624  166703  003510'                 SUB     RETLOC,R3       ;GET OFF-SET FROM RTS PC INSTRUCTION
```

```
    172 011630  005403                    NEG     R3              ;NEED NEGATIVE OF VALUE
    173 011632  006203                    ASR     R3              ;GET WORD OFFSET
    174 011634  042703  177400            BIC     #177400,R3      ;LEAVE ONLY DISPLACEMENT FIELD
    175 011640  050361  177776            BIS     R3,-2(R1)       ;STORE DISPLACEMENT INTO BNE INSTRUCTION
    176 011644  012603                    MOV     (SP)+,R3
    177 011646  000207                    RETURN
```

```
                                .SBTTL  SUBROUTINES
                        ;
                        ;-------------------------------------------------------------------------
                        ;  ACRFLD is called to accrue a sort field descriptor of the
                        ;  form tas.l where
                        ;    t = type code (returned as FT$xxx in FLDTYP)
                        ;    a = Ascending/Descending (FLDORD: 0==>ascending, 1==>descending)
                        ;    s = starting column number (value-1 returned in FLDPOS)
                        ;    l = field length (returned in FLDSIZ)
                        ;  When called, R0 must contain the 1st letter of the field descriptor
                        ;  and R1 must point to the 2nd letter.
                        ;  On return, R1 points to the delimiter hit.
                        ;
011650  010546          ACRFLD: MOV     R5,-(SP)
011652  012705  003542'         MOV     #KEYTC,R5       ;POINT TO DATA TYPE LETTER TABLE
011656  105715          2$:     TSTB    (R5)            ;REACHED END OF TABLE?
011660  001004                  BNE     5$              ;BR IF NOT
011662                          XXX     3               ;INVALIE DATA TYPE LETTER
011672  120025          5$:     CMPB    R0,(R5)+        ;LOOK FOR CHAR IN TABLE
011674  001370                  BNE     2$              ;LOOP TILL FOUND
011676  116500  000011          MOVB    KEYCOD-KEYTC-1(R5),R0;GET FT$xxx TYPE CODE
011702  010067  003504'         MOV     R0,FLDTYP       ;SAVE FIELD TYPE CODE
                        ;  Get second character which specifies Ascending/Descending
011706  005067  003506'         CLR     FLDORD          ;ASSUME ASCENDING
011712  112100                  MOVB    (R1)+,R0
011714  120027  000101          CMPB    R0,#'A          ;ASCENDING?
011720  001421                  BEQ     7$              ;BR IF YES
011722  120027  000104          CMPB    R0,#'D          ;DESCENDING?
011726  001003                  BNE     4$              ;BR IF NOT
011730  005267  003506'         INC     FLDORD          ;SET DESCENDING ORDER FLAG
011734  000413                  BR      7$
                        ;  If A/D was left out default to A.
011736  120027  000060  4$:     CMPB    R0,#'0          ;HIT A DIGIT?
011742  103403                  BLO     8$              ;BR IF NOT
011744  120027  000071          CMPB    R0,#'9
011750  101404                  BLOS    6$              ;HIT START OF COLUMN # FIELD
011752                  8$:     XXX     6               ;NOT A OR D
011762  005301          6$:     DEC     R1              ;POINT TO START OF NUMBER
                        ;  Accrue field starting column number.
011764  004767  001122  7$:     CALL    ACRDEC          ;ACCRUE STARTING COLUMN NUMBER
011770  005305                  DEC     R5              ;CONVERT TO BYTE NUMBER
011772  002004                  BGE     12$             ;BR IF VALUE VALID
011774                          XXX     5               ;INVALID STARTING COLUMN NUMBER
012004  010567  003500'  12$:    MOV     R5,FLDPOS       ;SAVE STARTING BYTE NUMBER
                        ;  Accrue field length if specified
012010  122127  000056          CMPB    (R1)+,#'.       ;PERIOD SHOULD BE THE DELIMITER
012014  001005                  BNE     10$             ;BR IF SIZE NOT SPECIFIED
012016  004767  001070          CALL    ACRDEC          ;ACCRUE SIZE VALUE
012022  010567  003502'         MOV     R5,FLDSIZ       ;SAVE SIZE VALUE
012026  000403                  BR      AFCKV
012030  005067  003502'  10$:    CLR     FLDSIZ          ;SAY SIZE WAS ZERO
012034  005301                  DEC     R1              ;POINT TO START OF NEXT SWITCH
                        ;
                        ;  Check validity of field spec.
                        ;
012036  016700  003504'  AFCKV:  MOV     FLDTYP,R0       ;GET FIELD DATA TYPE
012042  020027  000006          CMP     R0,#FT$T        ;DATE?
```

```
                                        BNE     1$              ;BR IF NOT
                                        MOV     #6,FLDSIZ       ;SAY SIZE IS 6 CHARS.
                                        BR      10$
                                1$:     CMP     R0,#FT$B        ;B--SIGNED BINARY?
                                        BEQ     2$              ;BR IF YES
                                        CMP     R0,#FT$U        ;U--UNSIGNED BINARY?
                                        BNE     3$              ;BR IF NOT
                                2$:     MOV     FLDSIZ,R0       ;GET SPECIFIED SIZE
                                        BNE     4$              ;BR IF IT WAS SPECIFIED
                                        MOV     #2,FLDSIZ       ;DEFAULT TO 2 BYTES
                                        BR      4$
                                3$:     CMP     R0,#FT$F        ;F--FLOATING?
                                        BNE     10$             ;BR IF NOT
                                        MOV     FLDSIZ,R0       ;GET SPECIFIED SIZE
                                        BNE     4$              ;BR IF SIZE WAS SPECIFIED
                                        MOV     #4,FLDSIZ       ;DEFAULT TO 4 BYTESS
                                ;  Common code for computational data types.
                                4$:     BIT     #1,R0           ;IS SIZE EVEN?
                                        BEQ     5$              ;BR IF YES
                                9$:     XXX     4               ;INVALID FIELD LENGTH
                                5$:     CMP     R0,#8.          ;NOT TOO LONG?
                                        BHI     9$              ;BR IF TOO LONG
                                        BIT     #1,FLDPOS       ;IS BYTE POSITION EVEN?
                                        BEQ     10$             ;BR IF YES
                                        XXX     5               ;INVALID STARTING COLUMN
                                ;  End of accruing field.
                                10$:    MOV     (SP)+,R5
                                        RETURN
```

```
                                ;-------------------------------------------------------------------
                                ;  SLLIT is called to accrue a literal value which is part of a record
                                ;  selection switch.
                                ;  When called, R2 must point to the current select control block,
                                ;  R0 must contain the 1st character of the literal and R1 must point
                                ;  to the 2nd character of the literal.
                                ;
012204  010346                  SLLIT:  MOV     R3,-(SP)
012206  010446                          MOV     R4,-(SP)
012210  010546                          MOV     R5,-(SP)
                                ;  Reserve room for literal below current select control block.
012212  016703  000000G                 MOV     TOPMEM,R3       ;GET FREE MEMORY POINTER
012216  016204  000010                  MOV     S$SIZ1(R2),R4   ;GET SIZE OF FIELD
012222  160403                          SUB     R4,R3           ;RESERVE ROOM FOR LITERAL
012224  042703  000001                  BIC     #1,R3           ;FORCE ADDRESS EVEN
012230  010362  000012                  MOV     R3,S$POS2(R2)   ;SAVE ADDRESS OF START OF LITERAL AREA
012234  010367  000000G                 MOV     R3,TOPMEM       ;UPDATE FREE MEMORY POINTER
012240  004767  000626                  CALL    QUOTCK          ;IS THIS A QUOTED STRING?
012244  001022                          BNE     SLLNUM          ;BR IF NOT (MUST BE NUMERIC LITERAL)
                                ;
                                ;  Accrue alphanumeric literal string.
                                ;
012246  112100                  SLLALP: MOVB    (R1)+,R0        ;GET NEXT CHAR FROM STRING
012250  001410                          BEQ     3$              ;BR IF HIT END OF COMMAND STRING
012252  120067  003526'                  CMPB    R0,QUOTE        ;IS THIS CHAR THE CLOSING QUOTE?
012256  001405                          BEQ     3$              ;BR IF YES
012260  005704                          TST     R4              ;HAVE WE MOVED ENOUGH CHARS?
012262  001771                          BEQ     SLLALP          ;BR IF YES
012264  110023                          MOVB    R0,(R3)+        ;MOVE CHAR TO HOLDING AREA
012266  005304                          DEC     R4              ;COUNT CHARS MOVES
012270  000766                          BR      SLLALP          ;GO MOVE MORE
                                ;  Reached end of string.
                                ;  See if we need to blank fill the end.
012272  005704                  3$:     TST     R4              ;DID WE COMPLETELY FILL FIELD?
012274  001404                          BEQ     4$              ;BR IF YES
012276  112723  000040                  MOVB    #' ,(R3)+       ;BLANK FILL REST OF FIELD
012302  005304                          DEC     R4
012304  000772                          BR      3$
012306  000167  000550          4$:     JMP     SLLEND
                                ;
                                ;  Accrue a numeric literal.
                                ;
                                ;  First move characters in literal to temp holding buffer.
012312  105067  003527'          SLLNUM: CLRB    NEGVAL          ;ASSUME VALUE IS NOT TO BE NEGATED
012316  105067  003530'                  CLRB    DECPT           ;ASSUME VALUE HAS NO DEC PT
012322  012703  001770'                  MOV     #SLBUF,R3       ;POINT TO TEMP HOLDING BUFFER
012326  120027  000060          6$:     CMPB    R0,#'0          ;DETERMINE WHAT TYPE OF CHAR THIS IS
012332  103017                          BHIS    1$              ;BR IF POSSIBLY DIGIT
012334  120027  000055                  CMPB    R0,#'-          ;MINUS SIGN?
012340  001003                          BNE     2$              ;BR IF NOT
012342  105267  003527'                  INCB    NEGVAL          ;REMEMBER TO NEGATE VALUE
012346  000415                          BR      3$
012350  120027  000053          2$:     CMPB    R0,#'+          ;PLUS SIGN?
012354  001412                          BEQ     3$              ;IGNORE IF YES
012356  120027  000056                  CMPB    R0,#'.          ;DEC POINT?
012362  001011                          BNE     4$              ;BR IF NOT
012364  105267  003530'                  INCB    DECPT           ;REMEMBER WE HIT DEC PT
```

```
     58 012370  000403                          BR      5$
     59 012372  120027  000071          1$:     CMPB    R0,#'9          ;IS CHAR A DIGIT?
     60 012376  101003                          BHI     4$              ;BR IF NOT (HIT END OF LITERAL)
     61 012400  110023                  5$:     MOVB    R0,(R3)+        ;MOVE CHAR TO HOLDING BUFFER
     62 012402  112100                  3$:     MOVB    (R1)+,R0        ;GET NEXT CHAR FROM LITERAL
     63 012404  000750                          BR      6$
     64                                 ;  Number is now stored in SLBUF.  Put null at end.
     65 012406  105013                  4$:     CLRB    (R3)            ;PUT NULL AT END OF STRING
     66 012410  005301                          DEC     R1              ;POINT BACK TO DELIMITER
     67 012412  105767  003530'                 TSTB    DECPT           ;DID NUMBER CONTAIN A DECIMAL POINT?
     68 012416  001404                          BEQ     SLCKNV          ;BR IF NOT
     69 012420                                  XXX     30              ;WE DON'T ALLOW DECIMAL VALUES
     70                                 ;
     71                                 ;  Branch off to appropriate processing routines with R3 pointing
     72                                 ;  to null at end of string and R4 containing a count of the number
     73                                 ;  of digits in the string.
     74                                 ;
     75 012430  010146                  SLCKNV: MOV     R1,-(SP)        ;MAKE R1 AVAILABLE TO NUMERIC LIT ROUTINES
     76 012432  010304                          MOV     R3,R4           ;GET POINTER PAST END OF STRING
     77 012434  162704  001770'                 SUB     #SLBUF,R4       ;GET COUNT OF # CHARS IN STRING
     78 012440  116200  000003                  MOVB    S$TYP(R2),R0    ;GET USAGE TYPE CODE
     79 012444  020027  000002                  CMP     R0,#FT$B        ;COMP?
     80 012450  001520                          BEQ     SLCOMP          ;BR IF YES
     81 012452  020027  000004                  CMP     R0,#FT$U        ;UNSIGNED BINARY?
     82 012456  001515                          BEQ     SLCOMP          ;BR IF YES
     83                                 ;
     84                                 ;  Usage is not comp.
     85                                 ;  Add enough leading zeroes to make number fit full field size.
     86                                 ;
     87 012460  016705  000000G                 MOV     TOPMEM,R5       ;FINAL VALUE IS TO BE STORED HERE
     88 012464  016201  000010                  MOV     S$SIZ1(R2),R1   ;GET # BYTES WANTED IN FINAL VALUE
     89 012470  122762  000020  000003          CMPB    #FT$J,S$TYP(R2) ;IS THIS A TRAILING SEPARATE SIGN FIELD?
     90 012476  001001                          BNE     4$              ;BR IF NOT
     91 012500  005301                          DEC     R1              ;LAST CHAR POS IS USED FOR SIGN
     92 012502  160401                  4$:     SUB     R4,R1           ;GET # EXTRA ZEROES WE NEED TO ADD
     93 012504  001411                          BEQ     1$              ;BR IF EXACT FIT
     94 012506  003004                          BGT     2$              ;BR IF NEED LEADING ZEROES
     95 012510                                  XXX     30              ;VALUE GIVEN IS LARGER THAN FIELD
     96 012520  112725  000060          2$:     MOVB    #'0,(R5)+       ;INSERT LEADING ZEROES
     97 012524  005301                          DEC     R1
     98 012526  003374                          BGT     2$
     99 012530  012701  001770'          1$:     MOV     #SLBUF,R1       ;POINT TO HOLDING BUFFER WITH VALUE
    100 012534  112100                  3$:     MOVB    (R1)+,R0        ;GET NEXT DIGIT FROM STRING
    101 012536  001402                          BEQ     5$              ;BR IF END HIT
    102 012540  110025                          MOVB    R0,(R5)+        ;MOVE IT TO FINAL RESTING PLACE
    103 012542  000774                          BR      3$
    104                                ;
    105                                ;  Branch off to different routines according to the sign type.
    106                                ;  At this point R5 points past the last digit in the string.
    107                                ;
    108 012544  116200  000003          5$:     MOVB    S$TYP(R2),R0    ;GET SIGN TYPE CODE
    109 012550  020027  000012                  CMP     R0,#FT$D        ;TRAILING NON-SEPARATE SIGN?
    110 012554  001423                          BEQ     SLTNS           ;BR IF YES
    111 012556  020027  000016                  CMP     R0,#FT$I        ;LEADING NON-SEPARATE SIGN?
    112 012562  001430                          BEQ     SLISS           ;BR IF YES
    113 012564  020027  000020                  CMP     R0,#FT$J        ;TRAILING NON-SEPARATE SIGN?
    114 012570  001437                          BEQ     SLTSS           ;BR IF YES
```

```
    115 012672  020027  000000                  CMP     R0,#FT$C        ;CHARACTER (UNSIGNED NUMERIC DISPLAY)?
    116 012676  001412                          BEQ     SLTNS           ;BR IF YES
    117 012600  020027  000014                  CMP     R0,#FT$A        ;UPPER CASE CHARACTER?
    118 012604  001407                          BEQ     SLTNG           ;BR IF YES
    119 012606  020027  000006                  CMP     R0,#FT$T        ;DATE?
    120 012612  001404                          BEQ     SLTNG           ;BR IF YES
    121 012614                                  XXX     30              ;INVALID USAGE TYPE
    122                                 ;
    123                                 ;  Trailing non-separate sign character.
    124                                 ;
    125 012624  105767  003527'         SLTNS:  TSTB    NEGVAL          ;IS VALUE TO BE NEGATIVE OR POSITIVE?
    126 012630  001513                          BEQ     SLNEND          ;BR IF POSITIVE
    127 012632  114500                          MOVB    -(R5),R0        ;GET LAST DIGIT OF VALUE
    128 012634  062700  000100                  ADD     #64.,R0         ;MARK AS NEGATIVE
    129 012640  110015                          MOVB    R0,(R5)         ;PUT BACK INTO NUMBER
    130 012642  000506                          BR      SLNEND
    131                                 ;
    132                                 ;  Leading separate sign.
    133                                 ;
    134 012644  012700  000053          SLLSS:  MOV     #'+,R0          ;ASSUME POSITIVE VALUE
    135 012650  105767  003527'                 TSTB    NEGVAL          ;NEGATIVE?
    136 012654  001402                          BEQ     1$              ;BR IF NOT
    137 012656  012700  000055                  MOV     #'-,R0          ;GET NEGATIVE SIGN CHAR
    138 012662  110077  000000G         1$:     MOVB    R0,@TOPMEM      ;STORE AS 1ST CHAR OF NUMBER
    139 012666  000474                          BR      SLNEND
    140                                 ;
    141                                 ;  Trailing separate sign.
    142                                 ;
    143 012670  012700  000053          SLTSS:  MOV     #'+,R0          ;ASSUME POSITIVE VALUE
    144 012674  105767  003527'                 TSTB    NEGVAL          ;NEGATIVE?
    145 012700  001402                          BEQ     1$              ;BR IF NOT
    146 012702  012700  000055                  MOV     #'-,R0          ;GET NEGATIVE SIGN CHAR
    147 012706  110015                  1$:     MOVB    R0,(R5)         ;STORE AS LAST DIGIT OF NUMBER
    148 012710  000463                          BR      SLNEND
    149                                 ;
    150                                 ;  Convert value to comp.
    151                                 ;
    152 012712  016201  000010          SLCOMP: MOV     S$SIZ1(R2),R1   ;GET # BYTES WANTED IN RESULT
    153 012716  006201                          ASR     R1              ;CONVERT TO # WORDS
    154 012720  016703  000000G                 MOV     TOPMEM,R3       ;POINT TO AREA FOR RESULT
    155 012724  005023                  1$:     CLR     (R3)+           ;CLEAR RESULT AREA
    156 012726  005301                          DEC     R1
    157 012730  003375                          BGT     1$
    158 012732  012704  001770'                 MOV     #SLBUF,R4       ;POINT TO BUFFER WITH DIGIT STRING
    159                                 ;  Get a digit and add it to accumulated value after multiplying by 10.
    160 012736  112400                  3$:     MOVB    (R4)+,R0        ;GET NEXT DIGIT FROM NUMBER STRING
    161 012740  001432                          BEQ     4$              ;BR IF END HIT
    162 012742  162700  000060                  SUB     #'0,R0          ;CONVERT FROM ASCII TO BINARY
    163 012746  016201  000010                  MOV     S$SIZ1(R2),R1   ;GET # BYTES IN RESULT FIELD
    164 012752  006201                          ASR     R1              ;CONVERT TO # WORDS
    165 012754  016703  000000G                 MOV     TOPMEM,R3       ;POINT TO RESULT FIELD
    166 012760  012705  003442'                 MOV     #CMPTMP,R5      ;POINT TO TEMP HOLDING BUFFER
    167 012764  005015                  2$:     CLR     (R5)            ;CLEAR THE CARRY TEMP
    168 012766  006313                          ASL     (R3)            ;2* OLD VALUE
    169 012770  006115                          ROL     (R5)            ;ROTATE CARRY INTO TEMP AREA
    170 012772  012515                          MOV     (R5)+,(R5)      ;COPY THE HIGH-ORDER WORD
    171 012774  061300                          ADD     (R3),R0         ;2*VALUE + NEW DIGIT
```

```
013076  005515                  ADC     (R5)            ;PROPOGATE CARRY
013100  006313                  ASL     (R3)            ;4*VALUE
013102  006145                  ROL     -(R5)
013104  006313                  ASL     (R3)            ;8*VALUE
013106  006115                  ROL     (R5)
013110  060023                  ADD     R0,(R3)+        ;10.*VALUE + DIGIT
013112  005525                  ADC     (R5)+
013114  061545                  ADD     (R5),-(R5)      ;ADD HIGH ORDER PARTS
013116  011500                  MOV     (R5),R0         ;COPY HIGH-ORDER FOR NEXT DIGIT
013120  005301                  DEC     R1              ;DONE?
013122  003360                  BGT     2$              ;BR IF MORE WORDS TO DO
                        ;  Go check next digit from number string.
013124  000744                  BR      3$
                        ;  See if we need to negate the comp value we just accrued.
013126  105767  003527'    4$: TSTB    NEGVAL          ;NEGATIVE?
013132  001412                  BEQ     SLNEND          ;BR IF NOT
013134  016703  000000G         MOV     TOPMEM,R3       ;POINT TO VALUE
013140  016205  000010          MOV     S$SIZ1(R2),R5   ;GET # BYTES IN VALUE
013144  006205                  ASR     R5              ;GET # WORDS
013146  000401                  BR      5$
013150  005513                6$: ADC     (R3)            ;PROPOGATE CARRY
013152  005423                5$: NEG     (R3)+           ;NEGATE A WORD OF THE VALUE
013154  005305                  DEC     R5              ;MORE TO DO?
013156  003374                  BGT     6$              ;BR IF YES
                        ;
013160  012601                SLNEND: MOV     (SP)+,R1
                        ;
                        ;  finished accrueing literal value.
                        ;
013162  012605                SLLEND: MOV     (SP)+,R5
013164  012604                  MOV     (SP)+,R4
013166  012603                  MOV     (SP)+,R3
013170  000207                  RETURN
                        ;
                        ;-------------------------------------------------------------------------
                        ;  QUOTCK is called to see if the character in R0 is a quote or
                        ;  apostrophe character.
                        ;  On return, the condition codes are set for BEQ/BNE.
                        ;  The character in R0 is stored in a call called QUOTE.
                        ;
013172  110067  003526' QUOTCK: MOVB    R0,QUOTE        ;STORE THE CHARACTER
013176  120027  000042          CMPB    R0,#'"          ;IS IT QUOTE?
013202  001402                  BEQ     1$              ;BR IF YES
013204  120027  000047          CMPB    R0,#''          ;APOSTROPHE?
013210  000207                1$: RETURN                  ;RETURN WITH CONDITION CODE SET
```

```
                                ;-------------------------------------------------------------
                                ;  ACRDEC IS CALLED TO ACCRUE A DECIMAL VALUE.
                                ;  WHEN CALLED R1 MUST POINT TO THE 1ST DIGIT OF THE NUMBER.
                                ;  ON RETURN THE LOW-ORDER 16 BITS OF THE NUMBER ARE RETURNED
                                ;  IN R5.  THE HIGH-ORDER 16 BITS ARE STORED IN VALHI.
                                ;  ALL OTHER REGISTERS ARE PRESERVED.
                                ;
013112  010046                  ACRDEC: MOV     R0,-(SP)
013114  010446                          MOV     R4,-(SP)
013116  005004                          CLR     R4
013120  005005                          CLR     R5
013122  112100                  1$:     MOVB    (R1)+,R0        ;PICK UP NEXT CHAR
013124  162700  000060                  SUB     #'0,R0          ;SEE IF DIGIT & CVT TO BINARY
013130  002421                          BLT     9$              ;BR IF NOT DIGIT
013132  020027  000011                  CMP     R0,#9.          ;IS THIS A DIGIT?
013136  101016                          BHI     9$              ;BR IF NOT
                                ;  MULTIPLY PREVIOUS VALUE BY 10.
013140  006305                          ASL     R5              ;LOW ORDER *2
013142  006104                          ROL     R4              ;HIGH ORDER *2
013144  010446                          MOV     R4,-(SP)
013146  010546                          MOV     R5,-(SP)
013150  006305                          ASL     R5              ;*4
013152  006104                          ROL     R4
013154  006305                          ASL     R5              ;*8.
013156  006104                          ROL     R4
013160  062605                          ADD     (SP)+,R5
013162  005504                          ADC     R4
013164  062604                          ADD     (SP)+,R4
                                ;  ADD IN VALUE OF NEW DIGIT
013166  060005                          ADD     R0,R5
013170  005504                          ADC     R4
013172  000753                          BR      1$              ;GO GET NEXT CHAR
                                ;  HIT END OF NUMBER
013174  010467  003454'          9$:     MOV     R4,VALHI        ;SAVE HIGH ORDER PART
013200  005301                          DEC     R1              ;POINT BACK TO DELIMITER
013202  012604                          MOV     (SP)+,R4
013204  012600                          MOV     (SP)+,R0
013206  000207                          RETURN
```

```
                                ;-------------------------------------------------------------
                                ;  ACROCT IS CALLED TO ACCRUE AN OCTAL VALUE.
                                ;  WHEN CALLED R1 MUST POINT TO START OF NUMBER.
                                ;  ON RETURN RESULT IS IN R5.
                                ;  ALL OTHER REGISTERS ARE PRESERVED.
                                ;
013210  010046                  ACROCT: MOV     R0,-(SP)
013212  005005                          CLR     R5
013214  112100                  2$:     MOVB    (R1)+,R0        ;GET NEXT CHAR
013216  162700  000060                  SUB     #'0,R0          ;CVT TO BINARY
013222  002410                          BLT     9$              ;BR IF HIT END OF NUMBER
013224  020027  000007                  CMP     R0,#7           ;MAKE SURE ITS OCTAL
013230  101005                          BHI     9$              ;BR IF HIT END OF NUMBER
013232  006305                  1$:     ASL     R5              ;MULTIPLY PREVIOUS VALUE BY 8
013234  006305                          ASL     R5
013236  006305                          ASL     R5
013240  060005                          ADD     R0,R5           ;ADD IN NEW DIGIT
013242  000764                          BR      2$              ;GO GET NEXT CHAR
                                ;  HIT END OF NUMBER
013244  005301                  9$:     DEC     R1              ;POINT BACK TO DELIMITER
013246  012600                          MOV     (SP)+,R0
013250  000207                          RETURN
```

```
                                ;-------------------------------------------------------------
                                ;  GTCPRM IS CALLED TO ACCRUE A SWITCH PARAMETER OF
                                ;  THE FORM ":CHAR:COL".
                                ;  ON RETURN, THE CHARACTER CODE IS IN R2 AND THE
                                ;  COLUMN NUMBER IS IN R5.  R1 POINTS PAST THE PARAM.
                                ;  ALL OTHER REGS ARE PRESERVED.
                                ;
013252  004767  000130        GTCPRM: CALL    CKCOL           ;CHECK FOR COLON
013256  121127  000047                CMPB    (R1),#''        ;QUOTED CHAR?
013262  001004                       BNE     1$              ;BR IF NOT
013264  005201                       INC     R1              ;SKIP OVER QUOTE
013266  112102                       MOVB    (R1)+,R2        ;GET THE CHAR
013270  005201                       INC     R1              ;SKIP TRAILING QUOTE
013272  000403                       BR      2$
013274  004767  177710        1$:     CALL    ACROCT          ;ACCRUE OCTAL CHARACTER CODE
013300  010502                       MOV     R5,R2
013302  004767  000100        2$:     CALL    CKCOL           ;CHECK FOR COLON
013306  004767  177600                CALL    ACRDEC          ;ACCRUE DEC COLUMN #
013312  000207                       RETURN
                                .IF DF RT11
                                ;
                                ;-------------------------------------------------------------
                                ;  LOADIT IS CALLED TO LOAD A DEVICE HANDLER INTO THE
                                ;  TOP OF MEMORY.  WHEN CALLED R1 MUST POINT TO A CELL
                                ;  CONTAINING THE RAD50 DEVICE NAME AND R5 MUST POINT TO
                                ;  THE TOP OF FREE MEMORY.
                                ;  ON RETURN R5 POINTS TO THE NEW TOP OF MEMROY.
                                ;
013314  005711        LOADIT: TST     (R1)            ;IS THERE A DEVICE NAME SPECIFIED?
013316  001432                BEQ     9$
013320  010046                MOV     R0,-(SP)
                        ;  CHECK STATUS OF DEVICE
                                .SERR
                                .DSTAT  #CBLK,R1
013340  103004                BCC     1$              ;BR IF DEVICE RECOGNIZED
013342                  4$:     XXX     7               ;DON'T RECOGNIZE DEVICE
013352  005767  003436'    1$:     TST     CBLK+4          ;IS HANDLER RESIDENT NOW?
013356  001006                BNE     2$              ;BR IF YES
013360  166705  003434'        SUB     <CBLK+2>,R5     ;RESERVE ROOM FOR HANDLER
013364                          .FETCH  R5,R1           ;LOAD HANDLER
013372  103763                BCS     4$              ;BR IF CAN'T GET IT
013374                  2$:     .HERR
013402  012600                MOV     (SP)+,R0
013404  000207          9$:     RETURN
                                .ENDC
                                ;
                                ;-------------------------------------------------------------
                                ;  CKCOL IS CALLED TO CHECK THAT THE NEXT CHARACTER IS
                                ;  A COLON.  IF NOT AN ERROR ABORT OCCURS.
                                ;  IF YES THE COLON IS SKIPPED OVER.
                                ;
013406  122127  000072        CKCOL:  CMPB    (R1)+,#':       ;IS NEXT CHAR A COLON?
013412  001404                BEQ     1$              ;BR IF YES
013414                          XXX     19              ;SYNTAX ERROR IF NOT
013424  000207          1$:     RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Subroutine to determine the correct default file extension based on
                                ;  the file type.
                                ;  When called, R2 must contain the file type flags (RT$xxx).
                                ;  On return, R2 contains the Rad-50 default file extension.
                                ;
013426  032702  000000G       GETDFX: BIT     #RT$CBL,R2      ;COBOL FILE?
013432  001417                        BEQ     1$              ;BR IF NOT
                                ;  COBOL file.
013434  032702  000000G               BIT     #RT$OX,R2       ;ORGANIZATION = INDEXED?
013440  001403                        BEQ     2$              ;BR IF NOT
013442  016702  000060'               MOV     R50CIO,R2       ;EXT = CIO
013446  000207                        RETURN
013450  032702  000000G       2$:     BIT     #RT$RB,R2       ;MODE = BINARY?
013454  001403                        BEQ     3$              ;BR IF NOT
013456  016702  000056'               MOV     R50CDB,R2       ;EXT = CDB
013462  000207                        RETURN
013464  016702  000054'       3$:     MOV     R50CDA,R2       ;EXT = CDA
013470  000207                        RETURN
013472  032702  000000G       1$:     BIT     #RT$DBL,R2      ;DIBOL FILE?
013476  001403                        BEQ     4$              ;BR IF NOT
                                ;  DIBOL file.
013500  016702  000050'               MOV     R50DDF,R2       ;EXT = DDF
013504  000207                        RETURN
013506  016702  000052'       4$:     MOV     R50DAT,R2       ;SET EXTENTION TO "DAT"
013512  000207                        RETURN
```

```
                                        .SBTTL  PRTNUM -- Print a decimal number
                                ;-------------------------------------------------------------------
                                ;  PRTNUM is called to convert a 32-bit binary value to ascii and print
                                ;  the resulting string on the terminal without trailing Cr-Lf.
                                ;
                                ;  Inputs:
                                ;   R4 = low-order 16-bits of value
                                ;   R5 = High-order 16-bits of value
                                ;
013514  010146                  PRTNUM: MOV     R1,-(SP)
                                ;
                                ;  Convert the value to an asciz string.
                                ;
013516  004767  000016                  CALL    CVTNUM          ;CONVERT VALUE TO ASCIZ STRING
                                ;
                                ;  Print the result.
                                ;
013522  112100                  1$:     MOVB    (R1)+,R0        ;GET NEXT DIGIT FROM STRING
013524  001403                          BEQ     2$              ;BR IF ASCIZ NULL HIT
013526                                  .TTYOUT                 ;PRINT THE DIGIT
013532  000773                          BR      1$
                                ;
                                ;  Finished
                                ;
013534  012601                  2$:     MOV     (SP)+,R1
013536  000207                          RETURN
                                        .SBTTL  CVTNUM -- Convert binary value to asciz string
                                ;-------------------------------------------------------------------
                                ;  CVTNUM is called to convert a 32-bit binary value to an ascii digit string.
                                ;
                                ;  Inputs:
                                ;   R4 = Low-order 16-bits of value
                                ;   R5 = High-order 16-bits of value
                                ;
                                ;  Outputs:
                                ;   R1 = Address of start of asciz string.
                                ;
013540  010446                  CVTNUM: MOV     R4,-(SP)
013542  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get pointer to end of asciz result buffer.
                                ;
013544  012701  011702'                 MOV     #CVTBUF+9.,R1   ;POINT TO NULL BYTE AT END OF CONVERT BUFFER
                                ;
                                ;  Split off low order digit.
                                ;
013550  012700  000012          1$:     MOV     #10.,R0         ;SET DIVIDE BY 10.
013554  004767  0000006                 CALL    DIVIDE          ;DIVIDE R4-R5 BY 10.
013560  062700  000060                  ADD     #'0,R0          ;CONVERT REMAINDER TO ASCII DIGIT
013564  110041                          MOVB    R0,-(R1)        ;MOVE TO RESULT BUFFER
013566  005704                          TST     R4              ;MORE TO CONVERT?
013570  001367                          BNE     1$              ;BR IF YES
013572  005705                          TST     R5              ;CHECK HIGH-ORDER PART OF QUOTIENT
013574  001365                          BNE     1$
                                ;
                                ;  Finished
                                ;
```

CVTNUM -- Convert binary value to asciz string

```
     58 013576  012605                          MOV     (SP)+,R5
     59 013600  012604                          MOV     (SP)+,R4
     60 013602  000207                          RETURN
```

```
                        .SBTTL  EXECUTION TERMINATION
                ;-------------------------------------------------------------
                ;  TERMINATION ROUTINE.
                ;  ENTER WITH ERROR CODE IN R0.
                ;
013604 010067 000136'   TERM:   MOV     R0,ABTBUF
013610 012706 000000G           MOV     #STACK,SP       ;RESET SP
                ;  MAKE SURE ALL CHANNELS ARE CLOSED
013614                          $KEEP   #FD1
013662                          $PURGE  #FD2
014012                          $KEEP   #FD3
014060 005767 000000G           TST     T2SIZ           ;DID WE USE TEMP FILE 2?
014064 001454                   BEQ     7$              ;BR IF NOT
014066                          $PURGE  #FD4
014216                  7$:
                        .IF DF RT11
                ;  RELEASE DEVICE HANDLERS
014216 105767 000000G   12$:    TSTB    SPCFLG          ;Special chain mode?
014222 001123                   BNE     13$             ; => yes
014224                          .SERR
014232 012701 000000G           MOV     #FILES,R1       ;R1 -> TABLE OF -> FILE SPECS
014236 005711           11$:    TST     (R1)            ;(R1) == 0 IF AT END OF TABLE
014240 001407                   BEQ     8$              ;EXIT IF END
014242 005731                   TST     @(R1)+          ;CHECK DEVICE NAME
014244 001774                   BEQ     11$             ;IGNORE BLANK DEVICE
014246 005741                   TST     -(R1)           ;REVERSE AUTO-INCREMENT FROM ABOVE
014250                          .RELEAS (R1)+           ;RELEASE HANDLER
014256 000767                   BR      11$
014260                  8$:
                ;       .SETTOP #SIEND          ;RESET TOP OF MEMORY
                ;
                ;  IF WE WERE CALLED IN MESSAGE MODE SEND BACK A RESPONSE
                ;
014260 105767 000000G           TSTB    MODE            ;MESSAGE MODE?
014264 002022                   BGE     1$              ;BR IF NOT
014266 016767 000000G 000142'   MOV     SRTCNT,MRC1     ;SET SORTED RECORD COUNT
014274 016767 000002G 000140'   MOV     SRTCNT+2,MRC2
014302 012700 000126'           MOV     #SNDBLK,R0      ;SEND REPLY
014306 104375                   EMT     375
014310 103006                   BCC     10$             ;BR IF NO ERROR
014312 105067 000000G           CLRB    MODE            ;GET OUT OF MESSAGE MODE
014316                          XXX     26
014326 000167 163636    10$:    JMP     SORTR           ;WAIT FOR ANOTHER MESSAGE
                ;
                ;  SEE IF ANY ERROR OCCURED.
014332 016701 000136'   1$:     MOV     ABTBUF,R1       ;GET ERROR CODE
014336 001034                   BNE     2$              ;BR IF ERROR OCCURED
014340 152737 000001 000053     BISB    #NOERR,@#USERRB ;SET SUCCESS FLAG
                ;  ONLY PRINT "SORT COMPLETED" IF IN DIRECT RUN MODE.
014346 105767 000000G           TSTB    MODE
014352 001040                   BNE     3$              ;BR IF IN CHAIN MODE
                ;  Print number of records read and sorted.
014354                          .PRINT  #TXNRR          ;NUMBER OF RECORDS READ:
014362 016704 000000G           MOV     RCNT1,R4        ;GET LOW-ORDER VALUE
014366 016705 000000G           MOV     RCNT2,R5        ;GET HIGH-ORDER VALUE
014372 004767 177116            CALL    PRINUM          ;PRINT THE VALUE
014376                          .PRINT  #TXNRS          ;NUMBER OF RECORDS SORTED:
```

```
     58 014404  016704  000000G                 MOV     SRTCNT,R4       ;GET LOW-ORDER VALUE
     59 014410  016705  000002G                 MOV     SRTCNT+2,R5     ;GET HIGH-ORDER VALUE
     60 014414  004767  177074                  CALL    PRTNUM          ;PRINT VALUE
     61 014420                                  .PRINT  #CRLF           ;TERMINATE THE LINE
     62 014426  000412                          BR      3$
     63                                 ;  PRINT SORT ERROR MESSAGE
     64 014430  006301                  2$:     ASL     R1              ;CVT CODE TO WORD TABLE INDEX
     65 014432                                  .PRINT  #SRTFHD         ;PRINT ERROR MESSAGE HEADER
     66 014440                                  .PRINT  ERRTAB(R1)      ;PRINT ERROR MESSAGE
     67 014446  152737  000010  000053          BISB    #SEVERR,@#USERRB;SET SEVERE-ERROR FLAG
     68 014454  005767  000000G         3$:     TST     CINSPC          ;CHAIN WANTED?
     69 014460  001002                          BNE     4$              ;BR IF CHAIN
     70 014462  000167  163502          5$:     JMP     SORTR           ;GO GET NEXT SORT COMMAND LINE
     71                                 ;  SEE IF WE SHOULD CHAIN TO NEXT PROGRAM
     72 014466  005701                  4$:     TST     R1              ;ERROR DURING SORT?
     73 014470  001374                          BNE     5$              ;BR IF YES
     74 014472  012702  000500          13$:    MOV     #500,R2         ;SET UP CHAIN AREA
     75 014476  012701  000000G                 MOV     #CINSPC,R1
     76 014502  012122                  6$:     MOV     (R1)+,(R2)+
     77 014504  020227  000510                  CMP     R2,#510
     78 014510  103774                          BLO     6$
     79 014512                                  SETNRT  CINNRT,CHNPPN
     80 014536  016701  000136'                 MOV     ABTBUF,R1       ;Check if special error mode
     81 014542  001410                          BEQ     14$             ; => no
     82 014544  012722  041101                  MOV     #"AB,(R2)+      ;Yes, return the message "ABORT X", where
     83 014550  012722  051117                  MOV     #"OR,(R2)+      ;   X is a single byte binary value of the
     84 014554  012722  020124                  MOV     #"T ,(R2)+      ;   error code
     85 014560  110122                          MOVB    R1,(R2)+
     86 014562  105022                          CLRB    (R2)+
     87 014564                          14$:    .CHAIN                  ;CHAIN TO NEXT PROGRAM
     88
     89                                 .IFF    ;ELSE IF RSX-11
     90
     91                                         JMP     SORTR           ;GO GET NEXT COMMAND
     92
     93                                 .ENDC
     94                                 ;
     95          014574'                SIEND   =       .+2
     96          000001                         .END
```

Symbol table

```
ABTBUF  000136R      002 EM20    004730R      002 F$INPL= ****** G        KLEN  = ****** G        OWCNT = ****** G
ACRDEC  013112R      005 EM21    004766R      002 F$PAUS= ****** G        KSTRT = ****** G        PARSW   001306R      005
ACRFLD  011650R      005 EM22    005017R      002 F$VLEN= ****** G        KTYPE = ****** G        PFSPC = 000002
ACROCT  013210R      005 EM23    005065R      002 GENBRU  011614R      005 KYBIN = ****** G        PRTNUM  013514R      005
AFCKV   012036R      005 EM24    005141R      002 GENBUF  002432R      002 KYCOMP= ****** G        POBASE= ****** G
ANDFLG= ****** G         EM25    005206R      002 GENKPT  011554R      005 KYDATE= ****** G        POTOP = ****** G
ASTRTX  011622R      002 EM26    005255R      002 GENRPT  011446R      005 KYDIS = ****** G        P1SIZ = ****** G
BELL  = 000007           EM27    005311R      002 GENVEC  003610R      002 KYLSS = ****** G        P1TOP = ****** G
BIGEST= 177777           EM28    005367R      002 GENWIN  011430R      005 KYTSS = ****** G        P2SIZ = ****** G
BLKFC = ****** G         EM29    005441R      002 GETDFX  013426R      005 KYUCA = ****** G        P2TOP = ****** G
BT$HI = 000001 G         EM3     004035R      002 GETEND  001060R      005 LF    = 000012          QBLK  = ****** G
BT$LO = 000002 G         EM30    005502R      002 GETSW   001516R      005 LOADIT  013314R      005 QLEN  = 000003
BT$NE = 000003           EM31    005544R      002 GETTYP  002756R      005 LOWMEM= ****** G        QUOTCK  013072R      005
BT$REV= 000004 G         EM32    005616R      002 GNASCI  011104R      005 MAPEND= ****** G        QUOTE   003526R      002
BUFSIZ= 001000           EM33    005672R      002 GNBIN   011344R      005 MODE  = ****** G        RCNT1 = ****** G
CBLK    003432R      002 EM34    005753R      002 GNCALL  011374R      005 MRC1    000142R      002 RCNT2 = ****** G
CHADEX  000104R      002 EM35    006012R      002 GNCMP   011306R      005 MRC2    000140R      002 RECLEN= ****** G
CHNBIT= 000400           EM36    006051R      002 GNDATE  011336R      005 MSGBLK  000116R      002 REC1  = ****** G
CHNO  = 000000           EM37    006077R      002 GNDIS   011360R      005 MSGBUF  000644R      002 REC2  = ****** G
CHNPPN= ****** G         EM38    006165R      002 GNLSS   011322R      005 MSGNAM= ****** G        RETLOC  003510R      002
CINFIL  001604R      002 EM4     004060R      002 GNTSS   011330R      005 MULTPY= ****** G        RMON  = 000054
CINNRT= ****** G         EM5     004105R      002 GNUCA   011366R      005 MWAIT   000374R      005 RSPCS = ****** G
CINSPC= ****** G         EM6     004143R      002 GNUSI   011252R      005 MXREC1= ****** G        RSTAT   003512R      002
CISRTI= ****** G         EM7     004174R      002 GOTLIN  000352R      005 MXREC2= ****** G        RSTEXT= ****** G
CKCOL   013406R      005 EM8     004234R      002 GTCPRM  013252R      005 M$POS1= 000000 G        RSTS  = ****** G
CKFTYP  004600R      005 EM9     004263R      002 GTERR   003244R      005 M$POS2= 000002 G        RSTSCH= ****** G
CMPCOD  000014R      002 ENDSW   004514R      005 GTLN    000114R      002 M$SIZE= 000004 G        RSTSJB= 000404
CMPOPS  000000R      002 EOFBUF= ****** G         GTOK    003240R      005 M$$SZ = 000006 G        RT$CBL= ****** G
CMPTMP  003442R      002 EOFCHR= ****** G         GTORG   003074R      005 NEGVAL  003527R      002 RT$CR = ****** G
CODGEN  006526R      005 EOFCOL= ****** G         GTRMD   003176R      005 NOERR = 000001           RT$DBL= ****** G
CODSIZ  003566R      002 ERR   = 000006           HB$ARS= 000006           NUMBUF= ****** G        RT$FTN= ****** G
COMPAR= ****** G         ERRLOC= 000052           HB$DFS= 000002           NUMCO = 000014          RT$F11= ****** G
CR    = 000015           ERRTAB  003632R      002 HB$DIP= 000030           NUMCOL= ****** G        RT$OR = ****** G
CRLF    011670R      002 FD1   = ****** G         HB$DUP= 000001           NUMKY2= ****** G        RT$OS = ****** G
CSPCOF= 000376           FD2   = ****** G         HB$RIU= 000032           NUMLEN= ****** G        RT$OX = ****** G
CTRLZ = 000032           FD3   = ****** G         HB$RSZ= 000004           NXSJ    001716R      005 RT$RA = ****** G
CURSAV= ****** G         FD4   = ****** G         HD$DEV= 000000           NXSJ1   002074R      005 RT$RB = ****** G
CURSEL  003476R      002 FILBUF  001344R      002 HD$$SZ= 000004           NXTSW   004466R      005 RT$TXT= ****** G
CVTBUF  011671R      002 FILES = ****** G         HLPMSG  006257R      002 OBUFSZ= ****** G        RT$VLN= ****** G
CVTNUM  013540R      005 FILPPN= ****** G         INDDEX  000074R      002 ONEASC  011160R      005 RT$X  = ****** G
DBLKFC= 000005           FLDMAP= ****** G         INDNRT  003516R      002 OPNDAT  007550R      005 RT11  = 000000
DECPT   003530R      002 FLDORD  003506R      002 INDRCM  000572R      005 OPNTMP  005532R      005 RWDLEN= ****** G
DEFEXT  000064R      002 FLDPOS  003500R      002 INFLEN= ****** G         OPRSTK  003540R      002 R50CDA  000054R      002
DIVIDE= ****** G         FLDSIZ  003502R      002 INIFIL  007142R      005 OPRSTP  003537R      002 R50CDB  000056R      002
EM0     003750R      002 FLDTYP  003504R      002 INIX    010022R      005 OPTRNS  000022R      002 R50CDO  000062R      002
EM1     003767R      002 FSTIME= ****** G         INRLEN= ****** G         OP$AND= 000006           R50CIO  000060R      002
EM10    004311R      002 FT$A  = 000014           INXDEV  003514R      002 OP$BOE= 000002           R50DAT  000052R      002
EM11    004351R      002 FT$B  = 000002           ISMARS= ****** G         OP$CND= 000011           R50DDF  000050R      002
EM12    004401R      002 FT$C  = 000000           ITYPE = ****** G         OP$COR= 000010           R50DK   000034R      002
EM13    004434R      002 FT$D  = 000012 G         JSW   = 000044           OP$EOE= 000001           R50TMP  000046R      002
EM14    004470R      002 FT$F  = 000010           KAD   = ****** G         OP$LPN= 000004           R50TN0  000036R      002
EM15    004527R      002 FT$I  = 000016           KEYBR   000000R      004 OP$NOT= 000007           R50TN1  000040R      002
EM16    004563R      002 FT$J  = 000020           KEYCOD  003554R      002 OP$OR = 000005           R50TN2  000042R      002
EM17  = 004401R      002 FT$T  = 000006 G         KEYEND  000042R      003 OP$RPN= 000003           R50XX   000044R      002
EM18    004621R      002 FT$U  = 000004           KEYTAB  000000R      003 ORFLG = ****** G         SAVBUF= ****** G
EM19    004676R      002 F$BIN = ****** G         KEYTC   003542R      002 OTYPE = ****** G         SCNCMD  000450R      005
EM2     004011R      002 F$CR  = ****** G         KEYWD   003452R      002 OVLBIT= 001000           SCNCM1  001060R      005
```

Symbol table

```
SELADR= ****** G        SLPSOP  004142R     005 SWBUF   002052R     002 TMPCNT= ****** G        XXX21 = 002664R     005
SEOFR   006700R     005 SLTNS   012624R     005 SWCHAN  002202R     005 TOPMEM= ****** G        XXX22 = 004532R     005
SETBLK  006754R     005 SLTSS   012670R     005 SWCR    004454R     005 TRYT2   006052R     005 XXX24 = 007014R     005
SETCHN  002264R     005 SNDBLK  000126R     002 SWEOF   002100R     005 TSXFL = ****** G        XXX25 = 000630R     005
SETFIL  004770R     005 SORTC   000360R     005 SWINCL  003250R     005 TTBUF   000144R     002 XXX26 = 014316R     005
SETNRT= ****** G        SORTD   000226R     005 SWITYP  002674R     005 TWOASC  011206R     005 XXX27 = 003524R     005
SEVERR= 000010          SORTR   000170R     005 SWKEY   002364R     005 TXNRR   011625R     002 XXX28 = 003620R     005
SFLAGS= ****** G        SO$AND= 000002 G        SWMAP   004270R     005 TXNRS   011644R     002 XXX29 = 004052R     005
SF$LIT= 000001 G        SO$CMP= 000000 G        SWNMBR  002464R     005 T1SIZ = ****** G        XXX3  = 011662R     005
SIEND = 014574R     005 SO$NOT= 000006 G        SWOTYP  002722R     005 T2SIZ = ****** G        XXX30 = 012614R     005
SKPCNT= ****** G        SO$OR = 000004 G        SWPAUS  002452R     005 UEOF  = ****** G        XXX31 = 004312R     005
SKPVAL= ****** G        SPCFLG= ****** G        SWPPN   002534R     005 USERRB= 000053          XXX32 = 004360R     005
SLBUF   001770R     002 SPC1  = ****** G        SWREC   001744R     005 USRLOC= 000046          XXX34 = 002702R     005
SLCKNV  012430R     005 SPC2  = ****** G        SWSIZE  001620R     005 VALHI   003454R     002 XXX35 = 002746R     005
SLCMOP  003474R     005 SPC3  = ****** G        SWSKIP  001722R     005 VERSNO= 000276          XXX36 = 002510R     005
SLCOMP  012712R     005 SPC4  = ****** G        SWSND   002006R     005 V2GLIN  000300R     005 XXX37 = 007330R     005
SLCSPC  003274R     005 SPC5  = ****** G        S$BR  = 000004 G        V3FLAG= ****** G        XXX38 = 010042R     005
SLLALP  012246R     005 SRTCNT= ****** G        S$FLG = 000005 G        V3GLIN  000244R     005 XXX4  = 012142R     005
SLLEND  013062R     005 SRTFHD  011607R     002 S$LINK= 000000 G        WRDS  = 000004          XXX5  = 012170R     005
SLLIT   012204R     005 SRTINI  000000RG    005 S$OPR = 000002 G        XAREA   003456R     002 XXX6  = 011752R     005
SLLNUM  012312R     005 SRTNAM  011601R     002 S$POS1= 000006 G        XBUF    001624R     002 XXX7  = 013342R     005
SLLSS   012644R     005 STACK = ****** G        S$POS2= 000012 G        XEMT  = ****** G        XXX8  = 010200R     005
SLNEND  013060R     005 START1= ****** G        S$SIZ1= 000010 G        XXX14 = 010510R     005 XXX9  = 006034R     005
SLOPND  003544R     005 SVRSTS  011000R     005 S$TYP = 000003 G        XXX19 = 013414R     005 . . . V1 = 000003
SLOPR   003630R     005 SWBIN   002066R     005 S$$SZ = 000014 G        XXX2  = 001634R     005 . . . V2 = 000061
SLOPX   004020R     005 SWBLOK  001770R     005 TERM    013604RG    005 XXX20 = 001600R     005

. ABS.  000000      000     (RW,I,GBL,ABS,OVR)
        000000      001     (RW,I,LCL,REL,CON)
SORTID  011704      002     (RW,D,LCL,REL,CON)
SWNAM   000042      003     (RW,I,GBL,REL,OVR)
SWBRV   000042      004     (RW,I,GBL,REL,OVR)
SORTI   014572      005     (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  16168 Words  ( 64 Pages)
Size of core pool:  18432 Words  ( 72 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:57.56
,DB4:CBSRTI,DK:CBSRTH.CR==DK:CBSRTH,CBSRTI/C
```

Cross reference table (CREF V05.03)

```
...V1   9-7    9-7    9-37   9-50   10-37  11-33  11-35  11-42  11-44  11-44  11-44  11-46  11-49  11-49
       11-52  11-52  11-53  12-12  16-26  22-11  22-11  22-11  22-12  22-12  22-12  22-17  22-17  22-80
       22-80  22-80  22-84  22-84  22-84  22-96  22-96  22-96  22-98  22-98  22-98  22-102 22-102 22-102
       22-102 22-102 25-32  25-124 25-124 25-124 25-129 25-129 26-27  26-27  26-27  26-30  26-30  26-32
       26-32  26-35  26-35  33-34  33-40  35-20  36-9   36-9   36-10  36-10  36-10  36-10  36-10  36-11
       36-11  36-14  36-14  36-14  36-14  36-14  36-27  36-53  36-57  36-61  36-65  36-66
...V2  11-44  11-44  11-44# 11-44# 11-49  11-49  11-49# 11-49# 11-52  11-52# 22-11  22-11  22-11# 22-11#
       22-12  22-12  22-12# 22-12# 22-17  22-17  22-17# 22-17# 22-80  22-80  22-80# 22-80# 22-84  22-84
       22-84# 22-84# 22-96  22-96  22-96# 22-96# 22-98  22-98  22-98# 22-98# 22-102 22-102 22-102 22-102#
       22-102# 22-102# 25-124 25-124 25-124# 25-124# 25-129 25-129 25-129# 25-129# 26-27  26-27  26-27# 26-27#
       26-30  26-30  26-30# 26-30# 26-32  26-32# 26-35  26-35# 36-9   36-9#  36-10  36-10  36-10  36-10#
       36-10# 36-10# 36-11  36-11# 36-14  36-14  36-14  36-14# 36-14# 36-14#
ABTBUF  4-32   4-35#  36-6*  36-46  36-80
ACRDEC 15-5   15-27  15-36  15-45  15-103 17-31  17-37  17-59  17-66  20-5   20-14  20-21  20-27  29-40
       29-48  31-8#  33-18
ACRFLD 17-10  19-36  19-83  29-14#
ACRDCT 15-98  32-7#  33-15
AFCKV  27-50  27-56#
ANDFLG  2-59  19-7*  19-29*  19-111*  19-132*
ASTRTX  8-54#  9-37
BELL    1-40#
BIGEST  1-36#
BLKFC   2-55  13-20*  15-46*  24-6   24-30*
BT$HI   2-38   3-24#  3-60   3-61   3-63   3-64   3-67
BT$LO   2-38   3-25#  3-60   3-61   3-62   3-65   3-67
BT$NE   3-67# 17-126
BT$REV  2-39   3-26#  3-60   3-63   3-65
BUFSIZ  1-34#  24-16  24-21  24-25  24-32
CBLK    4-49#  33-34  33-37  33-39
CHADEX  4-20#  16-26
CHNBIT  1-53#   9-26
CHNO    1-87#  22-11  22-11  22-12  22-12  22-17  22-17  22-80  22-80  22-84  22-84  22-96  22-96  22-98
       22-98  22-102 22-102 22-102 22-102 25-124 25-124 25-129 25-129 26-27  26-27  26-30  26-30  26-32
       26-32  26-35  26-35  36-9   36-9   36-10  36-10  36-10  36-10  36-11  36-11  36-14  36-14  36-14
       36-14
CHNPPN  2-66  13-30*  17-47  36-79
CINFIL  4-44#  16-8   16-26
CINNRT  2-65  16-39  36-79
CINSPC  2-46  13-28*  16-31  16-33  36-68  36-75
CISRTI  2-53  12-8
CKCOL  15-4   15-26  15-35  15-44  15-53  15-89  16-7   17-7   17-30  17-54  18-5   19-4   20-4   33-8
       32-17  33-52#
CKFTYP 21-21#
CMPCOD  3-60#  19-67
CMPOPS  3-47#   3-53  19-60
CMPTMP  4-50#  30-166
CODGEN 23-7#
CODSIZ  5-54#  23-10
COMPAR  2-60   23-21*
CR      1-38#   2-71#  8-5    8-6    8-7    8-8    8-9    8-10   8-11   8-12   8-13   8-14   8-15   8-16
        8-17   8-18   8-19   8-20   8-21   8-22   8-23   8-24   8-25   8-26   8-27   8-28   8-29   8-30
        8-31   8-32   8-33   8-34   8-35   8-36   8-37   8-38   8-39   8-40   8-41   8-42   8-43   8-44
        8-45   8-46   8-47   8-48   8-49   11-64
CRLF    8-57#  36-61
CSPCOF  1-65#  16-38  27-47
CTRLZ   1-41#   2-73#  11-60  15-86
```

Cross reference table (CREF V05.03)

```
CURSAV    2-52    22-19*
CURSEL    4-54#   19-46*   19-185
CVTBUF    8-58#   35-43
CVTNUM   35-14    35-38#
DBLKFC    1-35#
DECPT     4-68#   30-45*   30-57*   30-67
DEFEXT    4-15#   25-24*   25-27*   25-32
DIVIDE    2-54    22-48    22-66    22-68    35-48
EM0       6-12     7-2#
EM1       6-13     7-3#
EM10      6-22     7-12#
EM11      6-23     7-13#
EM12      6-24     7-14#    7-19
EM13      6-25     7-15#
EM14      6-26     7-16#
EM15      6-27     7-17#
EM16      6-28     7-18#
EM17      6-29     7-19#
EM18      6-30     7-20#
EM19      6-31     7-21#
EM2       6-14     7-4#
EM20      6-32     7-22#
EM21      6-33     7-23#
EM22      6-34     7-24#
EM23      6-35     7-25#
EM24      6-36     7-26#
EM25      6-37     7-27#
EM26      6-38     7-28#
EM27      6-39     7-29#
EM28      6-40     7-30#
EM29      6-41     7-31#
EM3       6-15     7-5#
EM30      6-42     7-32#
EM31      6-43     7-33#
EM32      6-44     7-34#
EM33      6-45     7-35#
EM34      6-46     7-36#
EM35      6-47     7-37#
EM36      6-48     7-38#
EM37      6-49     7-39#
EM38      6-50     7-40#
EM4       6-16     7-6#
EM5       6-17     7-7#
EM6       6-18     7-8#
EM7       6-19     7-9#
EM8       6-20     7-10#
EM9       6-21     7-11#
ENDSW    20-51    21-4#
EOFBUF    2-59    13-16*   23-52*
EOFCHR    2-46    13-21*   15-86*   15-99*   23-57
EOFCOL    2-46    13-19*   15-85*   15-104*  23-47
ERR       1-90#   22-17*   22-80*   22-84*   22-96*   22-98*   25-124*  26-27*   26-30*   26-30*   26-30*   26-32    26-32*   26-32*
         26-32*
ERRLOC    1-49#   22-17    22-80    22-84    22-96    22-98    25-124   26-27    26-30    26-32
ERRTAB    6-12#   36-66
F$BIN     2-64    15-79    28-57    28-64    28-85
```

Cross reference table (CREF V05.03)

```
F$CR      2-64    20-43
F$INPL    2-53    25-59
F$PAUS    2-64    17-23
F$VLEN    2-64    15-20   21-32
FD1       2-67     9-7*   22-11   22-12   22-17   25-124  25-129  26-27   26-30   26-32   26-35   36-9
FD2       2-67     9-7*   22-80   22-84   36-10
FD3       2-67     9-7*   36-11
FD4       2-67     9-7*   22-96   22-98   22-102  36-14
FILBUF    4-43#   11-12   12-32   25-32
FILES     2-67    22-16   25-40   25-100  25-103  25-117  27-48   36-21
FILPPN    2-66    13-31*  17-51   22-11   22-11   22-11   22-12   22-12   22-12   22-80   22-80   22-80   22-84   22-84
         22-84    22-96   22-96   22-96   22-98   22-98   22-98   22-102  22-102  22-102  25-124  25-124  25-124  26-27
         26-27    26-27   36-10   36-10   36-10   36-14   36-14   36-14
FLDMAP    2-58    13-17*  20-12*
FLDORD    4-58#   17-11   29-24*  29-30*
FLDPOS    4-55#   17-12   19-51   19-84   29-44*  29-80
FLDSIZ    4-56#   17-13   19-52   29-49*  29-51*  29-59*  29-65   29-67*  29-71   29-73*
FLDTYP    4-57#   17-14   19-53   19-85   29-22*  29-56
FSTIME    2-43     9-18    9-25*
FT$A      3-36#    5-48   30-117
FT$B      3-31#    5-48   29-61   30-79
FT$C      3-30#    5-48   30-115
FT$D      2-54     3-35#   5-48   30-109
FT$F      3-34#    5-48   29-69
FT$I      3-37#    5-48   30-111
FT$J      3-38#    5-48   30-89   30-113
FT$T      2-37     3-33#   5-48   21-12   29-57   30-119
FT$U      3-32#    5-48   29-63   30-81
GENBRU   28-16    28-29   28-40   28-45   28-56   28-111  28-168#
GENBUF    4-48#   11-49   11-56   11-71   26-30   26-41
GENKPT   28-12    28-106  28-152#
GENRPT   28-24    28-35   28-51   28-127#  28-153
GENVEC    5-66#   23-31
GENWIN   23-20    23-40   28-118#
GETDFX   25-23    25-26   34-7#
GETEND   11-26    11-73#
GETSW    14-3#    20-55
GETTYP   17-54    17-93   18-5#
GNASCI    5-66    28-6#
GNBIN     5-70    28-84#
GNCALL   28-65    28-70   28-75   28-80   28-86   28-91   28-96   28-106#
GNCMP     5-67    28-63#
GNDATE    5-69    28-79#
GNDIS     5-71    28-90#
GNLSS     5-73    28-69#
GNTSS     5-74    28-74#
GNUCA     5-72    28-95#
GNUSI     5-68    28-51#
GOTLIN    7-44     7-63#
GTCPRM   33-2#
GTERR    18-32    18-53   18-55   18-74   18-84#
GTLN      4-23#    7-7
GTOK     18-60    18-72   18-79#
GTORG    18-14    18-18   18-22   18-26   18-30   18-37#
GTRMD    18-45    18-49   18-57   18-64#
HB$ARS    1-74#
```

Cross reference table (CREF V05.03)

```
HB$DFS    1-72#   26-64
HB$DIP    1-75#   26-65
HB$DUP    1-71#   26-56
HB$RIU    1-76#   26-42   26-43
HB$RSZ    1-73#   26-54
HD$$SZ    1-81#   26-83
HD$DEV    1-80#   26-71
HLPMSG    3-4#    10-37
INDDEX    4-17#   11-35
INDNRT    4-62#
INDRCM   11-25    11-30#
INFLEN    2-55
INIFIL   22-6     25-15#
INIX     25-113   26-13#
INRLEN    2-58    13-6*   15-9*   21-4    22-25   22-67   26-60*
INXDEV    4-61#   26-69*  26-73
ISMARS    2-47    26-55*
ITYPE     2-49    13-25*  17-87*  21-6    21-21   21-59*  22-34   25-22   25-56   25-111
JSW       1-51#    9-26   24-42
KAD       2-44    17-11*  28-127
KEYBR     5-8#    14-28
KEYCOD    5-48#   29-21
KEYEND    5-42#   14-21
KEYTAB    5-6#    14-18   14-27
KEYTC     5-47#   29-15   29-21
KEYWD     4-51#   14-3    14-10   14-17
KLEN      2-44    17-13*  21-14*  28-6    28-108
KSTRT     2-44    17-12*  28-26   28-37   28-53   28-154
KTYPE     2-44    17-14*  21-12   23-9    23-30
KYBIN     2-63    28-84
KYCOMP    2-58    28-63
KYDATE    2-63    28-79
KYDIS     2-59    28-90
KYLSS     2-63    28-69
KYTSS     2-63    28-74
KYUCA     2-55    28-95
LF        1-39#    2-72#   8-5     8-5     8-6     8-6     8-7     8-7     8-8     8-9     8-10    8-11    8-12    8-13
          8-14     8-15    8-16    8-17    8-18    8-19    8-20    8-21    8-22    8-23    8-24    8-25    8-26    8-27
          8-28     8-29    8-30    8-31    8-32    8-33    8-34    8-35    8-36    8-37    8-38    8-39    8-40    8-41
          8-42     8-43    8-44    8-45    8-46    8-47    8-48    8-49    8-50    9-52    9-57   11-66
LOADIT   25-102   26-76   33-29#
LOWMEM    2-61    24-41*  24-43*
M$$SZ     2-38     3-104#  20-11   20-31
M$POS1    2-37     3-100#  20-17*
M$POS2    2-38     3-101#  20-24*
M$SIZE    2-36     3-102#  20-28*
MAPEND    2-58    20-35*
MODE      2-43     9-20    9-31*   10-4*   10-10*  10-14*  36-34   36-41*  36-50
MRC1      4-27#   36-36*
MRC2      4-36#   36-37*
MSGBLK    4-24#   10-11
MSGBUF    4-26     4-42#   10-18*  10-19
MSGNAM    2-60     4-31    10-20
MULTPY    2-54    22-44   22-46   22-62   22-64
MWAIT     9-21    10-10#  10-34
MXREC1    2-45    13-33*  15-37*  22-41   26-47   26-49*
```

Cross reference table (CREF V05.03)

```
MXREC2    2-46    13-34*  15-38*  22-42   26-44   26-50*
NEGVAL    4-67#   30-44*  30-51*  30-125  30-135  30-144  30-186
NOERR     1-56#   36-48
NUMBUF    2-48    13-37   13-39
NUMCO     3-53#   19-59
NUMCOL    2-48    17-33*
NUMKY2    2-45    13-5*   17-5    17-6*   21-9    23-12   23-33
NUMLEN    2-48    13-23*  17-38*
NXSJ     15-11    15-13   15-21#  15-30   15-39   15-47
NXSJ1    15-73    15-80#
NXTSW    15-21    15-80   15-105  16-22   17-17   17-24   17-39   17-52   17-88   17-98   19-151  20-37   20-44   20-50#
OBUFSZ    2-56    24-35*
ONEASC   28-9     28-24#
OP$AND    3-78#   19-110
OP$BOE    3-74#    4-71   19-149
OP$CND    3-81#   19-131  19-177
OP$COR    3-80#   19-128  19-175
OP$EOE    3-73#   19-97
OP$LPN    3-76#   19-17   19-143
OP$NOT    3-79#   19-28
OP$OR     3-77#   19-118
OP$RPN    3-75#   19-101  19-145  19-152
OPNDAT   25-112   25-117#
OPNTMP   22-52    22-54   22-72   22-79#
OPRSTK    4-73#   19-5*   19-16*  19-17*  19-138  19-141  19-142*  19-168*  19-169*
OPRSTP    4-71#    4-73   19-5
OPTRNS    3-85#   19-160
ORFLG     2-59    19-8*   19-30*  19-119* 19-129*
OTYPE     2-49    13-26*  17-97*  21-24   21-60*  25-25
OVLBIT    1-52#   24-42
OWCNT     2-56    24-37*
POBASE    2-60
POTOP     2-61
P1SIZ     2-61
P1TOP     2-62    24-10   24-12   24-42
P2SIZ     2-61
P2TOP     2-57    24-9    24-41   24-42   24-42   24-42
PARSW    13-5#
PFSPC     1-88#   22-17*  22-80*  22-84*  22-96*  22-98*  22-102  22-102*  25-124*  25-129  25-129*  26-27*  26-35*  36-9*
         36-10    36-10*  36-11*  36-14   36-14*
PRTNUM   35-10#   36-56   36-60
QBLK      2-43     7-7
QLEN      1-48#    7-7
QUOTCK   12-21    12-37   15-56   15-92   16-11   30-18   30-212#
QUOTE     4-66#   12-25   12-42   15-60   30-25   30-212*
R50CDO    4-13#   26-70
R50CDA    4-10#   34-18
R50CDB    4-11#   34-16
R50CIO    4-12#   34-12
R50DAT    4-9#    34-25
R50DDF    4-8#    34-23
R50DK     4-2#    25-66
R50TMP    4-7#    25-94   25-95
R50TNO    4-3#    25-71   25-73
R50TN1    4-4#    25-72
R50TN2    4-5#    25-74
```

Cross reference table (CREF V05.03)

```
R50XX     4-54#   25-89
RCNT1     2-57    13-14*  36-54
RCNT2     2-57    13-15*  36-55
REC1      2-60    23-25   28-134  28-146
REC2      2-60    23-27   28-136  28-144
RECLEN    2-45    13-7*   20-6*   22-23   22-25*  22-26   22-43   22-63   23-50   23-58
RETLOC    4-59#   23-37   28-119* 28-171
RMON      1-57#    9-7    27-46
RSPCS     2-52    25-32   25-41   25-42   25-43   25-44   25-45   25-46   25-47   25-48   25-49   26-68   26-78
RSTAT     4-60#   23-22*  28-112* 28-130  28-132* 28-140  28-142* 28-160*
RSTEXT    5-7      9-7     9-7
RSTS      2-66     9-7*    9-7*   11-40   12-17   16-35   25-82   27-38
RSTSCH    5-7      9-7     9-7
RSTSJB    1-64#   25-84
RT$CBL    2-49    18-13   18-54   34-7
RT$CR     2-54    21-51   21-58
RT$DBL    2-49    18-17   21-23   34-20
RT$F11    2-50    18-33
RT$FTN    2-50    18-21   21-47   21-54
RT$OR     2-51    18-48   21-23   21-41   21-49   21-56
RT$OS     2-51    18-44   18-59   21-30
RT$OX     2-51    17-95   18-56   21-6    21-27   21-29   21-41   22-34   25-56   25-111  34-10
RT$RA     2-53    18-59   18-71
RT$RB     2-53    18-75   21-23   21-38   21-41   21-45   21-52   34-14
RT$TXT    2-50    18-29
RT$VLN    2-54    21-34   21-35   21-40   21-43
RT$X      2-50    18-25   21-41   21-45   21-52
RT11      1-47#    9-13   13-27   15-48   16-1    17-18   17-40   20-38   25-1    33-20   36-16
RWDLEN    2-57    22-29*
S$$SZ     2-36     3-13#  19-38   19-155  19-181
S$BR      2-35     3-7#   19-67*  19-126  19-190  19-190*
S$FLG     2-35     3-8#   19-48*  19-79*  19-191*
S$LINK    2-35     3-4#   19-42*  19-47*  19-156* 19-158* 19-182* 19-184*
S$OPR     2-35     3-5#   19-49*  19-160* 19-189  19-189*
S$POS1    2-36     3-9#   19-51*  19-186  19-186*
S$POS2    2-36     3-11#  19-84*  30-16*
S$SIZ1    2-36     3-10#  19-52*  19-187  19-187* 30-13   30-88   30-152  30-163  30-189
S$TYP     2-35     3-6#   19-53*  19-85   19-188  19-188* 30-78   30-89   30-108
SAVBUF    2-65    25-118
SCNCM1   12-4#
SCNCMD    9-64    10-6    10-29#  11-72
SELADR    2-58    13-18*  19-44*
SEOFR    23-47#
SETBLK   23-48    24-6#
SETCHN   11-18    16-19   16-25#
SETFIL   22-5#
SETNRT    2-66    22-11   22-12   22-80   22-84   22-96   22-98   22-102  25-124  26-27   36-10   36-14   36-79
SEVERR    1-55#   36-67
SF$LIT    2-37     3-42#  19-79
SFLAGS    2-64    13-24*  15-20*  15-79*  17-23*  20-43*  21-32   25-59*  28-57*  28-64*  28-85*
SIEND     9-7     11-33   11-42   12-14   36-95#
SKPCNT    2-45    13-8*   15-28*
SKPVAL    2-52    13-13*  15-29*
SLBUF     4-46#   30-46   30-77   30-99   30-158
SLCKNV   30-68    30-75#
SLCMOP   17-55#
```

Cross reference table (CREF V05.03)

```
SLCOMP  30-80    30-82    30-152#
SLCSPC  19-12#   19-18    19-179
SLLALP  30-23#   30-28    30-31
SLLEND  30-39    30-201#
SLLIT   19-78    30-8#
SLLNUM  30-19    30-44#
SLLSS   30-112   30-134#
SLNEND  30-126   30-130   30-139   30-148   30-187   30-197#
SLOPND  19-71#   19-192
SLOPR   19-60    19-86    19-92#   19-148
SLOPX   19-31    19-98    19-102   19-112   19-120   19-130   19-138#  19-163
SLPSOP  19-139   19-168#
SLTNS   30-110   30-116   30-118   30-120   30-125#
SLTSS   30-114   30-143#
SNDBLK   4-30#   36-38
SO$AND   2-37     3-18#    3-91     3-94
SO$CMP   2-36     3-17#   19-49
SO$NOT   2-34     3-20#    3-92
SO$OR    2-37     3-19#    3-90     3-93
SORTC    9-27    10-4#
SORTD    9-22     9-31#
SORTR    9-17#   10-38    36-43    36-70
SPC1     2-67    25-54
SPC2     2-67    22-11    22-80    22-84    25-63
SPC3     2-67    22-12    22-96    22-98    25-67
SPC4     2-67    26-27
SPC5     2-67    26-21
SPCFLG   2-66    11-19*   36-18
SRTCNT   2-62    13-9*    13-10*   36-36    36-37    36-58    36-59
SRTFHD   8-53#   36-65
SRTINI   2-34     9-6#
SRTNAM   4-25     8-52#
STACK    2-56    36-7
START1   2-56    24-44
SVRSTS  25-36    27-38#
SWBIN    5-34    15-79#
SWBLOK   5-29    15-44#
SWBUF    4-47#   12-34*   12-56    14-1
SWCHAN   5-26    16-7#
SWCR     5-32    20-43#
SWEOF    5-25    15-85#
SWINCL   5-30    19-4#
SWITYP   5-35    17-84#
SWKEY    5-27    17-5#    17-16
SWMAP    5-31    20-4#
SWNMBR   5-37    17-30#
SWOTYP   5-36    17-93#
SWPAUS   5-28    17-23#
SWPPN    5-38    17-46#
SWREC    5-24    15-35#
SWSIZE   5-22    15-4#
SWSKIP   5-23    15-26#
SWSND    5-33    15-53#
T1SIZ    2-47    22-87*
T2SIZ    2-47    22-79*   22-104*  36-12
TERM     2-34    10-15    10-35    11-37    14-23    15-8     15-16    16-21    17-36    17-78    17-86    17-99    19-32    19-64
```

Cross reference table (CREF V05.03)

```
        19-87   19-117  19-147  20-9    20-20   20-54   21-8    22-86   24-18   25-34   25-58   25-126  26-23   26-29
        26-34   29-18   29-37   29-43   29-77   29-82   30-69   30-95   30-121  33-36   33-54   36-6#   36-42
TMPCNT   2-62   13-11*  13-12*
TOPMEM   2-45   12-13*  19-38*  19-39   19-155* 19-156  19-157  19-181* 19-182  19-183  20-10   20-36*  23-8    23-18*
        23-49   23-54*  24-8    25-99   25-106* 26-67   26-86*  30-12   30-17*  30-87   30-138* 30-154  30-165  30-188
TRYT2   22-89#
TSXFL    2-43    9-7*    9-7*   25-87
TTBUF    4-41#   9-33    9-63
TWOASC  28-10   28-35#
TXNRR    8-55#  36-53
TXNRS    8-56#  36-57
UEOF     2-62   13-22*
USERRB   1-50#  36-48*  36-67*
USRLOC   1-54#   9-7*   11-32*  12-14*
V2GLIN   7-35    7-48#
V3FLAG   2-62    9-7*    9-7*    9-34
V3GLIN   7-37#   7-43
VALHI    4-52#  15-38   31-34*
VERSNO   1-58#   9-7
WRDS     1-89#  26-30*  26-30*  26-32
XAREA    4-53#  11-44   11-49
XBUF     4-45#   9-7    11-35   11-42   11-44   11-46   11-53   16-26   16-30
XEMT    22-11   22-11   22-11   22-12   22-12   22-12   22-17   22-80   22-80   22-80   22-84   22-84   22-84   22-96
        22-96   22-96   22-98   22-98   22-98   22-102  22-102  22-102  25-124  25-124  25-124  25-129  26-27   26-27
        26-27   26-30   26-30   26-30   36-10   36-10   36-10   36-14   36-14   36-14
XXX14   26-34#
XXX19   10-35#  15-16#  20-54#  25-34#  33-54#
XXX2    15-8#
XXX20   14-25#
XXX21   16-21#  17-78#
XXX22   21-8#
XXX24   24-18#
XXX25   11-37#
XXX26   10-15#  36-42#
XXX27   19-64#
XXX28   19-87#
XXX29   19-32#  19-117#  19-147#
XXX3    29-18#
XXX30   30-69#  30-95#   30-121#
XXX31   20-9#
XXX32   20-20#
XXX34   17-86#
XXX35   17-99#
XXX36   17-36#
XXX37   25-58#
XXX38   26-23#
XXX4    29-77#
XXX5    29-43#  29-82#
XXX6    29-37#
XXX7    33-36#
XXX8    25-126# 26-29#
XXX9    22-86#
```

Cross reference table (CREF V05.03)

```
$ERASE   1-638#  22-11   22-12
$EXIT    1-656#
$FDB     1-145#
$GETBO   1-402#  24-42
$GETCM   1-356#
$GETTO   1-424#  12-12
$KEEP    1-448#  26-35   36-9    36-11
$LOCK    1-328#  11-34   22-5
$NEW     1-228#  22-80   22-84   22-96   22-98
$OLD     1-260#  25-124  26-27
$PRINT   1-664#
$PRT     1-696#
$PURGE   1-465#  22-102  36-10   36-14
$READ    1-492#  26-30
$REOPE   1-309#  22-17
$SAVES   1-299#  25-129
$SPC     1-159#
$SYDON   1-674#
$SYINI   1-589#   9-7
$SYSTE   1-171#
$UNLOC   1-338#  11-54   22-105
$WAIT    1-558#  26-32
$WRITE   1-525#
...CM0    9-7     9-7     9-7     9-37    9-37    9-37    9-37    11-35   11-35   11-35   11-42   11-46   11-53   16-26
         16-26   16-26   25-32   25-32   25-32   33-34   33-40   36-27
...CM1   11-44   11-49   22-17   22-80   22-84   22-96   22-98   25-124  25-129  26-27   26-30
...CM2   11-44   11-44   11-49   11-49   11-49   11-49   22-11   22-11   22-12   22-12   22-17   22-80   22-80   22-80
         22-84   22-84   22-84   22-96   22-96   22-96   22-98   22-98   22-98   22-102  22-102  25-124  25-124  25-129
         26-27   26-27   26-30   26-30   26-30   26-30   36-10   36-10   36-14   36-14
...CM3   11-52   22-102  26-35   36-9    36-10   36-11   36-14
...CM5    9-7     9-50   10-37   11-33   11-42   11-44   11-46   11-49   11-53   12-12   22-11   22-12   22-17   22-80
         22-84   22-96   22-98   22-102  25-124  25-129  26-27   26-30   33-34   33-40   35-20   36-10   36-14   36-27
         36-53   36-57   36-61   36-65   36-66
...CM7   11-49   26-30
.CHAIN    1-96#   1-99#  36-87
.CLOSE    1-98#  11-52   22-102  26-35   36-9    36-10   36-11   36-14
.CSISP   1-100#   9-7    11-35   16-26   25-32
.DELET    1-97#  22-11   22-12   22-102  36-10   36-14
.DSTAT    1-98#  33-34
.ENTER    1-96#  22-80   22-84   22-96   22-98
.EXIT     1-99#   7-23
.FETCH    1-98#  11-42   33-40
.GTLIN   1-101#   9-37
.GVAL    1-100#
.HERR     1-97#   9-7    22-11   22-12   22-102  33-42   36-10   36-14
.LOCK     1-96#  11-34   22-5
.LOOKU    1-96#  11-44   25-124  26-27
.PRINT    1-97#  10-37   36-53   36-57   36-61   36-65   36-66
.PURGE    1-99#
.QSET     1-98#   9-7
.READ    1-101#  26-30
.READW   1-100#  11-49
.RELEA    1-99#  11-46   11-53   36-27
.REOPE   1-100#  22-17
.SAVES   1-100#  25-129
.SERR     1-97#   9-7    22-11   22-12   22-102  33-33   36-10   36-14   36-20
```

Cross reference table (CREF V05.03)

```
. SETTO   1-97#   11-33   12-12
. TTYIN   1-99#    9-51    9-56
. TTYOU   1-98#    9-50   35-20
. UNLOC   1-96#   11-54   22-105
. WAIT    1-101#  26-32
. WRITE   1-101#
. WRITW   1-101#
POP       1-118#  22-11   22-11   22-12   22-12   22-17   22-17   22-80   22-80   22-84   22-84   22-96   22-96   22-98
         22-98    22-102  25-124  25-124  25-129  25-129  26-27   26-27   26-30   26-30   26-32   26-35   36-9    36-10
         36-11    36-14
PUSH      1-112#  22-11   22-11   22-12   22-12   22-17   22-17   22-80   22-80   22-84   22-84   22-96   22-96   22-98
         22-98    22-102  25-124  25-124  25-129  25-129  26-27   26-27   26-30   26-30   26-32   26-35   36-9    36-10
         36-11    36-14
SETNRT    1-210#  22-11   22-12   22-80   22-84   22-96   22-98   22-102  25-124  26-27   36-10   36-14   36-79
SW        5-10#    5-22    5-23    5-24    5-25    5-26    5-27    5-28    5-29    5-30    5-31    5-32    5-33    5-34
          5-35     5-36    5-37    5-38
XXX       6-4#    10-15   10-35   11-37   14-23   15-8    15-16   16-21   17-36   17-78   17-86   17-99   19-32   19-64
         19-87    19-117  19-147  20-9    20-20   20-54   21-8    22-86   24-18   25-34   25-58   25-126  26-23   26-29
         26-34    29-18   29-37   29-43   29-77   29-82   30-69   30-95   30-121  33-36   33-54   36-42
```

SORT1 SORT -- PHASE 1 MACRO V05.01b Monday 06-May-85 16:43
Table of contents

```
                                        .TITLE  SORTI   SORT -- PHASE 1
000000                                  .CSECT  SORTI
                                        .ENABL  LC
                000000'             P1BASE  =       .
                                    ;
                                    ;   THIS SECTION OF SORT CREATES THE INITIAL SORTED RUNS.
                                    ;   IT USES A REPLACEMENT SELECTION ALGORITHM FOR THIS PROCESS.
                                    ;   SEE KNUTH 'ART OF COMPUTER PROGRAMMING' VOLUME 3 SECT. 5.4.1
                                    ;   FOR A DISCUSSION OF THIS TECHNIQUE.  THE SYMBOLS AND LABELS USED IN
                                    ;   THIS PROGRAM AGREE WITH THE CONVENTIONS USED BY KNUTH.
                                    ;
                                    ;   Copyright (c) 1980, 1981.
                                    ;   S&H Computer Systems, Inc.
                                    ;   Nashville, Tennessee USA
                                    ;
                                    ;   All rights reserved.
                                    ;   This software may not be sold, distributed or otherwise made
                                    ;   available for use except as licensed by S&H Computer Systems, Inc.
                                    ;
                                    ;   COBOL-Plus, SORT-Plus, RTSORT, TSX and TSX-Plus are registered
                                    ;   trademarks of S&H Computer Systems, Inc.
                                    ;
                                    ;
                                    ;
                                    ;   GLOBAL DEFINITIONS.
                                    ;
                                        .GLOBL  SORTP1,P1TOP,P1BASE,P1SIZ
                                    ;
                                    ;   GLOBAL REFERENCES.
                                    ;
                                        .GLOBL  KTYPE,FD1,SETNRT
                                        .GLOBL  TOPMEM,RECLEN,FT$D,SO$AND,SO$NOT
                                        .GLOBL  UEOF
                                        .GLOBL  FILRN,FILST,FILFL,NUMKY2
                                        .GLOBL  RWDLEN,MXFILS,MXFIL2,FILNB
                                        .GLOBL  SKPCNT,INFLEN,OWCNT,BLKFC
                                        .GLOBL  OBUF1,OBUF2,OBUFSZ,EOFBUF
                                        .GLOBL  OBFEND,MXREC1,MXREC2
                                        .GLOBL  KYDATE,KYLSS,KYTSS
                                        .GLOBL  KYBIN,ITYPE,SRTCNT,TMPCNT
                                        .GLOBL  RT$CBL,RT$DBL,RT$FTN,RT$TXT
                                        .GLOBL  RT$OS,RT$OR,RT$OX,RT$F11
                                        .GLOBL  RT$RA,RT$RB,RT$VLN,RT$CR
                                        .GLOBL  NUMBUF,NUMCOL,NUMLEN
                                        .GLOBL  RCNT1,RCNT2,EOFCOL,EOFCHR,ABORT
                                        .GLOBL  NXTBLK,ORPOS,FLUSH,COMPAR
                                        .GLOBL  P1TERM,LOWMEM,FLBLK,ISMARS
                                        .GLOBL  OUTWRT,IFPNT,CUROFL,KSTRT,FT$T
                                        .GLOBL  FLDMAP,INRLEN,MAPEND,SELADR
                                        .GLOBL  S$OPR,SO$CMP,SO$OR,S$POS1,S$POS2
                                        .GLOBL  S$FLG,SF$LIT,S$SIZ1,S$TYP,BT$LO
                                        .GLOBL  BT$HI,BT$REV,S$LINK,KYCOMP,KYDIS
                                        .GLOBL  KYUCA,M$POS1,M$POS2,M$SIZE,M$$SZ
                                        .GLOBL  S$BR,ORFLG,ANDFLG,CISRT1
                                        .GLOBL  CURSAV,SKPVAL
```

```
                                ;
                        000620  MAXREC  =       400.            ;MAX # OF RECORDS TO PLACE IN CORE
                                ;
                                ;  RECORD DESCRIPTOR TABLES.
000000                          LOC:    .BLKW   MAXREC          ;ADDRESS OF RECORD(I).
001440                          LOSER:  .BLKW   MAXREC          ;INDEX OF LOSER BELOW THIS NODE.
003100                          FE:     .BLKW   MAXREC          ;EXTERNAL BLOCK FORWARD LINK.
004540                          FI:     .BLKW   MAXREC          ;INTERNAL BLOCK FORWARD LINK.
006200                          RN:     .BLKW   MAXREC          ;RUN # THIS RECORD WILL GO IN.
                                ;  GENERAL DATA AREA.
                        000015  CR      =       15              ;ASCII CARRIAGE RETURN
                        000012  LF      =       12              ;ASCII LINE FEED
                        000032  CTRLZ   =       32              ;CTRL-Z EOF CHARACTER
007640  000000                  NUMREC: .WORD   0               ;# OF RECORDS WHICH WILL FIT IN CORE.
007642  000000                  IBUF1:  .WORD   0               ;ADDRESS OF PRIMARY INPUT BUFFER.
007644  000000                  IBUF2:  .WORD   0               ;ADDRESS OF SECONDARY INPUT BUFFER.
007646  000000                  RQ:     .WORD   0               ;RUN # RECORD(Q) WILL GO IN.
007650  000000                  RC:     .WORD   0               ;CURRENT RUN NUMBER.
007652  000000                  RMAX:   .WORD   0               ;# OF HIGHEST RUN IN FILE.
007654  000000                  Q:      .WORD   0               ;INDEX OF NEW RECORD.
007656  000000                  LSTKEY: .WORD   0               ;ADDRESS OF LAST WINNER RECORD.
007660  000000                  IBFEND: .WORD   0               ;ADDR OF END OF INPUT BUF.
007662  000000                  IRPOS:  .WORD   0               ;POINTER TO NEXT INPUT REC.
007664  000000                  INRDSZ: .WORD   0
007666  000000                  OFLINK: .WORD   0
007670  000000                  MAPBUF: .WORD   0
007672  000000                  RECPTR: .WORD   0
007674  000000                  SELREC: .WORD   0
007676  000000                  LSTACK: .WORD   0
                                ;  BYTE DATA ITEMS.
007700      000                 EOFFLG: .BYTE   0
                                ;  Logical operation temp stack.
007701                                  .BLKB   6.
007707                          LSTKTP:
                                ;
                                        .EVEN
                                ;
                                ;  MACRO TO CAUSE ABORT
                                ;
                                        .MACRO  XXX     CODE
                                XXX'CODE        =       .
                                        MOV     #CODE,R0
                                        CALL    ABORT
                                        .ENDM   XXX
```

```
                                        .SBTTL  ALLOCATE CORE SPACE
                                ;
                                ;  Check in for debugging breakpoint.
                                ;
007710  004767  000000G         SORTP1: CALL    CISRT1          ;CHECK IN TO ROOT
                                ;
                                ;  ALLOCATE AVAILABLE CORE SPACE.
                                ;
007714                                  $GETBOT R1,#P1TOP
                                ;  Allocate room for temp record buffer if field mapping is wanted.
007734  005767  000000G                 TST     FLDMAP          ;IS FIELD MAPPING WANTED?
007740  001407                          BEQ     6$              ;BR IF NOT
007742  010167  177722                  MOV     R1,MAPBUF       ;SAVE ADDRESS OF START OF BUFFER
007746  066701  000000G                 ADD     INRLEN,R1       ;ADD LENGTH OF INPUT LOGICAL RECORDS
007752  005201                          INC     R1              ;BOUND UP TO WORD BOUNDARY
007754  042701  000001                  BIC     #1,R1
                                ;  ALLOCATE FILE I/O BUFFERS.
007760  010167  177656          6$:     MOV     R1,IBUF1        ;INPUT BUFFER 1.
007764  010167  177672                  MOV     R1,IRPOS        ;ADDRESS OF 1ST. REC.
007770  010167  177664                  MOV     R1,IBFEND       ;FORCE SETUP OF 1ST BUFFER.
007774  066701  000000G                 ADD     OBUFSZ,R1
010000  010167  177640                  MOV     R1,IBUF2        ;INPUT BUFFER 2.
010004  066701  000000G                 ADD     OBUFSZ,R1
010010  010167  000000G                 MOV     R1,OBUF1        ;OUTPUT BUFFER 1.
010014  066701  000000G                 ADD     OBUFSZ,R1
010020  010167  000000G                 MOV     R1,OBUF2        ;OUTPUT BUFFER 2.
010024  066701  000000G                 ADD     OBUFSZ,R1
                                ;  SET UP RECORD LOCATION TABLE.
010030  010167  177622                  MOV     R1,LSTKEY       ;BUFFER FOR LAST WINNER
010034  066701  000000G                 ADD     RWDLEN,R1
010040  005002                          CLR     R2
010042  016703  000000G                 MOV     TOPMEM,R3       ;GET HIGHEST AVAILABLE MEMORY ADDRESS.
010046  166703  000000G                 SUB     RWDLEN,R3       ;LEAVE ROOM FOR A RECORD.
010052  012704  000000'                 MOV     #LOC,R4         ;GET ADDRESS OF LOCATION TABLE.
010056  020103          2$:     CMP     R1,R3           ;IS THERE ROOM FOR ANOTHER RECORD?
010060  101010                  BHI     1$              ;BRANCH IF NOT.
010062  010124                  MOV     R1,(R4)+        ;SET ADDRESS OF RECORD.
010064  005202                  INC     R2
010066  020227  000620          CMP     R2,#MAXREC      ;ALLOCATED MAX # OF RECORDS?
010072  103003                  BHIS    1$              ;BRANCH IF YES.
010074  066701  000000G         ADD     RWDLEN,R1       ;MOVE ON TO NEXT RECORD.
010100  000766                  BR      2$
010102  010267  177532  1$:     MOV     R2,NUMREC       ;SAVE # OF RECORDS TO MERGE.
010106  020227  000004          CMP     R2,#4           ;CHECK FOR REASONABLE MINIMUM
010112  103004                  BHIS    5$
010114                          XXX     24.             ;ERROR IF NOT ENOUGH CORE
010124  010205  5$:     MOV     R2,R5
010126  005302                  DEC     R2              ;START J=P-1.
010130  010503  3$:     MOV     R5,R3           ;GET P.
010132  060203                  ADD     R2,R3           ;P+J.
010134  006203                  ASR     R3              ;(P+J)/2
010136  006303                  ASL     R3              ;CONVERT TO WORD TABLE INDEX.
010140  010204                  MOV     R2,R4           ;GET J.
010142  006204                  ASR     R4              ;J/2
010144  006304                  ASL     R4              ;CONVERT TO WORD TABLE INDEX.
010146  006302                  ASL     R2              ;CONVERT J TO WORD TBL INDEX.
010150  010362  003100'         MOV     R3,FE(R2)       ;EXTERNAL NODE FLINK.
```

ALLOCATE CORE SPACE

```
     58 010154  010462  004540'                 MOV     R4,FI(R2)          ;INTERNAL NODE FLINK.
     59 010160  005062  006200'                 CLR     RN(R2)             ;SAY NO RUNS IN FILE YET.
     60 010164  010262  001440'                 MOV     R2,LOSER(R2)
     61 010170  006202                          ASR     R2
     62 010172  005302                          DEC     R2
     63 010174  002355                          BGE     3$
     64                                ;  INITIALIZE FILE MANAGEMENT TABLES.
     65 010176  005002                 IFMT:    CLR     R2
     66 010200  005062  000000G        4$:      CLR     FILST(R2)          ;FILE STARTING BLOCK #
     67 010204  005062  000000G                 CLR     FILRN(R2)          ;# RUNS IN FILE
     68 010210  005062  000000G                 CLR     FILNB(R2)          ;# BLOCKS IN FILE
     69 010214  062702  000002                  ADD     #2,R2              ;DONE ALL?
     70 010220  020227  000000G                 CMP     R2,#MXFIL2
     71 010224  103765                          BLO     4$                 ;BR IF NOT
     72 010226  005067  000000G                 CLR     CUROFL             ;START OUTPUT TO FILE # 0
     73 010232  005067  177430                  CLR     OFLINK             ;NO RUNS YET IN 1ST FILE
     74 010236  012767  000001  000000G         MOV     #1,FILST           ;START FILE 0 WITH CLUSTER 1
     75 010244  012767  000001  000000G         MOV     #1,FILFL
     76 010252  012767  000002  000000G         MOV     #2,NXTBLK          ;NEXT FREE CLUSTER
     77 010260  016701  000000G                 MOV     OBUF1,R1           ;INITIALIZE OUTPUT BUFFER.
     78 010264  010167  000000G                 MOV     R1,OBFEND          ;SET END OF OUTPUT BUFFER
     79 010270  066767  000000G 000000G        ADD     OBUFSZ,OBFEND
     80 010276  005021                          CLR     (R1)+              ;ZERO FILE FLINK.
     81 010300  005021                          CLR     (R1)+              ;ZERO # OF RECORDS IN BLOCK.
     82 010302  010167  000000G                 MOV     R1,ORPOS           ;SET UP ADDRESS OF NXT OUTPUT REC.
     83 010306  105067  177366                  CLRB    EOFFLG             ;HAVEN'T HIT END OF FILE YET
     84                                ;  START INPUT READS
     85 010312  005067  000000G                 CLR     IFPNT              ;START WITH BLOCK 0 OF FILE
     86 010316  032767  000000G 000000G        BIT     #RT$OX,ITYPE       ;IS THIS AN ISAM FILE?
     87 010324  001402                          BEQ     1$                 ;BR IF NOT
     88 010326  005267  000000G                 INC     IFPNT              ;ISAM FILES START WITH BLOCK 1
     89 010332                         1$:      $READ   #FD1,IBUF2,OWCNT,IFPNT
     90 010512  005067  177132                  CLR     RC                 ;INITIALIZE MISC VALUES
     91 010516  005067  177124                  CLR     RQ
     92 010522  005067  177124                  CLR     RMAX
     93 010526  005067  177122                  CLR     Q
```

```
                                        .SBTTL  MAIN SORT ROUTINE
                                ;
                                ;  ENTER MAIN SORT ROUTINE
                                ;
010532  016701  177112          SR2:    MOV     RC,R1           ;GET CURRENT RUN NUMBER.
010536  020167  177104                  CMP     R1,RQ           ;DOES THIS RECORD GO IN THIS RUN?
010542  001471                          BEQ     SR3             ;BRANCH IF THIS RECORD IS IN THIS RUN.
                                ;
                                ;  WIND UP PROCESSING ON RUN NUMBER RC.
                                ;
010544  005767  177100                  TST     RC              ;IS THIS THE INITIALIZATION RUN?
010550  001463                          BEQ     ENDINI          ;BRANCH IF INITIALIZATION RUN.
010552  016702  000000G         RUNEND: MOV     CUROFL,R2       ;GET # OF CURRENT OUTPUT FILE.
010556  005262  000000G                 INC     FILRN(R2)       ;INCREMENT # OF RUNS IN THIS FILE.
010562  016703  000000G                 MOV     OBUF1,R3        ;GET ADDRESS OF CURRENT OUTPUT BUFFER.
010566  052713  100000                  BIS     #^O100000,(R3)  ;SET END OF RUN FLAG IN BUFFER.
010572  016767  177070  000000G        MOV     OFLINK,FLBLK    ;LINK FORWARD TO PREV RUNS IN FILE
010600  016701  000000G                 MOV     ORPOS,R1        ;CALCULATE # OF CHARS IN BUFFER
010604  160301                          SUB     R3,R1
010606  010163  000002                  MOV     R1,2(R3)        ;SET # OF CHARS IN BUFFER
010612  162701  000004                  SUB     #4.,R1          ;SUBTRACT # CONTROL BYTES IN BUFFER
010616  060167  000000G                 ADD     R1,TMPCNT       ;ACCUMULATE # DATA BYTES BEING WRITTEN
010622  005567  000002G                 ADC     TMPCNT+2        ;TO TEMP FILE
010626  004767  000000G                 CALL    FLUSH           ;WRITE OUT LAST BLK
010632  005567  000002G                 ADC     TMPCNT+2        ;PROPOGATE CARRY
                                ;
                                ;  ADVANCE TO NEXT OUTPUT RUN
                                ;
010636  062702  000002                  ADD     #2,R2           ;ADVANCE TO NEXT FILE NUMBER
010642  020227  000000G                 CMP     R2,#MXFIL2      ;HAVE WE PASSED LAST FILE?
010646  103401                          BLO     2$              ;BRANCH IF NOT.
010650  005002                          CLR     R2              ;RECYCLE TO FILE # 0.
010652  010267  000000G         2$:     MOV     R2,CUROFL
010656  026767  176764  176766        CMP     RQ,RMAX         ;FINISHED WITH LAST RUN?
010664  101402                          BLOS    3$              ;BR IF NOT
010666  000167  004134                  JMP     S1STOP          ;FINISH PHASE 1
010672  016267  000000G 176766  3$:     MOV     FILST(R2),OFLINK;SAVE POINTER TO REST OF FILE
010700  016703  000000G                 MOV     NXTBLK,R3       ;ASSIGN IT A STARTING BLOCK
010704  010362  000000G                 MOV     R3,FILST(R2)
010710  010367  000000G                 MOV     R3,FILFL
010714  005267  000000G                 INC     NXTBLK
010720  016767  176722  176722  ENDINI: MOV     RQ,RC           ;ADVANCE RUN NUMBER
                                ;
                                ;  OUTPUT TOP OF TREE.  (Q NOW POINTS TO THE CHAMPION AND RQ IS ITS
                                ;  RUN NUMBER)
                                ;
010726  005767  176714          SR3:    TST     RQ              ;IS THIS INITIALIZATION RUN?
010732  001466                          BEQ     SR4             ;BRANCH IF YES.
                                ;  MOVE RECORD(Q) TO OUTPUT BUFFER.
010734  016702  176714                  MOV     Q,R2            ;GET INDEX TO WINNER
010740  016203  000000'                 MOV     LOC(R2),R3      ;GET ADDRESS OF WINNER RECORD
010744  016762  176706  000000'        MOV     LSTKEY,LOC(R2)  ;LAST WINNER STAYS IN 'LSTKEY' BUFFER
010752  010367  176700                  MOV     R3,LSTKEY
010756  016702  000000G                 MOV     ORPOS,R2        ;POSITION IN OUTPUT BUFFER
010762  016705  000000G                 MOV     OBFEND,R5       ;ADDR OF END OF BUFFER
010766  010304                          MOV     R3,R4           ;POINT TO END OF RECORD
010770  066704  000000G                 ADD     RECLEN,R4
```

```
     58 010774  020205                    2$:     CMP     R2,R5           ;HIT END OF BUFFER?
     59 010776  103432                            BLO     1$              ;BR IF NOT
     60                                   ;  OUTPUT BUFFER IS FULL -- WRITE IT OUT
     61 011000  016701  000000G                   MOV     OBUF1,R1        ;POINT TO BUFFER HEADER
     62 011004  160102                            SUB     R1,R2           ;CALCULATE # OF CHARS IN BUFFER
     63 011006  010261  000002                    MOV     R2,2(R1)        ;STORE LENGTH IN HEADER
     64 011012  162702  000004                    SUB     #4.,R2          ;SUBTRACT # CONTROL BYTES IN BUFFER
     65 011016  060267  000000G                   ADD     R2,TMPCNT       ;COUNT # DATA BYTES WRITTEN TO TEMP FILE
     66 011022  005567  000002G                   ADC     TMPCNT+2
     67 011026  016767  000000G 000000G          MOV     NXTBLK,FLBLK    ;SET FILE FLINK
     68 011034  004767  000000G                   CALL    FLUSH           ;WRITE OUT THE BLOCK
     69 011040  005567  000002G                   ADC     TMPCNT+2        ;PROPOGATE CARRY
     70 011044  005267  000000G                   INC     NXTBLK          ;POINT TO NEXT FREE TEMP BLOCK
     71 011050  016702  000000G                   MOV     OBUF1,R2        ;START RECORD IN NEW BUFFER
     72 011054  062702  000004                    ADD     #4,R2           ;SKIP HEADER WORDS
     73 011060  016705  000000G                   MOV     OBFEND,R5       ;GET ADDR OF END OF BUFFER
     74                                   ;  CONTINUE MOVING RECORD
     75 011064  112322                    1$:     MOVB    (R3)+,(R2)+     ;MOVE CHAR TO OUTPUT BUFFER
     76 011066  001004                            BNE     3$              ;BR IF NOT NULL
     77 011070  032767  000000G 000000G           BIT     #RT$VLN,ITYPE   ;ARE RECORDS VARIABLE LENGTH?
     78 011076  001002                            BNE     4$              ;IF NOT THEN CONTINUE MOVING
     79 011100  020304                    3$:     CMP     R3,R4           ;MOVED ALL OF RECORD?
     80 011102  103734                            BLO     2$              ;BR IF MORE TO MOVE
     81 011104  010267  000000G           4$:     MOV     R2,ORPOS        ;SAVE START OF NEXT RECORD POINTER
     82                                   ;
     83                                   ;  INPUT NEW RECORD INTO POSITION LOC(Q).
     84                                   ;
     85 011110  016702  176540            SR4:    MOV     Q,R2            ;GET ADDRESS OF RECORD(Q)
     86 011114  016201  000000'                   MOV     LOC(R2),R1
     87 011120  004767  000232                    CALL    GETREC          ;GET NEXT LOGICAL INPUT RECORD
     88 011124  103423                            BCS     EOF             ;BRANCH IF END OF FILE HIT.
     89 011126  005767  176516                    TST     RC              ;INITIALIZATION RUN?
     90 011132  001405                            BEQ     1$              ;BRANCH IF YES.
     91 011134  016702  176516                    MOV     LSTKEY,R2       ;GET ADDRESS OF LAST WINNER.
     92 011140  004777  000000G                   CALL    @COMPAR         ;KEY(Q):KEY(LSTKEY)
     93 011144  002020                            BGE     SR5             ;BRANCH IF KEY(Q)>=KEY(LSTKEY).
     94 011146  016703  176474            1$:     MOV     RQ,R3           ;GET RUN # OF CURRENT RECORD.
     95 011152  005203                            INC     R3              ;PUT NEW RECORD IN NEXT RUN.
     96 011154  010367  176466                    MOV     R3,RQ
     97 011160  020367  176466                    CMP     R3,RMAX         ;SAVE NUMBER OF HIGHEST RUN.
     98 011164  101410                            BLOS    SR5
     99 011166  010367  176460                    MOV     R3,RMAX
    100 011172  000405                            BR      SR5
    101                                  ;
    102                                  ;  WE JUST HIT THE END OF THE INPUT FILE
    103                                  ;
    104 011174  016767  176452  176444   EOF:    MOV     RMAX,RQ         ;SET RQ TO FLUSH OUT REMAINING
    105 011202  005267  176440                    INC     RQ              ;RECORDS.
    106                                  ;  PREPARE TO UPDATE  (NOW Q POINTS TO A NEW RECORD WHOSE RUN
    107                                  ;  NUMBER IS IN RQ.
    108 011206  016701  176442           SR5:    MOV     Q,R1            ;GET INDEX TO NEW RECORD.
    109 011212  016104  003100'                   MOV     FE(R1),R4       ;INDEX TO FATHER NODE (KNUTH'S 'T').
    110                                  ;  SET NEW LOSER
    111 011216  016701  176424           SR6:    MOV     RQ,R1           ;GET RUN # OF NEW RECORD.
    112 011222  001435                            BEQ     SR7             ;BRANCH IF INITIALIZATION RUN.
    113 011224  020164  006200'                   CMP     R1,RN(R4)       ;RQ:RN(T)
    114 011230  101016                            BHI     1$              ;BRANCH IF NEW RECORD GOES IN NEXT RUN.
```

```
    115 011232  001031                                BNE     SR7
    116 011234  016401  001440'                       MOV     LOSER(R4),R1    ;GET LOSER(T).
    117 011240  016101  000000'                       MOV     LOC(R1),R1      ;GET LOC(LOSER(T)).
    118 011244  016703  176404                        MOV     Q,R3            ;GET  INDEX OF NEW RECORD.
    119 011250  016302  000000'                       MOV     LOC(R3),R2      ;GET LOCATION OF NEW RECORD.
    120 011254  004777  000000G                       CALL    @COMPAR         ;LOSER(T):Q.
    121 011260  002016                                BGE     SR7             ;BRANCH IF LOSER(T)>=Q.
    122                                     ;  RECORD(Q) BECOMES THE NEW LOSER AT THIS LEVEL.
    123 011262  016701  176360                        MOV     RQ,R1
    124 011266  016467  006200' 176352  1$:   MOV     RN(R4),RQ       ;RN(T)-->RQ.
    125 011274  010164  006200'                       MOV     R1,RN(R4)       ;RQ-->RN(T).
    126 011300  016703  176350                        MOV     Q,R3
    127 011304  016467  001440' 176342                MOV     LOSER(R4),Q     ;LOSER(T)-->Q.
    128 011312  010364  001440'                       MOV     R3,LOSER(R4)    ;Q-->LOSER(T).
    129                                     ;  MOVE UP TREE.   (T IS ADVANCED TO POINTER TO NEXT HIGHER NODE)
    130 011316  020427  000002            SR7:  CMP     R4,#2           ;IS T POINTING TO NODE 1?
    131 011322  001403                                BEQ     1$              ;BR IF YES
    132 011324  016404  004540'                       MOV     FI(R4),R4       ;FI(T)-->T.   MOVE T UP TO ITS FATHER.
    133 011330  000732                                BR      SR6
    134                                     ;  WE HAVE PROPOGATED THE WINNER RECORD TO THE TOP OF THE TREE.
    135 011332  005767  176314            1$:   TST     RMAX            ;IS INPUT FILE NULL?
    136 011336  001402                                BEQ     2$              ;BR IF YES
    137 011340  000167  177166                        JMP     SR2             ;GO PROCESS NEXT INPUT RECORD
    138                                     ;  INPUT FILE IS NULL
    139 011344  012767  000001  176274    2$:   MOV     #1,RQ           ;FORCE FINAL CLOSEOUT
    140 011352  000167  177174                        JMP     RUNEND
```

```
                                        .SBTTL  I/O ROUTINES
                                ;
                                ;-----------------------------------------------------------------------
                                ;  GETREC is called to obtain the next logical record from the
                                ;  input file.  GETREC is responsible for reading records,
                                ;  doing record selection, doing field mapping and checking
                                ;  for end of file.
                                ;  When called, R1 must contain the address of the buffer where
                                ;  the record is to be stored.
                                ;  On return, the C-flag is set if End of file is hit.
                                ;
011356  105767  176316          GETREC: TSTB    EOFFLG          ;HAVE WE HIT END OF FILE PREVIOUSLY?
011362  001402                          BEQ     1$              ;BR IF NOT
011364  000261                          SEC                     ;SIGNAL END OF FILE AGAIN
011366  000207                          RETURN
011370  010167  176276          1$:     MOV     R1,RECPTR       ;SAVE ADDRESS OF RECORD BUFFER
                                ;
                                ;  If field mapping is wanted, move the record to a logical buffer
                                ;  first.
                                ;
011374  005767  000000G                 TST     FLDMAP          ;IS FIELD MAPPING WANTED?
011400  001402                          BEQ     GRREAD          ;BR IF NOT
011402  016701  176262                  MOV     MAPBUF,R1       ;GET ADDRESS OF HOLDING BUFFER
                                ;
                                ;  See if we have already read max allowed number of records.
                                ;
011406  026767  000000G 000000G GRREAD: CMP     RCNT1,MXREC1    ;READ MAX ALLOWED # RECORDS?
011414  001004                          BNE     1$              ;BR IF NOT
011416  026767  000000G 000000G         CMP     RCNT2,MXREC2    ;CHECK HIGH ORDER PART OF NUMBER
011424  001506                          BEQ     GREOF           ;READ MAX # OF RECORDS -- SIGNAL EOF
                                ;
                                ;  Read a record from input file into buffer pointed to by R1.
                                ;
011426  004767  000234          1$:     CALL    READIN          ;READ A RECORD
011432  103503                          BCS     GREOF           ;BR IF END OF FILE
                                ;
                                ;  Check for control-z as DIBOL end-of-file sentinel.
                                ;
011434  032767  000000G 000000G         BIT     #RT$DBL,ITYPE   ;IS THIS A DIBOL DATA FILE?
011442  001403                          BEQ     3$              ;BR IF NOT
011444  121127  000032                  CMPB    (R1),#CTRLZ     ;IS CONTROL-Z THE 1ST CHAR IN RECORD?
011450  001474                          BEQ     GREOF           ;IF YES THEN THIS IS EOF
                                ;
                                ;  Count number of records read.
                                ;
011452  005267  000000G         3$:     INC     RCNT1           ;INCREMENT LOW ORDER COUNTER
011456  001002                          BNE     2$              ;BR IF NO OVERFLOW
011460  005267  000000G                 INC     RCNT2           ;INCREMENT HIGH ORDER COUNTER
                                ;
                                ;  See if we need to skip over some records at front of file.
                                ;
011464  005767  000000G         2$:     TST     SKPCNT          ;IS SKIPPING WANTED?
011470  001403                          BEQ     GRCKEF          ;BR IF NOT
011472  005367  000000G                 DEC     SKPCNT          ;DEC # LEFT TO SKIP
011476  000743                          BR      GRREAD          ;GO READ NEXT RECORD
                                ;  See if this record contains user specified EOF character.
011500  016700  000000G         GRCKEF: MOV     EOFCOL,R0       ;DID USER SPECIFY EOF CHARACTER?
```

```
     58 011504  001414                          BEQ     2$              ;BR IF NOT
     59 011506  060100                          ADD     R1,R0           ;POINT TO COLUMN WITH CHARACTER
     60 011510  126760  000000G 177777          CMPB    EOFCHR,-1(R0)   ;IS THIS EOF RECORD?
     61 011516  001007                          BNE     2$              ;BR IF NOT
     62 011520  105267  176154                  INCB    EOFFLG          ;REMEMBER WE'VE HIT EOF
     63 011524  105267  000000G                 INCB    UEOF            ;REMEMBER THIS IS USER EOF RECORD
     64 011530  004767  003224                  CALL    EOFSAV          ;SAVE THE EOF RECORD
     65 011534  000442                          BR      GREOF           ;SIGNAL END OF FILE
     66                                ;
     67                                ;  Do any required record selection.
     68                                ;
     69 011536  005767  000000G        2$:     TST     SELADR          ;IS RECORD SELECTION WANTED?
     70 011542  001406                          BEQ     3$              ;BR IF NOT
     71 011544  004767  002250                  CALL    SELECT          ;SEE IF WE SHOULD INCLUDE THIS RECORD
     72 011550  103003                          BCC     3$              ;BR IF WE SHOULD ACCEPT THE RECORD
     73 011552  105067  000000G                 CLRB    UEOF            ;CLEAR USER-EOF RECORD FLAG
     74 011556  000713                          BR      GRREAD          ;GO READ NEXT RECORD
     75                                ;
     76                                ;  Count number of records passed to sort.
     77                                ;
     78 011560  062767  000001  000000G 3$:     ADD     #1,SRTCNT       ;COUNT # RECORDS SORTED
     79 011566  005567  000002G                 ADC     SRTCNT+2        ;PROPOGATE CARRY
     80                                ;
     81                                ;  Do any required field mapping.
     82                                ;
     83 011572  005767  000000G                 TST     FLDMAP          ;IS FIELD MAPPING WANTED?
     84 011576  001404                          BEQ     6$              ;BR IF NOT
     85 011600  004767  002620                  CALL    DOMAP           ;DO FIELD MAPPING
     86 011604  016701  176062                  MOV     RECPTR,R1       ;POINT TO REAL RECORD BUFFER AREA
     87                                ;
     88                                ;  Insert BCD record number if wanted.
     89                                ;
     90 011610  005767  000000G        6$:     TST     NUMLEN          ;IS RECORD NUMBER WANTED?
     91 011614  001402                          BEQ     7$              ;BR IF NOT
     92 011616  004767  002772                  CALL    INSNUM          ;INSERT RECORD NUMBER
     93                                ;
     94                                ;  Strip trailing spaces from ascii records.
     95                                ;
     96 011622  032767  000000G 000000G 7$:     BIT     #RT$VLN,ITYPE   ;SHOULD WE STRIP TRAILING SPACES?
     97 011630  001402                          BEQ     8$              ;BR IF NOT
     98 011632  004767  003060                  CALL    STRIP           ;STRIP OFF TRAILING SPACES
     99                                ;
    100                                ;  This is a good record.
    101                                ;
    102 011636  000241                 8$:     CLC                     ;SAY NOT END OF FILE
    103 011640  000207                          RETURN
    104                                ;
    105                                ;  We just hit the end of the input file.
    106                                ;  See if there are more input files to process.
    107                                ;
    108 011642  004767  001536         GREOF:  CALL    NEXTIN          ;TRY TO SWITCH TO NEXT INPUT FILE
    109 011646  103403                          BCS     1$              ;BR IF NO MORE INPUT FILES
    110                                ;  There is another input file.
    111 011650  016701  176016                  MOV     RECPTR,R1       ;RECOVER RECORD BUFFER POINTER
    112 011654  000640                          BR      GETREC          ;GO READ RECORD FROM NEW INPUT FILE
    113                                ;
    114                                ;  We have reached the end of the last input file.
```

```
                                        ;  Return with C-flag set.
                                        ;
011656  105267  176016          1$:     INCB    EOFFLG          ;REMEMBER WE HAVE HIT EOF
011662  000261                          SEC                     ;SIGNAL END OF FILE
011664  000207                          RETURN
```

```
                                        ;
                                        ;-------------------------------------------------------------
                                        ;  ROUTINE TO GET THE NEXT RECORD FROM THE INPUT STREAM.
                                        ;  THE RECORD IS MOVED TO THE LOCATION POINTED TO
                                        ;  BY R1.   IF AN END OF FILE IS HIT THE "C"-INDICATOR
                                        ;  IS SET OTHERWISE IT IS CLEARED.
                                        ;
011666  105767  176006        READIN: TSTB    EOFFLG          ;ARE WE PAST END OF FILE?
011672  001402                        BEQ     5$              ;BR IF NOT
011674  000261                        SEC                     ;SIGNAL END OF FILE
011676  000207                        RETURN
011700  010246                5$:     MOV     R2,-(SP)
011702  010346                        MOV     R3,-(SP)
011704  010446                        MOV     R4,-(SP)
011706  010546                        MOV     R5,-(SP)
011710  010146                        MOV     R1,-(SP)
011712  016702  000000G               MOV     INRLEN,R2       ;GET DESIRED LENGTH OF INPUT RECORDS
011716  016704  175740                MOV     IRPOS,R4        ;GET POINTER INTO CURRENT FILE BUFFER
011722  016705  175732                MOV     IBFEND,R5       ;GET POINTER PAST END OF CURRENT BUFFER
                                        ;
                                        ;  Determine what type of record this is.
                                        ;
011726  016700  000000G               MOV     ITYPE,R0        ;GET RECORD TYPE FLAGS
011732  032700  000000G               BIT     #RT$CBL,R0      ;COBOL RECORD?
011736  001416                        BEQ     1$              ;BR IF NOT
011740  032700  000000G               BIT     #RT$OX,R0       ;INDEXED ORGANIZATION FILE?
011744  001402                        BEQ     3$              ;BR IF NOT
011746  000167  000656                JMP     RICX            ;GO GET COBOL ISAM FILE RECORD
011752  032700  000000G        3$:     BIT     #RT$RB,R0       ;COBOL BINARY RECORD?
011756  001402                        BEQ     2$              ;BR IF NOT
011760  000167  000514                JMP     RICB            ;GO HANDLE BINARY RECORD
011764  032700  000000G        2$:     BIT     #RT$OR,R0       ;COBOL ASCII RELATIVE RECORD?
011770  001113                        BNE     RICAR           ;BR IF YES
011772  000436                        BR      RICA            ;COBOL ASCII SEQUENTIAL RECORD
011774  032700  000000G        1$:     BIT     #RT$CR,R0       ;IS CARRIAGE-RETURN THE RECORD TERMINATOR?
012000  001033                        BNE     RICA            ;BR IF YES
012002  032700  000000G               BIT     #RT$F11,R0      ;FILES-11 SEQUENTIAL??
012006  001403                        BEQ     RIFL            ;NOT FILES-11
012010  032700  000000G               BIT     #RT$OS,R0       ;   SEQUENTIAL?
012014  001145                        BNE     RIRS            ;BRANCH IF SO
                                        ;
                                        ;  Fixed length input record.
                                        ;
012016  004767  002530        RIFL:   CALL    INCNUM          ;INCREMENT RECORD COUNT
012022  160405                3$:     SUB     R4,R5           ;CALC # BYTES LEFT IN INPUT BUFFER
012024  010203                        MOV     R2,R3           ;GET # BYTES WANTED FOR RECORD
012026  020305                        CMP     R3,R5           ;GOT ENOUGH BYTES LEFT IN BUFFER?
012030  101402                        BLOS    1$              ;BR IF YES
012032  010503                        MOV     R5,R3           ;MOVE THIS MANY BYTES THIS TIME
012034  001410                        BEQ     4$              ;BR IF BUFFER EMPTY
012036  160302                1$:     SUB     R3,R2           ;THIS MANY BYTES NEEDED AFTER THIS MOVE
012040  112421                2$:     MOVB    (R4)+,(R1)+     ;MOVE BYTES FROM INPUT BUFFER TO RECORD BUFFER
012042  005303                        DEC     R3              ;NEED MORE?
012044  003375                        BGT     2$              ;BR IF YES
012046  005702                        TST     R2              ;DO WE NEED TO GET MORE OUT OF NEXT BUFFER?
012050  001002                        BNE     4$              ;BR IF YES
012052  000167  000374                JMP     RIEND
```

```
     58 012056  004767  000726          4$:     CALL    RIBUF           ;READ NEW BUFFER FULL
     59 012062  103357                          BCC     3$              ;BR IF NOT END OF FILE
     60 012064  000167  000404                  JMP     RIEOF           ;HIT END OF FILE
```

```
                                ;
                                ;  Variable length ASCII records terminated by carriage-return.
                                ;
012070  004767  002456          RICA:   CALL    INCNUM          ;INCREMENT RECORD COUNT
012074  020405                  6$:     CMP     R4,R5           ;ARE WE BEYOND END OF INPUT BUFFER?
012076  103403                          BLO     1$              ;BR IF NOT
012100  004767  000704                  CALL    RIBUF           ;READ NEXT INPUT BUFFER FULL
012104  103573                          BCS     RIEOF           ;BR IF END OF FILE HIT
012106  112400                  1$:     MOVB    (R4)+,R0        ;GET NEXT CHAR FROM INPUT BUFFER
012110  001005                          BNE     2$              ;BR IF NOT NULL CHAR
012112  032767  000000G 000000G        BIT     #RT$CBL,ITYPE   ;FILE = COBOL SEQUENTIAL ORGANIZATION?
012120  001165                          BNE     RIEOF           ;IF YES THEN NULL SIGNALS END OF FILE
012122  000764                          BR      6$              ;IGNORE NULLS OTHERWISE
012124  020027  000032          2$:     CMP     R0,#32          ;IS THIS A CONTROL CHARACTER?
012130  101012                          BHI     4$              ;BR IF NOT
012132  001560                          BEQ     RIEOF           ;CONTROL-Z ==> END OF FILE
012134  120027  000015                  CMPB    R0,#15          ;CARRIAGE-RETURN?
012140  001417                          BEQ     5$              ;BR IF GOT END OF RECORD
012142  120027  000012                  CMPB    R0,#12          ;LINE FEED?
012146  001752                          BEQ     6$              ;IGNORE LINE FEEDS
012150  120027  000014                  CMPB    R0,#14          ;FORM FEED?
012154  001747                          BEQ     6$              ;IGNORE FORM FEEDS
012156  005702                  4$:     TST     R2              ;ROOM IN RECORD BUFFER FOR ANOTHER CHAR?
012160  003403                          BLE     8$              ;BR IF NOT
012162  110021                          MOVB    R0,(R1)+        ;MOVE CHAR TO RECORD BUFFER
012164  005302                          DEC     R2              ;COUNT # CHARS LEFT TO GET
012166  000742                          BR      6$              ;KEEP LOOKING FOR END OF RECORD
012170                          8$:     XXX     33.             ;RECORD READ IS LONGER THAN INPUT BUFFER
                                ;
                                ;  Found end of record.
                                ;  Pad rest of buffer with blanks.
012200  005702                  5$:     TST     R2              ;DID WE COMPLETELY FILL BUFFER?
012202  003523                          BLE     RIEND           ;BR IF YES
012204  012703  000040                  MOV     #' ,R3          ;GET FILLER SPACE
012210  110321                  7$:     MOVB    R3,(R1)+        ;FILL RECORD AREA WITH BLANKS
012212  005302                          DEC     R2
012214  003375                          BGT     7$
012216  000515                          BR      RIEND
```

```
                                ;
                                ;  COBOL ascii record in relative organization file.
                                ;
012220  004767  002326          RICAR:  CALL    INCNUM          ;INCREMENT RECORD COUNTER
012224  020405                          CMP     R4,R5           ;IS BUFFER EMPTY?
012226  103403                          BLO     1$              ;BR IF NOT
012230  004767  000554                  CALL    RIBUF           ;READ IN ANOTHER BUFFER FULL
012234  103517                          BCS     RIEOF           ;BR IF EOF HIT
012236  105724                  1$:     TSTB    (R4)+           ;DOES RECORD EXIST?
012240  001014                          BNE     4$              ;BR IF YES
                                ;  Record does not exist.  Skip up to next one.
012242  010203                          MOV     R2,R3           ;GET RECORD LENGTH
012244  005203                          INC     R3              ;COUNT CR AT END OF RECORD
012246  060403                  3$:     ADD     R4,R3           ;GET POINTER TO START OF NEXT RECORD
012250  020305                          CMP     R3,R5           ;PAST END OF CURRENT BUFFER?
012252  103405                          BLO     2$              ;BR IF NOT
012254  160503                          SUB     R5,R3           ;COUNT # CHARS SKIPPED OVER IN THIS BUFFER
012256  004767  000526                  CALL    RIBUF           ;READ IN NEW BUFFER FULL
012262  103504                          BCS     RIEOF           ;BR IF EOF HIT
012264  000770                          BR      3$              ;CONTINUE SKIPPING
012266  010304                  2$:     MOV     R3,R4           ;GET POINTER TO START OF NEXT RECORD
012270  000753                          BR      RICAR           ;SEE IF THAT RECORD EXISTS
                                ;  Record exists -- Move to record buffer.
012272  020405                  4$:     CMP     R4,R5           ;BUFFER EMPTY?
012274  103403                          BLO     5$              ;BR IF NOT
012276  004767  000506                  CALL    RIBUF           ;READ NEW BUFFER LOAD
012302  103474                          BCS     RIEOF           ;BR IF EOF HIT
012304  112421                  5$:     MOVB    (R4)+,(R1)+     ;MOVE TO RECORD BUFFER
012306  005302                          DEC     R2              ;MOVE FULL RECORD SIZE
012310  003370                          BGT     4$
                                ;  Skip carriage-return at end of record.
012312  020405                          CMP     R4,R5           ;PAST END OF BUFFER?
012314  103403                          BLO     6$              ;BR IF NOT
012316  004767  000466                  CALL    RIBUF           ;READ NEW BUFFER FULL
012322  103464                          BCS     RIEOF           ;BR IF END OF FILE HIT
012324  005204                  6$:     INC     R4              ;SKIP OVER CR
012326  000451                          BR      RIEND
```

```
                                ;
                                ;  FILES-11 sequential
                                ;  (First word contains record length)
                                ;
012330                          RIRS:
012330  005204                          INC     R4              ;ROUND R4 UP TO NEAREST WORD
012332  042704  000001                  BIC     #1,R4
012336  020405                          CMP     R4,R5           ;IS INPUT BUFFER EMPTY?
012340  103403                          BLO     4$              ;BR IF NOT
012342  004767  000442                  CALL    RIBUF           ;READ NEW BUFFER FULL
012346  103452                          BCS     RIEOF           ;BR IF END OF FILE HIT
                                ;  Get record control word.
012350  004767  002176          4$:     CALL    INCNUM          ;INCREMENT RECORD COUNTER
012354  012403                          MOV     (R4)+,R3        ;GET RECORD SIZE
012356  001417                          BEQ     6$              ;BR IF RECORD LENGTH = 0
012360  160302                          SUB     R3,R2           ;COMPARE RECORD LENGTH WITH BUFFER SIZE
012362  002004                          BGE     1$              ;BR IF IT WILL FIT
012364                                  XXX     33.             ;RECORD IS LONGER THAN BUFFER
012374  006203                  1$:     ASR     R3              ;GET # WORDS IN RECORD
012376  020405                  3$:     CMP     R4,R5           ;PASSED END OF INPUT BUFFER?
012400  103403                          BLO     2$              ;BR IF NOT
012402  004767  000402                  CALL    RIBUF           ;READ NEW BUFFER FULL
012406  103432                          BCS     RIEOF           ;BR IF END OF FILE HIT
012410  012421                  2$:     MOV     (R4)+,(R1)+     ;MOVE WORD FROM FILE BUFFER TO RECORD BUFFER
012412  005303                          DEC     R3              ;MORE TO MOVE?
012414  003370                          BGT     3$              ;BR IF YES
                                ;  See if we need to pad with trailing nulls.
012416                          6$:
012416  005003                          CLR     R3              ;R3 = NULL, FOR PADDING END
012420  036767  000000G 000000G        BIT     RT$RB,ITYPE     ;BINARY RECORDS?
012426  001002                          BNE     7$              ;BRANCH IF SO
012430  112703  000040                  MOVB    #' ,R3          ;MUST BE ASCII, USE BLANK, NOT NULL
012434                          7$:
012434  005702                          TST     R2              ;DO WE NEED ANY PADDING NULLS AT END?
012436  003405                          BLE     RIEND           ;BR IF NOT
012440  110321                  5$:     MOVB    R3,(R1)+        ;PUT IN PADDING NULLS
012442  005302                          DEC     R2
012444  003375                          BGT     5$
012446  000167  000000                  JMP     RIEND
```

```
                                ;
                                ;  Finished getting record.
                                ;
012452  000241                RIEND:  CLC                     ;SIGNAL NO EOF
012454  010467  175202        RIFIN:  MOV     R4,IRPOS        ;SAVE POINTER INTO FILE BUFFER
012460  012601                        MOV     (SP)+,R1
012462  012605                        MOV     (SP)+,R5
012464  012604                        MOV     (SP)+,R4
012466  012603                        MOV     (SP)+,R3
012470  012602                        MOV     (SP)+,R2
012472  000207                        RETURN
                                ;
                                ;  End of file hit.
                                ;
012474  000261                RIEOF:  SEC                     ;SIGNAL END OF FILE
012476  000766                        BR      RIFIN           ;RETURN
```

```
                                ;
                                ;  COBOL binary records.
                                ;  (First word contains record-exists flag & record length)
                                ;
012500  020405                  RICB:   CMP     R4,R5           ;IS INPUT BUFFER EMPTY?
012502  103403                          BLO     RSBGCW          ;BR IF NOT
012504  004767  000300                  CALL    RIBUF           ;READ NEW BUFFER FULL
012510  103771                          BCS     RIEOF           ;BR IF END OF FILE HIT
                                ;  Get record control word.
012512  004767  002034          RSBGCW: CALL    INCNUM          ;INCREMENT RECORD COUNTER
012516  012403                          MOV     (R4)+,R3        ;GET RECORD CONTROL WORD
012520  002413                          BLT     RSBMOV          ;BR IF RECORD EXISTS
012522  001764                          BEQ     RIEOF           ;BR IF END OF FILE HIT
                                ;  Record is marked as deleted.  Skip to start of next record.
012524  060403                  1$:     ADD     R4,R3           ;GET ADDRESS OF START OF NEXT RECORD
012526  020305                          CMP     R3,R5           ;BEYOND END OF CURRENT INPUT BUFFER?
012530  103405                          BLO     2$              ;BR IF NOT
012532  160503                          SUB     R5,R3           ;SUBTRACT AMT IN BUFFER WE ARE SKIPPING OVER
012534  004767  000250                  CALL    RIBUF           ;READ NEXT BUFFER FULL
012540  103755                          BCS     RIEOF           ;BR IF EOF HIT
012542  000770                          BR      1$              ;CONTINUE SKIPPING
012544  010304                  2$:     MOV     R3,R4           ;GET ADDRESS OF START OF NEXT RECORD
012546  000761                          BR      RSBGCW          ;GO CHECK NEXT RECORD
                                ;  Record exists.
                                ;  Move it to record buffer.
012550  042703  100000          RSBMOV: BIC     #100000,R3      ;CLEAR RECORD-EXISTS FLAG
012554  001417                          BEQ     6$              ;BR IF RECORD LENGTH = 0
012556  160302                          SUB     R3,R2           ;COMPARE RECORD LENGTH WITH BUFFER SIZE
012560  002004                          BGE     1$              ;BR IF IT WILL FIT
012562                                  XXX     33.             ;RECORD IS LONGER THAN BUFFER
012572  006203                  1$:     ASR     R3              ;GET # WORDS IN RECORD
012574  020405                  3$:     CMP     R4,R5           ;PASSED END OF INPUT BUFFER?
012576  103403                          BLO     2$              ;BR IF NOT
012600  004767  000204                  CALL    RIBUF           ;READ NEW BUFFER FULL
012604  103733                          BCS     RIEOF           ;BR IF END OF FILE HIT
012606  012421                  2$:     MOV     (R4)+,(R1)+     ;MOVE WORD FROM FILE BUFFER TO RECORD BUFFER
012610  005303                          DEC     R3              ;MORE TO MOVE?
012612  003370                          BGT     3$              ;BR IF YES
                                ;  See if we need to pad with trailing nulls.
012614  005702                  6$:     TST     R2              ;DO WE NEED ANY PADDING NULLS AT END?
012616  003715                          BLE     RIEND           ;BR IF NOT
012620  105021                  5$:     CLRB    (R1)+           ;PUT IN PADDING NULLS
012622  005302                          DEC     R2
012624  003375                          BGT     5$
012626  000711                          BR      RIEND
```

```
                                ;
                                ;  COBOL Indexed organization (ISAM) file record.
                                ;
012630  020405                RICX:   CMP     R4,R5           ;IS BUFFER EMPTY?
012632  103403                        BLO     3$              ;BR IF NOT
012634  004767  000132              CALL    RDFILE          ;READ IN NEW BUFFER FULL
012640  103715                        BCS     RIEOF           ;BR IF END OF INPUT HIT
                                ;  See if record exists.
012642  005724                3$:     TST     (R4)+           ;TEST RECORD CONTROL WORD
012644  001002                        BNE     4$              ;BR IF RECORD IS DELETED
012646  005714                        TST     (R4)            ;TEST FOR END OF DELETE CHAIN
012650  001006                        BNE     RCXE            ;BR IF RECORD EXISTS
                                ;  Record is deleted.  Skip to start of next record.
012652  016703  000000G         4$:     MOV     ISMARS,R3       ;GET LENGTH OF RECORD TO SKIP OVER
012656  004767  000056              CALL    SKPXRC          ;SKIP OVER THE RECORD
012662  103704                        BCS     RIEOF           ;BR IF END OF INPUT HIT
012664  000766                        BR      3$              ;GO CHECK NEXT RECORD
                                ;  Record exists.  Move it to internal record buffer.
012666  016703  000000G         RCXE:   MOV     INRLEN,R3       ;GET LOGICAL RECORD LENGTH (BYTES)
012672  006203                        ASR     R3              ;CONVERT TO # WORDS
012674  020405                3$:     CMP     R4,R5           ;IS INPUT BUFFER EMPTY?
012676  103403                        BLO     2$              ;BR IF NOT
012700  004767  000066              CALL    RDFILE          ;READ IN A NEW BUFFER FULL
012704  103673                        BCS     RIEOF           ;BR IF END OF FILE HIT
012706  012421                2$:     MOV     (R4)+,(R1)+     ;MOVE RECORD TO INTERNAL BUFFER
012710  005303                        DEC     R3              ;MORE TO MOVE?
012712  003370                        BGT     3$              ;BR IF YES
                                ;  Skip over any accession numbers at end of record.
012714  016703  000000G               MOV     ISMARS,R3       ;GET TOTAL RECORD LENGTH
012720  166703  000000G               SUB     INRLEN,R3       ;SUBTRACT LOGICAL RECORD LENGTH
012724  001402                        BEQ     4$              ;BR IF NO ACCESSION NUMBERS
012726  004767  000006              CALL    SKPXRC          ;SKIP OVER ACCESSION NUMBERS
                                ;  Increment record counter.
012732  004767  001614        4$:     CALL    INCNUM          ;INCREMENT RECORD COUNTER
                                ;  Finished
012736  000645                        BR      RIEND
```

```
                                ;-----------------------------------------------------------------------------
                                ;  SKPXRC is called to skip over a part of an ISAM record.
                                ;
                                ;  Inputs:
                                ;   R3 = Number of bytes to skip over.
                                ;   R4 = Pointer to start of data to be skipped over.
                                ;   R5 = Pointer to end of buffer.
                                ;
                                ;  Outputs:
                                ;   R4 = Pointer to start of data following skipped data.
                                ;   R5 = Pointer to end of buffer.
                                ;   C-flag set on return if end of input hit.
                                ;
012740  010346                  SKPXRC: MOV     R3,-(SP)
012742  060403                  1$:     ADD     R4,R3           ;POINT TO START OF NEXT RECORD
012744  020305                          CMP     R3,R5           ;IS IT PAST END OF THE BUFFER?
012746  103405                          BLO     2$              ;BR IF NOT
012750  160503                          SUB     R5,R3           ;SUBTRACT AMT WE ARE SKIPPING OVER IN THIS BUFFER
012752  004767  000014                  CALL    RDFILE          ;READ IN NEW BUFFER FULL
012756  103403                          BCS     4$              ;BR IF END OF INPUT HIT
012760  000770                          BR      1$              ;SEE IF WE HAVE REACHED DESIRED POINT
012762  010304                  2$:     MOV     R3,R4           ;RETURN RECORD POINTER IN R4
012764  000241                          CLC                     ;SIGNAL SUCCESS
012766  012603                  4$:     MOV     (SP)+,R3
012770  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------------
                                ;  RDFILE is called to read the next buffer full of data.
                                ;  If end of file is hit, NEXTIN is called to try to switch to the
                                ;  next input file.
                                ;
                                ;  Outputs:
                                ;   R4 = Pointer to start of buffer read.
                                ;   R5 = Pointer to end of buffer.
                                ;   C-flag set on return if end of all input was hit.
                                ;
012772  004767  000012          RDFILE: CALL    RIBUF           ;TRY TO READ NEXT BUFFER FULL
012776  103003                          BCC     1$              ;BR IF GOT ONE
                                ;  Hit end of file.
                                ;  Call NEXTIN to switch to next input file.
013000  005003                          CLR     R3              ;ZERO ANY REMAINING SKIP BYTES
013002  004767  000376                  CALL    NEXTIN          ;TRY TO SWITCH TO NEXT INPUT FILE
013006  000207                  1$:     RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Read the next input buffer full.
                                ;  On return R4 = Address of start of buffer, R5 = Address of end of buffer.
                                ;  The c-flag is set on return if end of file is hit.
                                ;
013010                          RIBUF:  $WAIT   #FD1            ;WAIT FOR PREVIOUS READ TO FINISH
013124  103006                          BCC     6$              ;BR IF NO ERROR
013126  105700                          TSTB    R0              ;REAL ERROR?
013130  001404                          BEQ     6$              ;BR IF NOT
013132                                  XXX     14.             ;ERROR ON READ
                                ;  Switch buffers and start next read.
013142                          6$:
013142  042700  000377                  BIC     #377,R0         ;CLEAR GARBAGE
013146  006300                          ASL     R0              ;CVT # WORDS READ TO # BYTES READ
013150  010067  174510                  MOV     R0,INRDSZ       ;SAVE # BYTES ACQUIRED BY THIS READ
013154  066767  000000G 000000G        ADD     BLKFC,IFPNT     ;ADVANCE FILE BLOCK #
013162  016704  174456                  MOV     IBUF2,R4        ;ADDRESS OF BUFFER WE JUST READ INTO
013166  016767  174450  174450        MOV     IBUF1,IBUF2     ;BUFFER WE WILL START NEW READ INTO
013174  010467  174442                  MOV     R4,IBUF1
013200  016705  174460                  MOV     INRDSZ,R5       ;GET # BYTES READ BY LAST READ
013204  001475                          BEQ     3$              ;BR IF END OF FILE HIT
013206  060405                          ADD     R4,R5           ;GET ADDRESS PAST END OF BUFFER
013210  010567  174444                  MOV     R5,IBFEND       ;SAVE ADDRESS OF END OF BUFFER
013214                          1$:     $READ   #FD1,IBUF2,OWCNT,IFPNT
013374  000241                  2$:     CLC                     ;SIGNAL NO END OF FILE
013376  000207                  9$:     RETURN
                                ;  End of file hit.
013400  000261                  3$:     SEC                     ;SIGNAL END OF FILE
013402  000775                          BR      9$
```

```
                                ;-----------------------------------------------------------------------------
                                ;  NEXTIN is called when the end of an input file is reached to try to
                                ;  switch to the next input file.
                                ;  If there is another input file, it is opened and the C-flag is cleared
                                ;  on return.  R4 = Address of buffer start, R5 = Address of buffer end.
                                ;  If there are no more input files the C-flag is set on return.
                                ;
013404                          NEXTIN: PUSH    <R1,R2>
                                ;  Close last input file.
013410                                  $KEEP   #FD1
                                ;  See if there is another input file.
013456  016702  000000G                 MOV     CURSAV,R2       ;R2 -> TABLE OF -> SPC BLOCKS
013462  012201                          MOV     (R2)+,R1        ;R1 -> NEXT FILE SPEC FOR INPUT
013464  001546                          BEQ     9$              ;EXIT IF NULL POINTER
013466  010267  000000G                 MOV     R2,CURSAV       ;SAVE -> TABLE ENTRY -> NEXT INPUT FILE SPEC
013472  011102                          MOV     (R1),R2         ;R2 -> REAL FILE SPECIFICATION
013474  005712                          TST     (R2)            ;CHECK 1ST WORD OF FILE NAME
013476  001541                          BEQ     9$              ;EXIT ON BLANK NAME
                                ;
                                ;  There is another input file.
                                ;  Open it.
                                ;
013500                                  $REOPEN #FD1,R1         ;REOPEN NEXT FILE
                                ;
                                ;  Start reads from new file.
                                ;
013564  005067  000000G                 CLR     IFPNT           ;START WITH BLOCK 0 OF FILE
013570  105067  174104          1$:     CLRB    EOFFLG          ;NO EOF YET
013574                                  $READ   #FD1,IBUF2,OWCNT,IFPNT
013754  004767  177030                  CALL    RIBUF           ;START NEXT READ-AHEAD AND GET BUFFER POINTERS
013760  010467  173676                  MOV     R4,IRPOS        ;POINTER TO START OF BUFFER
013764  010567  173670                  MOV     R5,IBFEND       ;POINTER TO END OF BUFFER
013770  016767  000000G 000000G         MOV     SKPVAL,SKPCNT   ;RESET RECORD-SKIP COUNTER
                                ;  Return with C-flag cleared.
013776  000241                          CLC                     ;SIGNAL ANOTHER FILE AVAILABLE
014000  000404                          BR      10$
                                ;
                                ;  No more input files.  Return with C-flag set.
                                ;
014002  162767  000002  000000G 9$:     SUB     #2,CURSAV       ;BACKUP FILE POINTER
014010  000261                          SEC                     ;SIGNAL NO MORE FILES AVAILABLE
014012                          10$:    POP     <R2,R1>
014016  000207                          RETURN
```

```
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  SELECT is called to determine if the current record should be
                                        ;  included in the sort.
                                        ;  When called, R1 must point to the start of the buffer containing
                                        ;  the record.
                                        ;  On return, the C-flag is set if the record is to be rejected.
                                        ;
014020  010167  173650          SELECT: MOV     R1,SELREC       ;REMEMBER ADDRESS OF BUFFER
014024  010246                          MOV     R2,-(SP)
014026  010346                          MOV     R3,-(SP)
014030  010446                          MOV     R4,-(SP)
014032  010546                          MOV     R5,-(SP)
014034  016705  000000G               MOV     SELADR,R5       ;POINT TO 1ST SELECT CONTROL BLOCK
014040  012767  007707' 173630        MOV     #LSTKTP,LSTACK  ;INIT LOGIC STACK
                                        ;
                                        ;  Interpret operation for next select control block.
                                        ;
014046  116500  000000G         SLINTP: MOVB    S$OPR(R5),R0    ;GET NEXT OPERATION FUNCTION CODE
014052  020027  000000G                 CMP     R0,#SO$CMP      ;FIELD COMPARISION?
014056  001434                          BEQ     SLCMP           ;BR IF YES
014060  020027  000000G                 CMP     R0,#SO$OR       ;LOGICAL "OR"?
014064  001422                          BEQ     SLOR            ;BR IF YES
014066  020027  000000G                 CMP     R0,#SO$AND      ;LOGICAL AND?
014072  001407                          BEQ     SLAND           ;BR IF YES
                                        ;
                                        ;  Perform logical NOT function
                                        ;
014074  105177  173576          SLNOT:  COMB    @LSTACK         ;COMPLEMENT LAST LOGIC VALUE
014100  142777  000376  173570        BICB    #376,@LSTACK    ;CLEAR ALL BUT LOW ORDER BIT
014106  000167  000212          SLJNXO: JMP     SLNXOP
                                        ;
                                        ;  Perform logical AND operation.
                                        ;
014112  117700  173560          SLAND:  MOVB    @LSTACK,R0      ;GET LOGICAL VALUE ON TOP OF STACK
014116  005267  173554                  INC     LSTACK          ;POP STACK
014122  005100                          COM     R0              ;PERFORM LOGICAL AND OPERATION WITH
014124  140077  173546                  BICB    R0,@LSTACK      ;LOGICAL VALUE NEXT LOWER ON THE STACK
014130  000766                          BR      SLJNXO          ;GO PROCESS NEXT CONTROL BLOCK
                                        ;
                                        ;  Perform logical OR operation.
                                        ;
014132  117700  173540          SLOR:   MOVB    @LSTACK,R0      ;GET LOGICAL VALUE ON TOP OF STACK
014136  005267  173534                  INC     LSTACK          ;POP STACK
014142  150077  173530                  BISB    R0,@LSTACK      ;OR VALUE WITH THAT NEXT LOWER ON STACK
014146  000757                          BR      SLJNXO
                                        ;
                                        ;  Perform field comparison.
                                        ;
014150  016701  173520          SLCMP:  MOV     SELREC,R1       ;POINT TO BUFFER WITH RECORD
014154  066501  000000G                 ADD     S$POS1(R5),R1   ;POINT TO FIELD WITHIN RECORD
014160  016502  000000G                 MOV     S$POS2(R5),R2   ;POINT TO 2ND FIELD
014164  132765  000000G 000000G        BITB    #SF$LIT,S$FLG(R5);IS 2ND FIELD A LITERAL OR ANOTHER FIELD?
014172  001002                          BNE     1$              ;BR IF IT IS A LITERAL
014174  066702  173474                  ADD     SELREC,R2       ;GET ADDRESS OF 2ND FIELD WITHIN RECORD
014200  016503  000000G          1$:    MOV     S$SIZ1(R5),R3   ;GET SIZE OF FIELD
014204  116500  000000G                 MOVB    S$TYP(R5),R0    ;GET DATA TYPE CODE
```

```
     58 014210  004770  014370'          6$:     CALL    @SELVEC(R0)     ;DO FIELD COMPARISON
     59 014214  001407                           BEQ     2$              ;BR IF FIELD ARE EQUAL
     60 014216  003003                           BGT     3$              ;BR IF 1ST FIELD > 2ND
     61                                  ;  Set result code for 1st<2nd
     62 014220  012703  000000G                  MOV     #BT$LO,R3       ;SET LOW CODE
     63 014224  000404                           BR      4$
     64                                  ;  Set result code for 1st>2nd
     65 014226  012703  000000G          3$:     MOV     #BT$HI,R3       ;SET HIGH CODE
     66 014232  000401                           BR      4$
     67                                  ;  Set result code for 1st=2nd
     68 014234  005003                   2$:     CLR     R3              ;NEITHER > OR <
     69                                  ;  Now compare result of comparison with type of result that means TRUE
     70 014236  116502  000000G          4$:     MOVB    S$BR(R5),R2     ;GET TRUE TYPE CODE
     71 014242  030203                           BIT     R2,R3           ;COMPARE WITH ACTUAL RESULT OF COMPARISON
     72 014244  001404                           BEQ     5$              ;BR IF THEY DON'T MATCH (NORMALLY FALSE)
     73 014246  032702  000000G                  BIT     #BT$REV,R2      ;SHOULD WE REVERSE LOGIC VALUE?
     74 014252  001414                           BEQ     SLTRUE          ;BR IF NOT (RESULT IS TRUE)
     75 014254  000403                           BR      SLFALS          ;RESULT IS FALSE
     76 014256  032702  000000G          5$:     BIT     #BT$REV,R2      ;SHOULD WE REVERSE LOGIC VALUE?
     77 014262  001010                           BNE     SLTRUE          ;BR IF YES (RESULT IS TRUE)
     78                                  ;  Result of comparison is false.
     79 014264  105767  000000G          SLFALS: TSTB    ORFLG           ;ARE THERE ANY "OR" OPERATIONS IN EXPRESSION?
     80 014270  001425                           BEQ     SLREJT          ;IF NOT THEN REJECT RECORD IMMEDIATELY
     81 014272  005367  173400                   DEC     LSTACK          ;PUSH "FALSE" VALUE ON LOGIC STACK
     82 014276  105077  173374                   CLRB    @LSTACK         ;0==>FALSE
     83 014302  000410                           BR      SLNXOP
     84                                  ;  Result of comparison is true.
     85 014304  105767  000000G          SLTRUE: TSTB    ANDFLG          ;ARE THERE ANY AND OPERATIONS IN EXPRESSION?
     86 014310  001417                           BEQ     SLACPT          ;IF NOT THEN ACCEPT RECORD IMMEDIATELY
     87 014312  005367  173360                   DEC     LSTACK          ;PUSH "TRUE" VALUE ON STACK
     88 014316  112777  000001  173352          MOVB    #1,@LSTACK      ;1==>TRUE
     89                                  ;
     90                                  ;  Move on to next select control block.
     91                                  ;
     92 014324  016505  000000G          SLNXOP: MOV     S$LINK(R5),R5   ;CHAIN FORWARD
     93 014330  001402                           BEQ     1$              ;BR IF HIT END OF EXPRESSION
     94 014332  000167  177510                   JMP     SLINTP          ;GO INTERPRET OPERATION CODE
     95                                  ;
     96                                  ;  We have finished last logical operation.
     97                                  ;  Result on top of stack represents final value of expression.
     98                                  ;
     99 014336  105777  173334          1$:     TSTB    @LSTACK         ;IS RESULT TRUE OR FALSE?
    100 014342  001002                           BNE     SLACPT          ;TRUE -- ACCEPT RECORD
    101                                 ;  Reject record.
    102 014344  000261                  SLREJT: SEC                     ;SIGNAL RECORD REJECTION
    103 014346  000401                          BR      SLRTN
    104                                 ;  Accept record.
    105 014350  000241                  SLACPT: CLC                     ;SIGNAL RECORD ACCEPTANCE
    106 014352  012605                  SLRTN:  MOV     (SP)+,R5
    107 014354  012604                          MOV     (SP)+,R4
    108 014356  012603                          MOV     (SP)+,R3
    109 014360  012602                          MOV     (SP)+,R2
    110 014362  016701  173306                  MOV     SELREC,R1
    111 014366  000207                          RETURN
    112                                 ;
    113                                 ;  Jump vector used by SELECT to call proper field comparison routine.
    114                                 ;
```

```
    115 014370  014412'                SELVEC:  .WORD   KYASCI          ;FT$C -- ASCII CHARACTERS
    116 014372  000000G                         .WORD   KYCOMP          ;FT$B -- SIGNED BINARY INTEGER
    117 014374  000000                          .WORD   0               ;FT$U -- UNSIGNED BINARY INTEGER
    118 014376  000000G                         .WORD   KYDATE          ;FT$T -- DATE (MMDDYY)
    119 014400  000000G                         .WORD   KYBIN           ;FT$F -- FLOATING
    120 014402  000000G                         .WORD   KYDIS           ;FT$D -- DIBOL INTEGER
    121 014404  000000G                         .WORD   KYUCA           ;FT$A -- UPPER CASE ASCII CHARACTERS
    122 014406  000000G                         .WORD   KYLSS           ;FT$I -- LEADING SEPARATE SIGN
    123 014410  000000G                         .WORD   KYTSS           ;FT$J -- TRAILING SEPARATE SIGN
```

```
                                ;
                                ;*******************************************************************************
                                ;  KYASCI is called to compare two ascii character strings.
                                ;  When called, R1 must point to the 1st string and R2 must point
                                ;  to the 2nd string.  R3 must contain the number of characters.
                                ;  On return, the conditon code is set for BHI/BLO/BEQ test.
                                ;
014412  122122                  KYASCI: CMPB    (R1)+,(R2)+     ;COMPARE CHARACTERS
014414  001002                          BNE     1$              ;BR IF NOT EQUAL
014416  005303                          DEC     R3              ;MORE TO CHECK?
014420  003374                          BGT     KYASCI          ;BR IF YES
014422  000207                  1$:     RETURN                  ;RETURN WITH CONDITION CODE SET
```

```
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;  DOMAP is called to perform the field mapping function.
                                        ;  When called, MAPBUF must point to the start of the buffer containing
                                        ;  the input source record and RECPTR must point to the buffer where
                                        ;  the destination record is to be stored.  The length of the resulting
                                        ;  record must be stored in RECLEN.
                                        ;
014424  010146                  DOMAP:  MOV     R1,-(SP)
014426  010246                          MOV     R2,-(SP)
014430  010346                          MOV     R3,-(SP)
014432  010446                          MOV     R4,-(SP)
                                        ;
                                        ;  Blank or null fill the result buffer.
                                        ;
014434  012700  000040                  MOV     #' ,R0          ;ASSUME SPACE FILL WANTED
014440  032767  000000G 000000G         BIT     #RT$RB,ITYPE    ;IS FILE RECORDING MODE BINARY?
014446  001401                          BEQ     1$              ;BR IF ASCII
014450  105000                          CLRB    R0              ;NULL FILL
014452  016703  000000G           1$:   MOV     RECLEN,R3       ;GET LENGTH OF RESULT BUFFE
014456  016702  173210                  MOV     RECPTR,R2       ;POINT TO RESULT BUFFER
014462  110022                  2$:     MOVB    R0,(R2)+        ;FILL WITH SPACES OR NULLS
014464  005303                          DEC     R3              ;COUNT # BYTES MOVED
014466  001375                          BNE     2$              ;BR IF MORE TO DO
                                        ;
                                        ;  Now move source fields to destination record.
                                        ;
014470  016704  000000G                 MOV     FLDMAP,R4       ;POINT TO 1ST MAP CONTROL BLOCK
                                        ;  Move next field.
014474  016401  000000G         MAPMOV: MOV     M$POS1(R4),R1   ;GET POSITION OF SOURCE FIELD
014500  066701  173164                  ADD     MAPBUF,R1       ;GET ABSOLUTE ADDRESS
014504  016402  000000G                 MOV     M$POS2(R4),R2   ;GET POSITION OF DESTINATION FIELD
014510  066702  173156                  ADD     RECPTR,R2       ;GET ABSOLUTE ADDRESS
014514  016403  000000G                 MOV     M$SIZE(R4),R3   ;GET # BYTES TO MOVE
014520  112122                  4$:     MOVB    (R1)+,(R2)+     ;MOVE THE FIELD
014522  005303                          DEC     R3              ;MORE TO MOVE?
014524  003375                          BGT     4$              ;BR IF YES
                                        ;  Move on to next map field descriptor.
014526  162704  000000G         5$:     SUB     #M$$SZ,R4       ;POINT TO NEXT DESCRIPTOR
014532  020467  000000G                 CMP     R4,MAPEND       ;DONE ALL?
014536  103356                          BHIS    MAPMOV          ;BR IF MORE TO DO
                                        ;
                                        ;  Mapping is finished
                                        ;
014540  012604                  MAPFIN: MOV     (SP)+,R4
014542  012603                          MOV     (SP)+,R3
014544  012602                          MOV     (SP)+,R2
014546  012601                          MOV     (SP)+,R1
014550  000207                          RETURN
```

```
                                ;*******************************************************************************
                                ;  INCNUM is called to increment the BCD record number counter.
                                ;
014552  005767  000000G  INCNUM: TST     NUMLEN          ;IS RECORD NUMBERING WANTED?
014556  001415                   BEQ     9$              ;GET OUT FAST IF NOT
014560  010146                   MOV     R1,-(SP)
014562  012701  000010G          MOV     #NUMBUF+8.,R1   ;POINT PAST END OF NUMBER
014566  114100           2$:     MOVB    -(R1),R0        ;GET A DIGIT
014570  005200                   INC     R0              ;INCREMENT IT
014572  120027  000072          CMPB    R0,#72          ;DID WE JUST INCREMENT PAST 9?
014576  103403                   BLO     1$              ;BR IF NOT
014600  112711  000060          MOVB    #'0,(R1)        ;REPLACE DIGIT WITH ZERO
014604  000770                   BR      2$              ;AND GO INCREMENT NEXT DIGIT TO LEFT
014606  110011           1$:     MOVB    R0,(R1)         ;INSERT INCREMENTED DIGIT
014610  012601                   MOV     (SP)+,R1
014612  000207           9$:     RETURN

                                ;*******************************************************************************
                                ;  INSNUM is called to insert the BCD record number into the current record.
                                ;  Inputs:
                                ;    RECPTR = Pointer to start of current record.
                                ;    NUMCOL = Column (byte) number where number insert is to begin.
                                ;    NUMLEN = Number of columns to be used by number.
                                ;    NUMBUF = Buffer where 8 digit BCD number is stored.
                                ;
014614  010146  INSNUM: MOV     R1,-(SP)
014616  010246          MOV     R2,-(SP)
014620  010346          MOV     R3,-(SP)
014622  016701  173044  MOV     RECPTR,R1       ;ADDRESS OF START OF RECORD
014626  066701  000000G ADD     NUMCOL,R1       ;ADDRESS OF START OF NUMBER INSERT AREA
014632  016703  000000G MOV     NUMLEN,R3       ;LENGTH OF NUMBER FIELD
014636  060301          ADD     R3,R1           ;ADDRESS PAST END OF NUMBER INSERT AREA
014640  012702  000010G MOV     #NUMBUF+8.,R2   ;POINT PAST END OF NUMBER COUNTER FIELD
014644  010300          MOV     R3,R0           ;GET DESIRED NUMBER LENGTH
014646  020027  000010  CMP     R0,#8.          ;ONLY GOT 8 DIGITS IN BUFFER
014652  101402          BLOS    1$              ;BR IF OK
014654  012700  000010  MOV     #8.,R0          ;ONLY MOVE 8 DIGITS
014660  160003  1$:     SUB     R0,R3           ;# FILLER ZEROES NEEDED
014662  114241  2$:     MOVB    -(R2),-(R1)     ;MOVE RECORD # TO RECORD
014664  005300          DEC     R0
014666  003375          BGT     2$
014670  005703          TST     R3              ;DO WE NEED TO INSERT EXTRA ZEROS
014672  001405          BEQ     4$              ;BR IF NOT
014674  112700  000060  MOVB    #'0,R0          ;GET ZERO CHARACTER
014700  110041  3$:     MOVB    R0,-(R1)        ;INSERT FILLER ZEROES
014702  005303          DEC     R3
014704  003375          BGT     3$
014706  012603  4$:     MOV     (SP)+,R3
014710  012602          MOV     (SP)+,R2
014712  012601          MOV     (SP)+,R1
014714  000207          RETURN
```

```
                                        ;-------------------------------------------------------------------------------
                                        ;  STRIP is called to strip trailing spaces off the ends of ascii records.
                                        ;  Inputs:
                                        ;    RECPTR = Address of start of current record.
                                        ;    RECLEN = Length of current record.
                                        ;
014716  010146                  STRIP:  MOV     R1,-(SP)
014720  010246                          MOV     R2,-(SP)
014722  016701  172744                  MOV     RECPTR,R1       ;GET ADDRESS OF FRONT OF RECORD
014726  016702  000000G                 MOV     RECLEN,R2       ;GET LENGTH OF RECORD
014732  060201                          ADD     R2,R1           ;GET ADDRESS OF END OF RECORD
014734  112700  000040                  MOVB    #' ,R0          ;GET SPACE CHAR FOR COMPARISON
014740  124100                  1$:     CMPB    -(R1),R0        ;IS NEXT CHAR A SPACE?
014742  001003                          BNE     2$              ;QUIT IF NOT
014744  105011                          CLRB    (R1)            ;REPLACE WITH A NULL
014746  005302                          DEC     R2              ;MORE TO CHECK?
014750  003373                          BGT     1$              ;BR IF YES
014752  012602                  2$:     MOV     (SP)+,R2
014754  012601                          MOV     (SP)+,R1
014756  000207                          RETURN
```

```
                                ;
                                ;*******************************************************************************
                                ;  EOFSAV is called to save a user EOF record in a holding buffer
                                ;  so that it can be written to the end of the output file.
                                ;  When called R1 must contain the address of the buffer containing
                                ;  the EOF record.
                                ;
014760  010146                  EOFSAV: MOV     R1,-(SP)
014762  010346                          MOV     R3,-(SP)
014764  010446                          MOV     R4,-(SP)
014766  016703  000000G                 MOV     RECLEN,R3       ;GET LENGTH OF RECORD
014772  016704  000000G                 MOV     EOFBUF,R4       ;GET ADDRESS OF EOF HOLDING BUFFER
014776  112124                  1$:     MOVB    (R1)+,(R4)+     ;MOVE RECORD TO HOLDING BUFFER
015000  001004                          BNE     2$              ;BR IF NON-NULL CHARACTER
015002  032767  000000G 000000G         BIT     #RT$VLN,ITYPE   ;IS THIS A VARIABLE LENGTH RECORD?
015010  001002                          BNE     9$              ;BR IF YES -- WE JUST HIT END OF IT
015012  005303                  2$:     DEC     R3              ;NEED TO MOVE MORE?
015014  003370                          BGT     1$              ;BR IF YES
015016  012604                  9$:     MOV     (SP)+,R4
015020  012603                          MOV     (SP)+,R3
015022  012601                          MOV     (SP)+,R1
015024  000207                          RETURN
```

```
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;
                                        ;  PASS1 TERMINATION ROUTINE.
                                        ;
015026  016703  000000G       S1STOP:  MOV     NXTBLK,R3          ;GET NXT BLOCK #.
015032  062703  000002                 ADD     #2,R3              ;LEAVE A LITTLE EXTRA ROOM.
015036  010367  000000G                MOV     R3,NXTBLK
015042  004767  000000G                CALL    OUTWRT             ;WRITE TO LAST BLK SO MONITOR WILL
015046  000167  000000G                JMP     P1TERM             ;RESERVE ROOM IN FILE.

                                        ;
015054'                       P1TOP    =       .+2
015054                        P1SIZ    =       P1TOP-P1BASE
000001                                 .END
```

Symbol table

```
ABORT = ****** G        ERREAD  011406R  002  MAXREC= 000620          RICA    012070R  002  SLFALS  014264R  002
ANDFLG= ****** G        HB$ARS= 000006       MXFILS= ****** G        RICAR   012220R  002  SLINTP  014046R  002
BELL  = 000007          HB$DFS= 000002       MXFIL2= ****** G        RICB    012500R  002  SLJNXO  014106R  002
BIGEST= 177777          HB$DIP= 000030       MXREC1= ****** G        RICX    012630R  002  SLNOT   014074R  002
BLKFC = ****** G        HB$DUP= 000001       MXREC2= ****** G        RIEND   012452R  002  SLNXOP  014324R  002
BT$HI = ****** G        HB$RIU= 000032       M$POS1= ****** G        RIEOF   012474R  002  SLOR    014132R  002
BT$LO = ****** G        HB$RSZ= 000004       M$POS2= ****** G        RIFIN   012454R  002  SLREJT  014344R  002
BT$REV= ****** G        HD$DEV= 000000       M$SIZE= ****** G        RIFL    012016R  002  SLRTN   014352R  002
BUFSIZ= 001000          HD$$SZ= 000004       M$$SZ = ****** G        RIRS    012330R  002  SLTRUE  014304R  002
CHNBIT= 000400          IBFEND  007660R  002  NEXTIN  013404R   002  RMAX    007652R  002  SORTP1  007710RG 002
CHNO  = 000000          IBUF1   007642R  002  NOERR = 000001        RMON  = 000054          SO$AND= ****** G
CISRT1= ****** G        IBUF2   007644R  002  NUMBUF= ****** G      RN      006200R  002  SO$CMP= ****** G
COMPAR= ****** G        IFMT    010176R  002  NUMCOL= ****** G      RQ      007646R  002  SO$NOT= ****** G
CR    = 000015          IFPNT = ****** G      NUMKY2= ****** G      RSBGCW  012512R  002  SO$OR = ****** G
CSPCOF= 000376          INCNUM  014552R  002  NUMLEN= ****** G      RSBMOV  012550R  002  SRTCNT= ****** G
CTRLZ = 000032          INFLEN= ****** G      NUMREC  007640R   002  RSTSJB= 000404          SR2     010532R  002
CUROFL= ****** G        INRDSZ  007664R  002  NXTBLK= ****** G      RT$CBL= ****** G        SR3     010726R  002
CURSAV= ****** G        INRLEN= ****** G      OBFEND= ****** G      RT$CR = ****** G        SR4     011110R  002
DBLKFC= 000005          INSNUM  014614R  002  OBUFSZ= ****** G      RT$DBL= ****** G        SR5     011206R  002
DOMAP   014424R  002    IRPOS   007662R  002  OBUF1 = ****** G      RT$FTN= ****** G        SR6     011216R  002
ENDINI  010720R  002    ISMARS= ****** G      OBUF2 = ****** G      RT$F11= ****** G        SR7     011316R  002
EOF     011174R  002    ITYPE = ****** G      OFLINK  007666R   002  RT$OR = ****** G        STRIP   014716R  002
EOFBUF= ****** G        JSW   = 000044        ORFLG = ****** G      RT$OS = ****** G        S$BR  = ****** G
EOFCHR= ****** G        KSTRT = ****** G      ORPOS = ****** G      RT$OX = ****** G        S$FLG = ****** G
EOFCOL= ****** G        KTYPE = ****** G      OUTWR1= ****** G      RT$RA = ****** G        S$LINK= ****** G
EOFFLG  007700R  002    KYASCI  014412R  002  OVLBIT= 001000        RT$RB = ****** G        S$OPR = ****** G
EOFSAV  014760R  002    KYBIN = ****** G      OWCNT = ****** G      RT$TXT= ****** G        S$POS1= ****** G
ERR   = 000006          KYCOMP= ****** G      PFSPC = 000002        RT$VLN= ****** G        S$POS2= ****** G
ERRLOC= 000052          KYDATE= ****** G      P1BASE= 000000RG   002  RT11  = 000000          S$SIZ1= ****** G
FD1   = ****** G        KYDIS = ****** G      P1SIZ = 015054 G      RUNEND  010552R  002  S$TYP = ****** G
FE      003100R  002    KYLSS = ****** G      P1TERM= ****** G      RWDLEN= ****** G        S1STOP  015026R  002
FI      004540R  002    KYTSS = ****** G      P1TOP = 015054RG   002  SELADR= ****** G        TMPCNT= ****** G
FILFL = ****** G        KYUCA = ****** G      P2TOP = ****** G      SELECT  014020R  002  TOPMEM= ****** G
FILNB = ****** G        LF    = 000012        Q       007654R   002  SELREC  007674R  002  UEOF  = ****** G
FILRN = ****** G        LOC     000000R  002  QLEN  = 000003        SELVEC  014370R  002  USERRB= 000053
FILST = ****** G        LOSER   001440R  002  RC      007650R   002  SETNRT= ****** G        USRLOC= 000046
FLBLK = ****** G        LOWMEM= ****** G      RCNT1 = ****** G      SEVERR= 000010          VERSNO= 000276
FLDMAP= ****** G        LSTACK  007676R  002  RCNT2 = ****** G      SF$LIT= ****** G        WRDS  = 000004
FLUSH = ****** G        LSTKEY  007656R  002  RCXE    012666R   002  SKPCNT= ****** G        XEMT  = ****** G
FT$D  = ****** G        LSTKTP  007707R  002  RDFILE  012772R   002  SKPVAL= ****** G        XXX14.= 013132R  002
FT$T  = ****** G        MAPBUF  007670R  002  READIN  011666R   002  SKPXRC  012740R  002  XXX24.= 010114R  002
GETREC  011356R  002    MAPEND= ****** G      RECLEN= ****** G      SLACPT  014350R  002  XXX33.= 012562R  002
GRCKEF  011500R  002    MAPFIN  014540R  002  RECPTR  007672R   002  SLAND   014112R  002  ...V1 = 000003
GREOF   011642R  002    MAPMOV  014474R  002  RIBUF   013010R   002  SLCMP   014150R  002  ...V2 = 000061
```

```
. ABS.   000000    000   (RW,I,GBL,ABS,OVR)
         000000    001   (RW,I,LCL,REL,CON)
SORT1    015052    002   (RW,I,GBL,REL,OVR)
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  15783 Words  ( 62 Pages)
```

Symbol table

Size of core pool: 10432 Words (72 Pages)
Operating system: RT-11

Elapsed time: 00:00:23.38
,DB4:CBSRTH,DK:CBSRTH.CRF=DK:CBSRTH,CBSRT1/C

Cross reference table (CREF V05.03)

```
...V1    4-89     4-89    16-6    16-6    16-24   16-24   17-10   17-10   17-23   17-23   17-29   17-29
...V2    4-89     4-89     4-89#   4-89#  16-6    16-6#   16-24   16-24   16-24#  16-24#  17-10   17-10#  17-23   17-23
        17-23#  17-23#  17-29   17-29   17-29#  17-29#
ABORT    2-45     4-46     8-28   10-18   12-30   16-10
ANDFLG   2-54    18-85
BELL     1-40#
BIGEST   1-36#
BLKFC    2-36    16-16
BT$HI    2-52    18-65
BT$LO    2-51    18-62
BT$REV   2-52    18-73   18-76
BUFSIZ   1-34#
CHNBIT   1-53#
CHNO     1-87#    4-89     4-89    16-6    16-6    16-24   16-24   17-10   17-10   17-23   17-23   17-29   17-29
CISRT1   2-54     4-5
COMPAR   2-46     5-92    5-120
CR       1-38#    3-11#
CSPCOF   1-65#
CTRLZ    1-41#    3-13#    6-41
CUROFL   2-48     4-72*    5-13     5-33*
CURSAV   2-55    17-12   17-15*  17-40*
DBLKFC   1-35#
DOMAP    6-55    20-9#
ENDINI   5-12     5-42#
EOF      5-98     5-104#
EOFBUF   2-37    23-12
EOFCHR   2-45     6-60
EOFCOL   2-45     6-57
EOFFLG   3-31#    4-83*    6-12    6-62*   6-117*  7-8     17-28*
EOFSAV   6-54    23-8#
ERR      1-70#    4-89*    4-89*   4-89*   16-6    16-6*   16-6*   16-6*   16-24*  16-24*  16-24*  17-23*  17-29*  17-29*
        17-29*
ERRLOC   1-49#    4-89    16-6    16-24   17-23   17-29
FD1      2-51     4-89    16-6    16-24   17-10   17-23   17-29
FE       3-7#     4-57*    5-109
FI       3-8#     4-58*    5-132
FILFL    2-34     4-75*    5-40*
FILNB    2-35     4-68*
FILRN    2-34     4-67*    5-14*
FILST    2-34     4-66*    4-74*    5-37     5-39*
FLBLK    2-47     5-17*    5-67*
FLDMAP   2-49     4-11     6-21     6-83    20-28
FLUSH    2-46     5-24     5-68
FT$D     2-52
FT$T     2-48
GETREC   5-57     6-12#    6-112
GRCKEF   4-53     6-57#
GREOF    6-30     6-35     6-42     6-65     6-108#
GRREAD   6-22     6-27#    6-55     6-74
HB$ARS   1-74#
HB$DFS   1-72#
HB$DIP   1-75#
HB$DUP   1-71#
HB$RIU   1-76#
HB$RSZ   1-73#
HD$$SZ   1-51#
```

Cross reference table (CREF V05.03)

```
HD$DEV    3-20#
IBFEND    3-22#    4-20*    7-19    16-23*   17-32*
IBUF1     3-15#    4-18*   16-18    16-19*
IBUF2     3-16#    4-22*    4-89    16-17    16-18*   16-24    17-29
IFMT      4-25*
IFPNT     2-43     4-85*    4-88*    4-89    16-16*   16-24    17-27*   17-29
INCNUM    7-44     8-4      9-4     10-13    12-10    13-34    21-4#
INFLEN    2-36
INRDSZ    3-24#   15-15*   16-20
INRLEN    2-49     4-14     7-17    13-19    13-30
INSNUM    5-92    21-26#
IRPOS     3-23#    4-19*    7-18    11-5*    17-31*
ISMARS    2-47    13-14    13-29
ITYPE     2-40     4-86     5-77     6-39     6-96     7-23     8-11     10-30    20-17    23-15
JSW       1-51#    4-9
KSTRT     2-48
KTYPE     2-31
KYASCI   18-115   19-8#    19-11
KYBIN     2-40    18-119
KYCOMP    2-52    18-116
KYDATE    2-39    18-118
KYDIS     2-52    18-120
KYLSS     2-39    18-122
KYTSS     2-39    18-123
KYUCA     2-53    18-121
LF        1-39#    3-12#
LOC       3-3#     4-34     5-51     5-52*    5-86     5-117    5-119
LOSER     3-4#     4-60*    5-116    5-127    5-128*
LOWMEM    2-47
LSTACK    3-29#   18-15*   18-29*   18-30*   18-35    18-36*   18-38*   18-43    18-44*   18-45*   18-81*   18-82*   18-87*   18-88*
          18-99
LSTKEY    3-21#    4-29*    5-52     5-53*    5-91
LSTKTP    3-34#   18-15
M$$SZ     2-33    20-39
M$POS1    2-53    20-30
M$POS2    2-53    20-32
M$SIZE    2-53    20-34
MAPBUF    3-26#    4-13*    6-23    20-31
MAPEND    2-49    20-40
MAPFIN   20-45#
MAPMOV   20-30#   20-41
MAXREC    3-2#     3-5      3-6      3-7      3-8      3-9      4-39
MXFIL2    2-55     4-70     5-30
MXFILS    2-35
MXREC1    2-38     6-27
MXREC2    2-38     6-29
NEXTIN    6-103   15-16    17-8#
NOERR     1-56#
NUMBUF    2-44    21-7     21-33
NUMCOL    2-44    21-30
NUMKY2    2-34
NUMLEN    2-44     6-90    21-4     21-31
NUMREC    3-14#    4-43*
NXTBLK    2-46     4-76*    5-38     5-41*    5-67     5-70*    24-6     24-8*
OBFEND    2-38     4-78*    4-79*    5-55     5-73
OBUF1     2-37     4-24*    4-77     5-15     5-61     5-71
```

Cross reference table (CREF V05.03)

```
OBUF2     9-37     4-26*
OBUFSZ    2-37     4-21     4-23     4-25     4-27     4-77
OFLINK    3-25#    4-73*    5-17     5-37*
ORFLG     2-34    10-79
ORPOS     2-46     4-82*    5-18     5-54     5-81*
OUTWRT    2-48    24-9
OVLBIT    1-52#    4-9
OWCNT     2-36     4-89     4-89    16-24    16-24    17-29    17-29
P1BASE    2-4#     2-27    24-14
P1SIZ     2-27    24-14#
P1TERM    2-47    24-10
P1TOP     2-27     4-9     24-13#   24-14
P2TOP     4-9      4-9      4-9
PFSPC     1-28#   17-10*   17-23*
Q         3-20#    4-93*    5-50     5-85     5-108    5-118    5-126    5-127*
QLEN      1-48#
RC        3-13#    4-90*    5-5      5-11     5-42*    5-89
RCNT1     2-45     6-27     6-46*
RCNT2     2-45     6-29     6-48*
RCXE     13-12    13-19#
RDFILE   13-6     13-23    14-19    15-11#
READIN    6-34     7-8#
RECLEN    2-32     5-57    20-20    22-10    23-11
RECPTR    3-27#    6-16*    6-86     6-111    20-21    20-33    21-29    22-9
RIBUF     7-58     8-7      9-7      9-18     9-26     9-34    10-10    10-22    12-7     12-19    12-34    15-11    16-6#    17-30
RICA      7-34     7-36     8-4#
RICAR     7-33     9-4#     9-22
RICB      7-31    12-5#
RICX      7-23    13-4#
RIEND     7-57     8-32     8-37     9-37    10-35    10-39    11-4#    12-41    12-45    13-36
RIEOF     7-60     8-8      8-12     8-16     9-8      9-19     9-27     9-35    10-11    10-23    11-15#   12-8     12-13    12-20
         12-35    13-7     13-16    13-24
RIFIN    11-5#    11-16
RIFL      7-38     7-44#
RIRS      7-40    10-5#
RMAX      3-17#    4-92*    5-34     5-97     5-99*    5-104    5-135
RMON      1-57#
RN        3-7#     4-59*    5-113    5-124    5-125*
RQ        3-17#    4-91*    5-6      5-34     5-42     5-47     5-94     5-96*    5-104*   5-105*   5-111    5-123    5-124*   5-139*
RSB@CW   12-6     12-10#   12-23
RSBMOV   12-12    12-26#
RSTSJB    1-44#
RT$CBL    2-41     7-24     8-11
RT$CR     2-43     7-35
RT$DBL    2-41     6-39
RT$F11    2-42     7-37
RT$FTN    2-41
RT$OR     2-42     7-32
RT$OS     2-42     7-39
RT$OX     2-42     4-86     7-26
RT$RA     2-43
RT$RB     2-43     7-29    10-30    20-17
RT$TXT    2-41
RT$VLN    2-43     5-77     6-96    23-15
RT11      1-47#
RUNEND    5-13#    5-140
```

Cross reference table (CREF V05.03)

```
RWDLEN    2-35     4-30     4-33     4-41
S$BR      3-34    18-70
S$FLG     2-31    18-53
S$LINK    2-52    18-72
S$OPR     2-30    18-19
S$POS1    2-30    18-51
S$POS2    2-30    18-52
S$SIZ1    2-31    18-56
S$TYP     2-31    18-57
S1STOP    5-36    24-6#
SELADR    2-49     6-69    18-14
SELECT    6-71    18-9#
SELREC    3-28#   18-9*    18-50    18-55    18-110
SELVEC   18-58    18-115#
SETNRT    2-31
SEVERR    1-55#
SF$LIT    2-31    18-53
SKPCNT    2-36     6-52     6-54*   17-33*
SKPVAL    2-35    17-33
SKPXRC   13-15    13-32    14-14#
SLACPT   13-86    18-100   18-105#
SLAND    13-25    13-35#
SLCMP    15-21    18-50#
SLFALS   13-75    13-79#
SLINTP   18-19#   18-94
SLJNXO   13-31#   18-39    18-46
SLNOT    13-59#
SLNXOP   15-31    13-83    18-92#
SLOR     15-53    18-43#
SLREJT   13-60    18-102#
SLRTN    13-103   13-106#
SLTRUE   13-74    13-77    18-85#
SO$AND    2-32    18-24
SO$CMP    2-30    18-20
SO$NOT    2-32
SO$OR     2-30    18-22
SORTP1    2-27     4-5#
SR2       5-3#     5-137
SR3       5-7      5-47#
SR4       5-43     5-85#
SR5       5-93     5-98     5-100    5-108#
SR6       5-111#   5-133
SR7       5-112    5-115    5-121    5-130#
SRTCNT    2-40     6-78*    6-79*
STRIP     6-98    22-7#
TMPCNT    2-40     5-22*    5-23*    5-25*    5-65*    5-66*    5-69*
TOPMEM    2-32     4-32
UEOF      2-33     6-63*    6-73*
USERRB    1-50#
USRLOC    1-54#
VERSNO    1-58#
WRDS      1-89#    4-89*    4-89*   16-6     16-24*   16-24*   17-29*   17-29*
XEMT      4-89     4-89     4-89    16-24    16-24    16-24    17-23    17-29    17-29    17-29
XXX14.   16-10#
XXX24.    4-46#
XXX33.    3-28#   10-18#   12-30#
```

Cross reference table (CREF V05.03)

```
$ERASE    1-638#
$EXIT     1-609#
$FDB      1-145#
$GETBO    1-402#   4-9
$GETCM    1-356#
$GETTO    1-424#
$KEEP     1-448#  17-10
$LOCK     1-328#
$NEW      1-228#
$OLD      1-260#
$PRINT    1-664#
$PRT      1-696#
$PURGE    1-465#
$READ     1-492#   4-89    16-24    17-29
$REOPE    1-309#  17-23
$SAVES    1-289#
$SPC      1-159#
$SYDON    1-674#
$SYINI    1-539#
$SYSTE    1-171#
$UNLOC    1-338#
$WAIT     1-558#  16-6
$WRITE    1-525#
...CM1    4-59    16-24    17-23    17-29
...CM2    4-59     4-89     4-89     4-89    16-24    16-24    16-24    16-24    17-23    17-29    17-29    17-29    17-29
...CM3   17-10
...CM5    4-59    16-24    17-23    17-29
...CM7    4-59    16-24    17-29
.CHAIN    1-96#    1-99#
.CLOSE    1-58#   17-10
.CSISP    1-100#
.DELET    1-57#
.DSTAT    1-93#
.ENTER    1-56#
.EXIT     1-99#
.FETCH    1-93#
.GTLIN    1-101#
.GVAL     1-100#
.HERR     1-97#
.LOCK     1-96#
.LOOKU    1-56#
.PRINT    1-57#
.PURGE    1-59#
.QSET     1-98#
.READ     1-101#   4-89    16-24    17-29
.READW    1-100#
.RELEA    1-59#
.REOPE    1-100#  17-23
.SAVES    1-100#
.SERR     1-97#
.SETTO    1-97#
.TTYIN    1-97#
.TTYOU    1-98#
.UNLOC    1-56#
.WAIT     1-101#  16-6
.WRITE    1-101#
```

Cross reference table (CREF V05.03)

.WRITW	1-101#											
POP	1-118#	4-89	4-89	16-6	16-24	16-24	17-10	17-23	17-23	17-29	17-29	17-42
PUSH	1-112#	4-89	4-89	16-6	16-24	16-24	17-8	17-10	17-23	17-23	17-29	17-29
SETNRT	1-210#											
XXX	3-40#	4-46	8-28	10-18	12-30	16-10						