

Table of contents

9-	1	SORT INITIALIZATION
13-	1	PARSE SWITCHES
22-	1	Initialize files
23-	1	Generate code to do key comparisons
24-	1	Determine buffer sizes
25-	2	Set up information about file specs
27-	1	RSTS oriented subroutines
28-	1	CODE GENERATOR SUBROUTINES
29-	1	SUBROUTINES
35-	1	PRNUM -- Print a decimal number
35-	27	CVTNUM -- Convert binary value to asciz string
36-	1	EXECUTION TERMINATION

```
1 .TITLE SORTI SORT INITIALIZATION
2 .ENABL LC
3 ;
4 ; Copyright (c) 1977,1978,1979,1980.
5 ; S&H Computer Systems, Inc.
6 ; Nashville, Tennessee
7 ;
8 000000 .PSECT SORTID,D ;DATA SECTION PSECT
9 ;
10 ; Written by Phil Sherrod.
11 ;
12 ;
13 ; THIS MODULE OF SORT IS THE INTERFACE TO
14 ; THE CALLING PROGRAM OR USER.
15 ; IT ALLOWS SORT TO BE RUN IN ANY OF THREE MODES:
16 ; 1) DIRECT RUN
17 ; 2) CHAIN FROM ANOTHER PROGRAM
18 ; 3) TSX MESSAGE COMMUNICATION
19 ;
20 ;
21 ; #####
22 ;
23 ; Modified in February, 1980 by K.C. Lidster of DISC to also execute
24 ; under the RT11 run-time for the RSTS operating system. This mainly
25 ; involved the handling of the extra file parameters pertaining to
26 ; RSTS (PPN, clustersize, etc.).
27 ;
28 ; #####
29 ;
30 ;
31 ;
32 ; GLOBAL DEFINITIONS
33 ;
34 .GLOBL SRTINI, TERM, FT$D, SO$NOT
35 .GLOBL S$LINK, S$OPR, S$TYP, S$BR, S$FLG
36 .GLOBL S$POS1, S$SIZ1, S$POS2, S$$SZ, SO$CMP
37 .GLOBL SO$AND, SO$OR, FT$T, SF$LIT, M$POS1
38 .GLOBL M$POS2, M$SIZE, M$$SZ, BT$HI, BT$LO
39 .GLOBL BT$REV
40 ;
41 ; GLOBAL REFERENCES
42 ;
43 .GLOBL QBLK, MODE, TSXFL, FSTIME
44 .GLOBL KTYPE, KAD, KSTRT, KLEN
45 .GLOBL TOPMEM, NUMKY2, RECLN, SKPCNT, MXREC1
46 .GLOBL MXREC2, CINSPC, EOFCHR, EOFCOL
47 .GLOBL T1$SZ, T2$SZ, ISMARS
48 .GLOBL NUMCOL, NUMLEN, NUMBUF
49 .GLOBL ITYPE, OTYPE, RT$CBL, RT$DBL
50 .GLOBL RT$FTN, RT$TXT, RT$X, RT$F11
51 .GLOBL RT$OS, RT$OR, RT$OX
52 .GLOBL CURSAV, SKPVAL, RSPCS
53 .GLOBL RT$RA, RT$RB, F$INFL, CISRTI
54 .GLOBL RT$VLN, RT$CR, MULTPY, DIVIDE
55 .GLOBL INFLEN, KYUCA, BLKFC
56 .GLOBL QWCNT, OBUFSZ, START1, STACK
57 .GLOBL RWDLEN, P2TOP, RCNT1, RCNT2
```

```
58 . GLOBL KYCOMP, FLDMAP, MAPEND, INRLEN, SELADR
59 . GLOBL KYDIS, EOFBUF, ANDFLG, ORFLG
60 . GLOBL MSGNAM, COMPAR, REC1, REC2, POBASE
61 . GLOBL POTOP, P2SIZ, LOWMEM, P1SIZ
62 . GLOBL UEOF, V3FLAG, SRTCNT, TPCNT, P1TOP
63 . GLOBL KYBIN, KYTSS, KYLSS, KYDATE
64 . GLOBL SFLAGS, F$BIN, F$CR, F$PAUS, F$VLEN
65 . GLOBL CINNRT, SAVBUF
66 . GLOBL SETNRT, RSTS, CHNPPN, FILEPN, SPCFLG
67 . GLOBL FD1, FD2, FD3, FD4, SPC1, SPC2, SPC3, SPC4, SPC5, FILES
68 ;
69 ; MISC. PARAMETERS
70 ;
71 000015 CR = 15 ; ASCII CARRIAGE RETURN
72 000012 LF = 12 ; ASCII LINE FEED
73 000032 CTRLZ = 32 ; CONTROL-Z
```

```

1          ; -----
2          ; Format of record selection control blocks.
3          ;
4          000000 S$LINK = 0 ;W- LINK TO NEXT CONTROL BLOCK
5          000002 S$OPR = 2 ;B- TYPE OF OPERATION (SO$xxx)
6          000003 S$TYP = 3 ;B- TYPE OF DATA (FT$x)
7          000004 S$BR = 4 ;B- TRUTH TEST (EQ,NE,GT,LT,GE,LE)
8          000005 S$FLG = 5 ;B- CONTROL FLAGS (SF$xxx)
9          000006 S$POS1 = 6 ;W- BYTE POSITION OF FIELD 1
10         000010 S$SIZ1 = 10 ;W- # BYTES IN FIELD 1
11         000012 S$POS2 = 12 ;W- BYTE POSITION OF FIELD 2
12         ;
13         000014 S$$SZ = 14 ;SIZE OF SELECT CONTROL BLOCK
14         ;
15         ; Operation codes stored in S$OPR
16         ;
17         000000 SO$CMP = 0 ;FIELD COMPARISION
18         000002 SO$AND = 2 ;LOGICAL AND
19         000004 SO$OR = 4 ;LOGICAL OR
20         000006 SO$NOT = 6 ;LOGICAL NOT
21         ;
22         ; Branch type codes stored in S$BR
23         ;
24         000001 BT$HI = 1 ;BRANCH IF A>B
25         000002 BT$LO = 2 ;BRANCH IF A<B
26         000004 BT$REV = 4 ;REVERSE SENSE OF BRANCH
27         ;
28         ; Field type codes stored in S$TYP
29         ;
30         000000 FT$C = 0 ;C CHARACTER
31         000002 FT$B = 2 ;B SIGNED BINARY INTEGER
32         000004 FT$U = 4 ;U UNSIGNED BINARY INTEGER
33         000006 FT$T = 6 ;T DATE (MMDDYY)
34         000010 FT$F = 10 ;F FLOATING
35         000012 FT$D = 12 ;D DIBOL SIGNED NUMBER
36         000014 FT$A = 14 ;A UPPER CASE ASCII CHARACTERS
37         000016 FT$I = 16 ;I LEADING SEPARATE SIGN
38         000020 FT$J = 20 ;J TRAILING SEPARATE SIGN
39         ;
40         ; Control flags stored in S$FLG
41         ;
42         000001 SF$LIT = 1 ;FIELD 2 IS A LITERAL
43         ;
44         ; -----
45         ; Table of comparison operators.
46         ;
47         000000 105 121 CMPOPS: .ASCII /EQ/
48         000002 116 105 .ASCII /NE/
49         000004 114 124 .ASCII /LT/
50         000006 114 105 .ASCII /LE/
51         000010 107 124 .ASCII /GT/
52         000012 107 105 .ASCII /GE/
53         000014 NUMCO = .-CMPOPS ;2*# OPERATORS
54         ;
55         ; Parallel table of control flags.
56         ; BT$HI==>True if A>B.
57         ; BT$LO==>True if A<B.

```

```

58      ; BT$REV==>Reverse sense of comparison.
59      ;
60 000014      007      CMPCOD: . BYTE   BT$HI!BT$LO!BT$REV      ; EQ
61 000015      003      . BYTE   BT$HI!BT$LO      ; NE
62 000016      002      . BYTE   BT$LO      ; LT
63 000017      005      . BYTE   BT$HI!BT$REV      ; LE
64 000020      001      . BYTE   BT$HI      ; GT
65 000021      006      . BYTE   BT$LO!BT$REV      ; GE
66      . EVEN
67      000003      BT$NE   =      BT$HI!BT$LO      ; BT CODE FOR NE OPERATION
68      ;
69      ;-----
70      ; Operation codes used while parsing logical selection expression.
71      ; Operation code value corresponds to precedence value for operator.
72      ;
73      000001      OP$EOE   =      1      ; END OF EXPRESSION
74      000002      OP$BOE   =      2      ; BEGINNING OF EXPRESSION
75      000003      OP$RPN   =      3      ; RIGHT PAREN.
76      000004      OP$LPN   =      4      ; LEFT PAREN.
77      000005      OP$OR    =      5      ; "OR"
78      000006      OP$AND   =      6      ; "AND"
79      000007      OP$NOT   =      7      ; "NOT"
80      000010      OP$COR   =      10     ; ", " USED WITH EQ (SPECIAL "OR")
81      000011      OP$CND   =      11     ; ", " USED WITH NE (SPECIAL "AND")
82      ;
83      ; Table to translate OP$xxx code values to SO$xxx codes.
84      ;
85 000022      000      OPTRNS: . BYTE   0
86 000023      000      . BYTE   0      ; OP$EOE
87 000024      000      . BYTE   0      ; OP$BOE
88 000025      000      . BYTE   0      ; OP$RPN
89 000026      000      . BYTE   0      ; OP$LPN
90 000027      004      . BYTE   SO$OR   ; OP$OR
91 000030      002      . BYTE   SO$AND  ; OP$AND
92 000031      006      . BYTE   SO$NOT  ; OP$NOT
93 000032      004      . BYTE   SO$OR   ; OP$COR
94 000033      002      . BYTE   SO$AND  ; OP$CND
95      . EVEN
96      ;
97      ;-----
98      ; Format of field mapping control blocks.
99      ;
100     000000      M$POS1  =      0      ; W- POSITION OF SOURCE FIELD
101     000002      M$POS2  =      2      ; W- POSITION OF DESTINATION FIELD
102     000004      M$SIZE  =      4      ; W- NUMBER OF BYTES TO MOVE
103     ;
104     000006      M$SZ    =      6      ; SIZE OF MAPPING CONTROL BLOCK

```

```

1
2 000034 015270 R50DK: .RAD50 /DK/
3 000036 074452 R50TNO: .RAD50 /SOR/
4 000040 100760 R50TN1: .RAD50 /T1X/
5 000042 101030 R50TN2: .RAD50 /T2X/
6 000044 113050 R50XX: .RAD50 /XA /
7 000046 077430 R50TMP: .RAD50 /TMP/
8 000050 014646 R50DDF: .RAD50 /DDF/
9 000052 014474 R50DAT: .RAD50 /DAT/
10 000054 011541 R50CDA: .RAD50 /CDA/
11 000056 011542 R50CDB: .RAD50 /CDB/
12 000060 012106 R50CIO: .RAD50 /CIO/
13 000062 011576 R50CDO: .RAD50 /CDO/
14
15 000064 014646 DEFEXT: .RAD50 /DDF/ ;INPUT FILE DEFAULT EXTENSION
16 000066 000000 000000 000000 .WORD 0,0,0 ;OUTPUT FILE DEFAULT EXTENSIONS
17 000074 074644 INDDX: .RAD50 /SRT/ ;INDIRECT PARAMETER FILE EXTENSION
18 000076 000000 000000 000000 .WORD 0,0,0
19
20 000104 073376 CHADEX: .RAD50 /SAV/ ;CHAIN FILE DEFAULT EXTENSION
21 000106 000000 000000 000000 .WORD 0,0,0
22
23 000114 000 110 GTLN: .BYTE 0,110
24 000116 000 106 MSGBLK: .BYTE 0,106
25 000120 011601' .WORD SRTNAM
26 000122 000644' .WORD MSGBUF
27 000124 000454 .WORD 300.
28
29 ; MESSAGE REPLY EMT BLOCK
30 000126 000 104 SNDBLK: .BYTE 0,104
31 000130 000000G .WORD MSGNAM
32 000132 000136' .WORD ABTBUF
33 000134 000006 .WORD 6.
34 ; MESSAGE BUFFER
35 000136 000000 ABTBUF: .WORD 0
36 000140 000000 MRC2: .WORD 0
37 000142 000000 MRC1: .WORD 0
38
39 ; DATA AREAS
40
41 000144 TTBUF: .BLKW 160.
42 000644 MSGBUF: .BLKW 160.
43 001344 FILBUF: .BLKW 80.
44 001604 CINFIL: .BLKW 8.
45 001624 XBUF: .BLKW 50.
46 001770 SLBUF: .BLKW 25.
47 002052 SWBUF: .BLKW 120.
48 002432 GENBUF: .BLKW 256.
49 003432 CBLK: .BLKW 4
50 003442 CMPTMP: .BLKW 4
51 003452 000000 KEYWD: .WORD 0
52 003454 000000 VALHI: .WORD 0
53 003456 XAREA: .BLKW 8.
54 003476 000000 CURSEL: .WORD 0
55 003500 000000 FLDPOS: .WORD 0
56 003502 000000 FLDSIZ: .WORD 0
57 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

```

1      ;
2      ;
3      ; MACRO TO DEFINE SWITCH NAMES AND PROCESSING ROUTINES.
4      ;
5 000000 . CSECT SWNAM
6 000000 KEYTAB:
7 000000 . CSECT SWBRV
8 000000 KEYBR:
9      ;
10     . MACRO SW      NAME, RTN
11     . NLIST BIN
12     . CSECT SWNAM
13     . ASCII /'NAME'/
14     . CSECT SWBRV
15     . WORD RTN
16     . PSECT SORTID, D
17     . LIST BIN
18     . ENDM SW
19     ;
20     ; DEFINE SWITCHES (2 CHARS) AND PROCESSING ROUTINES.
21     ;
22 000000 SW      SI, SWSIZE      ; /SIZE: NNN
23 003542 SW      SK, SWSKIP     ; /SKIP: NNN
24 003542 SW      RE, SWREC        ; /RECORDS: NNN
25 003542 SW      ED, SWEOF        ; /EOF: CHAR: COL
26 003542 SW      CH, SWCHAN      ; /CHAIN: NAME
27 003542 SW      KE, SWKEY        ; /KEY: TAS. L: TAS. L:...
28 003542 SW      PA, SWPAUS      ; /PAUSE
29 003542 SW      BL, SWBLOK      ; /BLOCK: NN
30 003542 SW      IN, SWINCL      ; /INCLUDE: (expression)
31 003542 SW      MA, SWMAP      ; /MAP: size: s. d. l: s. d. l:...
32 003542 SW      CR, SWCR        ; /CR
33 003542 SW      SE, SWSND        ; /SEND: TEXT
34 003542 SW      BI, SWBIN        ; /BIN
35 003542 SW      IT, SWITYP       ; /ITYPE: type
36 003542 SW      OT, SWOTYP      ; /OTYPE: type
37 003542 SW      NU, SWNMBR     ; /NUMBER: col. len
38 003542 SW      PP, SWPPN     ; /PPN: [chain ppn]: [file ppn]
39     ;
40     ; DEFINE END OF SWITCH TABLES
41 000042 . CSECT SWNAM
42 000042 KEYEND:
43 003542 . PSECT SORTID, D
44     ;
45     ; KEY TYPE LETTERS AND TYPE CODES
46     ;
47 003542 103      102      125 KEYTC: . ASCII /CBUTFDAIJ/
48 003545 124      106      104
49 003550 101      111      112
50 003553 000
51 003554 000      002      004 KEYCOD: . BYTE FT%C, FT%B, FT%U, FT%T, FT%F, FT%D, FT%A, FT%I, FT%J
52 003557 006      010      012
53 003562 014      016      020
54     . EVEN
55     ;
56     ; DEFINE # OF BYTES OF MEMORY NEEDED FOR THE CODE GENERATED
57     ; TO COMPARE KEYS OF VARIOUS TYPES.

```

```
53
54 003566 000036
55 003570 000030
56 003572 000022
57 003574 000030
58 003576 000030
59 003600 000030
60 003602 000030
61 003604 000030
62 003606 000030
63
64
65
66 003610 011104'
67 003612 011306'
68 003614 011252'
69 003616 011336'
70 003620 011344'
71 003622 011360'
72 003624 011366'
73 003626 011322'
74 003630 011330'
```

```

;
CODSIZ: .WORD 30.          ; ASCII
        .WORD 24.          ; SIGNED INTEGER
        .WORD 18.          ; UNSIGNED INTEGER
        .WORD 24.          ; DATE
        .WORD 24.          ; FLOATING POINT
        .WORD 24.          ; DIBOL SIGNED NUMBER
        .WORD 24.          ; UPPERCASE ASCII STRING
        .WORD 24.          ; LEADING SEPARATE SIGN
        .WORD 24.          ; TRAILING SEPARATE SIGN
;
; BRANCH VECTOR TO CODE GENERATOR ROUTINES
;
GENVEC: .WORD GNASCI       ; ASCII
        .WORD GNCMP       ; SIGNED INTEGER
        .WORD GNUSI       ; UNSIGNED INTEGER
        .WORD GNDATE      ; DATE
        .WORD GNBIN       ; FLOATING POINT
        .WORD GNDIS       ; SIGNED DIBOL NUMBER
        .WORD GNUCA       ; UPPER-CASE ASCII COMPARE
        .WORD GNLSS       ; LEADING SEPARATE SIGN
        .WORD GNTSS       ; TRAILING SEPARATE SIGN
```

```
1
2 ; DEFINE ABORT MACRO
3 ;
4 . MACRO XXX COD
5 XXX'COD =
6 MOV #'COD',R0
7 JMP TERM
8 . ENDM XXX
9 ;
10 ; SORT ERROR MESSAGES
11 ;
12 003632 003750' ERRTAB: . WORD EM0
13 003634 003767' . WORD EM1
14 003636 004011' . WORD EM2
15 003640 004035' . WORD EM3
16 003642 004060' . WORD EM4
17 003644 004105' . WORD EM5
18 003646 004143' . WORD EM6
19 003650 004174' . WORD EM7
20 003652 004234' . WORD EM8
21 003654 004263' . WORD EM9
22 003656 004311' . WORD EM10
23 003660 004351' . WORD EM11
24 003662 004401' . WORD EM12
25 003664 004434' . WORD EM13
26 003666 004470' . WORD EM14
27 003670 004527' . WORD EM15
28 003672 004563' . WORD EM16
29 003674 004401' . WORD EM17
30 003676 004621' . WORD EM18
31 003700 004676' . WORD EM19
32 003702 004730' . WORD EM20
33 003704 004766' . WORD EM21
34 003706 005017' . WORD EM22
35 003710 005065' . WORD EM23
36 003712 005141' . WORD EM24
37 003714 005206' . WORD EM25
38 003716 005255' . WORD EM26
39 003720 005311' . WORD EM27
40 003722 005367' . WORD EM28
41 003724 005441' . WORD EM29
42 003726 005502' . WORD EM30
43 003730 005544' . WORD EM31
44 003732 005616' . WORD EM32
45 003734 005672' . WORD EM33
46 003736 005753' . WORD EM34
47 003740 006012' . WORD EM35
48 003742 006051' . WORD EM36
49 003744 006077' . WORD EM37
50 003746 006165' . WORD EM38
51 ;
```

```
1          .NLIST BIN
2 003750 EM0:  .ASCIZ /Sort completed/
3 003767 EM1:  .ASCIZ /Invalid # of keys/
4 004011 EM2:  .ASCIZ /Invalid record size/
5 004035 EM3:  .ASCIZ /Invalid field type/
6 004060 EM4:  .ASCIZ /Invalid field length/
7 004105 EM5:  .ASCIZ /Invalid field starting column/
8 004143 EM6:  .ASCIZ /Not Ascending-Descending/
9 004174 EM7:  .ASCIZ /Cannot load some device handler/
10 004234 EM8:  .ASCIZ /Cannot open input file/
11 004263 EM9:  .ASCIZ /Cannot open temp file/
12 004311 EM10: .ASCIZ /Not enough room for output file/
13 004351 EM11: .ASCIZ /Cannot open output file/
14 004401 EM12: .ASCIZ /Overflow of sort temp file/
15 004434 EM13: .ASCIZ /Invalid date in sort record/
16 004470 EM14: .ASCIZ /Error while reading input file/
17 004527 EM15: .ASCIZ /Error on write to temp file/
18 004563 EM16: .ASCIZ /Error on write to output file/
19          EM17 = EM12
20 004621 EM18: .ASCIZ /Temp file link failure (internal sort error)/
21 004676 EM19: .ASCIZ /Bad command string syntax/
22 004730 EM20: .ASCIZ /Unrecognizable command switch/
23 004766 EM21: .ASCIZ /Invalid switch parameter/
24 005017 EM22: .ASCIZ 'Required switch missing (/SIZE, /KEY)'/
25 005065 EM23: .ASCIZ /Unable to reopen sort temp file for phase 2/
26 005141 EM24: .ASCIZ /Insufficient memory space to do sort/
27 005206 EM25: .ASCIZ /Cannot open sort indirect command file/
28 005255 EM26: .ASCIZ /Error on SRTPLS message EMT/
29 005311 EM27: .ASCIZ /Invalid comparison operator in INCLUDE clause/
30 005367 EM28: .ASCIZ /Field types don't match in INCLUDE clause/
31 005441 EM29: .ASCIZ /Invalid syntax in INCLUDE clause/
32 005502 EM30: .ASCIZ /Invalid literal in INCLUDE clause/
33 005544 EM31: .ASCIZ /Invalid record size value with MAP switch/
34 005616 EM32: .ASCIZ /Invalid field specification with MAP switch/
35 005672 EM33: .ASCIZ /Record read is longer than specified record size/
36 005753 EM34: .ASCIZ /Invalid ITYPE switch parameter/
37 006012 EM35: .ASCIZ /Invalid DTYPE switch parameter/
38 006051 EM36: .ASCIZ /Invalid NUMBER switch/
39 006077 EM37: .ASCIZ /Inplace sort not legal with Indexed organization file/
40 006165 EM38: .ASCIZ /Only one input file is allowed if organization is Indexed/
```

```

1      ;
2      ; Sort help message
3      ;
4 006257 HLPMSG:
5 006257 .ASCII /SORT-Plus version 1.2/<CR><LF><LF>
6 006307 .ASCII /SORT command line:/<CR><LF><LF>
7 006334 .ASCII "[outspc]l,tmp1spc]l,tmp2spc]]=inspc1,inspc2,...,inspc6/sw1/sw2..."<CR><LF><LF>
8 006437 .ASCII /Valid switches:/<CR><LF>
9 006460 .ASCII "/SIZE:nnn -- record size (# bytes)"<CR><LF>
10 006526 .ASCII "/REC:nnn -- number of records to sort"<CR><LF>
11 006600 .ASCII "/SKIP:nnn -- number of records to skip at start"<CR><LF>
12 006663 .ASCII "/EOF[:'char'[:col]] -- end of file sentinel character"<CR><LF>
13 006754 .ASCII "/PAUSE -- ask for output disk after reading all input"<CR><LF>
14 007050 .ASCII "/CHAIN:progspc -- chain to specified program after sort"<CR><LF>
15 007143 .ASCII "/PPN:[chain ppn]:[file ppn] -- set default chain and file open"<CR><LF>
16 007243 .ASCII " PPNs (brackets required). Ignored if not under"<CR><LF>
17 007326 .ASCII " RSTS"<CR><LF>
18 007336 .ASCII "/SEND:'text' -- pass message through chain area"<CR><LF>
19 007417 .ASCII "/ITYPE:lor -- Input file language, organization & recording mode"<CR><LF>
20 007521 .ASCII "/OTYPE:lor -- Output file language, org., & mode"<CR><LF>
21 007603 .ASCII " Language types:"<CR><LF>
22 007625 .ASCII " C = COBOL-Plus"<CR><LF>
23 007647 .ASCII " D = DIBOL"<CR><LF>
24 007664 .ASCII " F = FORTRAN"<CR><LF>
25 007703 .ASCII " T = General variable length ascii text"<CR><LF>
26 007755 .ASCII " X = General fixed length binary/ascii data"<CR><LF>
27 010033 .ASCII " Organizations:"<CR><LF>
28 010054 .ASCII " S = Sequential"<CR><LF>
29 010076 .ASCII " R = Relative (Random)"<CR><LF>
30 010127 .ASCII " I = Indexed"<CR><LF>
31 010146 .ASCII " Recording mode:"<CR><LF>
32 010170 .ASCII " A = Ascii"<CR><LF>
33 010205 .ASCII " B = Binary"<CR><LF>
34 010223 .ASCII "/BLOCK:nn -- use specified blocking factor"<CR><LF>
35 010301 .ASCII "/MESSAGE -- enter message mode"<CR><LF>
36 010341 .ASCII "/KEY:tas.l[:tas.l:... ] -- specify sort keys"<CR><LF>
37 010420 .ASCII " t = type of key: C=ASCII character string,"<CR><LF>
38 010477 .ASCII " A=Upper-case ASCII,"<CR><LF>
39 010550 .ASCII " B=signed binary (COBOL COMP),"<CR><LF>
40 010633 .ASCII " T=date (mmddy),"<CR><LF>
41 010701 .ASCII " F=FORTRAN Real and Integer,"<CR><LF>
42 010762 .ASCII " D=COBOL signed numeric display and signed DIBOL number"<CR><LF>
43 011076 .ASCII " I=COBOL leading separate sign,"<CR><LF>
44 011162 .ASCII " J=COBOL trailing separate sign."<CR><LF>
45 011247 .ASCII " a = "A" for ascending, "D" for descending./<CR><LF>
46 011326 .ASCII " s = starting column #. (First col. # = 1)"<CR><LF>
47 011404 .ASCII " l = length of key (# of columns). "<CR><LF>
48 011452 .ASCII "/INCLUDE:(logical expression)"<CR><LF>
49 011511 .ASCII "/MAP:size:s.d.l:s.d.l:... -- specify field mapping"<CR><LF>
50 011577 .ASCIZ <LF>
51      ;
52 011601 SRTNAM: .ASCII /RTSORT/ ;SORT INPUT MESSAGE CHANNEL
53 011607 SRTFHD: .ASCII /?SRTPLS-F-/?<200>
54 011622 ASTRFX: .ASCIZ /*/<200>
55 011625 TXNKR: .ASCII /Records read: /<200>
56 011644 TXNRS: .ASCII /, Records sorted: /<200>
57 011670 CRUF: .BYTE 0

```

58 011671 CVTBUF: .ASCIZ /012345678/
59 .EVEN
60

.LIST BIN

```

1          .SBTTL SORT INITIALIZATION
2 000000   .PSECT SORTI          ;BEGIN PROCEDURE SECTION
3
4          ; INITIAL ENTRY TO SORT
5          ;
6 000000   SRTINI:
7 000000       $SYINIT
8          ;
9          ;;; SORTR:      $GETCMD #TTBUF,#160.      ;MACRO TO GET A COMMAND LINE
10         ; USE THE FOLLOWING CONDITIONAL ASSEMBLY INSTEAD OF
11         ; A MACRO. I WOULD HAVE PREFERED A MACRO, BUT IT TURNED
12         ; OUT TO BE MUCH TOO MESSY.
13         .IF DF RT11
14         ;
15         ; DETERMINE HOW WE WERE CALLED
16         ;
17 000170   SORTR:
18 000170   105767 000000G       TSTB   FSTIME      ;GETTING 1ST COMMAND LINE ?
19 000174   001405           BEQ     1$          ;BRANCH IF SO
20 000176   105767 000000G       TSTB   MODE        ;LOOK TO SEE HOW TO GET COMMAND
21 000202   002474           BLT     MWAIT       ;-1 ==> GO WAIT FOR A MESSAGE
22 000204   001410           BEQ     SORTD       ; 0 ==> GO GET A NORMAL COMMAND
23 000206           .EXIT                ;EXIT SORT PROGRAM TO KMON
24 000210           1$:
25 000210   152767 000001 000000G   BISB   #1,FSTIME      ;INDICATE NO LONGER 1ST TIME THRU
26 000216   032737 000400 000044   BIT    #CHNBIT,@#JSW  ;WERE WE CHAINED TO?
27 000224   001055           BNE     SORTC       ;BR IF CHAINED TO
28         ;
29         ; We are in direct run mode.
30         ;
31 000226   105067 000000G   SORTD: CLRB   MODE        ;SAY DIRECT RUN MODE
32         ; ACCEPT COMMAND LINE FROM USER
33 000232   012701 000144'       MOV    #TTBUF,R1      ;STORE COMMAND IN TTBUF
34 000236   105767 000000G       TSTB   V3FLAG       ;IS THIS RT-11 V2 OR V3?
35 000242   001416           BEQ     V2GLIN      ;BR IF RT V2
36         ; USE VERSION 3 .GT LIN EMT TO GET COMMAND LINE
37 000244   V3GLIN: .GT LIN R1,#ASTRTX      ;ACCEPT COMMAND LINE
38         ; SEE IF LINE IS CONTINUED
39 000262   105721           1$: TSTB   (R1)+        ;SCAN FOR TRAILING NULL
40 000264   001376           BNE     1$
41 000266   005301           DEC    R1
42 000270   124127 000055       CMPB   -(R1),#'-     ;DID LINE END WITH '-'?
43 000274   001763           BEQ    V3GLIN       ;BR IF LINE WAS CONTINUED
44 000276   000425           BR     GOTLIN       ;END OF COMMAND
45         ;
46         ; GET LINE USING .TTYIN IF UNDER RT V2
47         ;
48 000300   V2GLIN:
49 000300   112700 000052       5$:  MOVB   #'*,R0      ;PRINT PROMPT
50 000304           .TTYOUT
51 000310           1$: .TTYIN      ;ACCEPT CHAR FROM USER
52 000314   120027 000012       CMPB   R0,#LF       ;END OF LINE?
53 000320   001413           BEQ    2$          ;BR IF END OF COMMAND
54 000322   120027 000055       CMPB   R0,#'-       ;WANT TO CONTINUE LINE?
55 000326   001006           BNE    3$          ;BR IF NOT
56 000330           4$: .TTYIN      ;EAT REST OF THIS LINE
57 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	RO, (R1)+	; ACCRUE COMMAND LINE
61	000346	000760		BR	1*	
62	000350	105041	2*:	CLRB	-(R1)	; PUT NULL AT END OF COMMAND
63	000352	012701	GOTLIN:	MOV	#TTBUF, R1	; SCAN COMMAND FROM TTBUF
64	000356	000434		BR	SCNCMD	; GO PARSE COMMAND

```

1      ;
2      ; WE WERE CHAINED TO --- COMMAND IS IN 510-777
3      ;
4 000360 112767 000001 0000000 SORTC:  MOVB  #1,MODE      ;SET CHAIN MODE
5 000365 012701 000510          MOV   #510,R1      ;SCAN COMMAND FROM 510
6 000372 000426          BR    SCNCMD      ;GO SCAN COMMAND
7      ;
8      ; MWAIT IS JUMPED TO TO WAIT FOR ANOTHER MESSAGE COMMAND
9      ;
10 000374 112767 177777 0000000 MWAIT:  MOVB  #-1,MODE      ;REMEMBER WE'RE IN MESSAGE MODE
11 000402 012700 000116'          MOV   #MSGBLK,R0      ;WAIT FOR A MESSAGE
12 000406 104375          EMT    375
13 000410 103006          BCC   1$          ;BR IF NO ERROR
14 000412 105067 0000000          CLRB  MODE      ;GET OUT OF MESSAGE MODE
15 000416          XXX    26
16      ; GOT A MESSAGE.
17      ; STORE A NULL AT END OF MESSAGE
18 000426 105060 000644' 1$:    CLRB  MSGBUF(R0)
19 000432 012701 000644'          MOV   #MSGBUF,R1      ;POINT TO SEND MESSAGE BUFFER
20 000436 012702 0000000          MOV   #MSGNAM,R2      ;SET RETURN CHANNEL NAME
21 000442 012122          MOV   (R1)+,(R2)+      ;MOVE RETURN CHANNEL NAME
22 000444 012122          MOV   (R1)+,(R2)+
23 000446 012112          MOV   (R1)+,(R2)
24      ;
25      ; SCAN MESSAGE, IGNORE SPACES, PUT FILSPC INTO FILBUF,
26      ; PUT SWITCHES INTO SWBUF.
27      ; R1 POINTS TO START OF COMMAND STRING. NULL TERMINATES STRING.
28      ;
29 000450 121127 000057 SCNCMD:  CMPB  (R1),# '/'      ;IS THERE A LEADING SWITCH
30 000454 001021          BNE   2$          ;BR IF NOT
31 000456 126127 000001 000110          CMPB  1(R1),#'H      ;/HELP?
32 000464 001410          BEQ   8$          ;BR IF HELP
33 000466 126127 000001 000115          CMPB  1(R1),#'M      ;/M ONLY LEGAL ONE
34 000474 001737          BEQ   MWAIT      ;IT MEANS WAIT FOR A MESSAGE
35 000476          XXX    19
36      ; PRINT HELP MESSAGE
37 000506 8$:    .PRINT #HLPMSG
38 000514 000167 177450          JMP   SORTR      ;GO GET ANOTHER COMMAND LINE

```

```

1
2 ; See if the command begins with a \SPEC\, where SPEC is an initial
3 ; specification of a /CHAIN file but turns on a special error
4 ; handling switch. This switch specifies that if an error occurs,
5 ; the specified file is chained to with the 7 character low core
6 ; message of "ABORT x", where x is a single byte binary value that
7 ; contains the error code of the aborted sort.
8
9 000520 121127 000134 2$: CMPB (R1),#\ ;Special error mode?
10 000524 001015 BNE 1$ ; => no
11 000526 005201 INC R1 ;Yes, get the specification
12 000530 012702 001344' MOV #FILBUF,R2
13 000534 112112 3$: MOVB (R1)+,(R2)
14 000536 001404 BEQ 5$
15 000540 122227 000134 CMPB (R2)+,#\
16 000544 001373 BNE 3$
17 000546 105042 CLRB -(R2)
18 000550 004767 001510 5$: CALL SEICHN ;Do the decode
19 000554 105267 0000000 INCB SPCFLG ;And flag the special mode
20
21 ; SEE IF INDIRECT COMMAND LINE WAS SPECIFIED
22 ; (@FILE SPEC)
23
24 000560 121127 000100 1$: CMPB @R1,#'@ ;INDIRECT COMMAND LINE?
25 000564 001402 BEQ INDRCM ;BR IF YES
26 000566 000167 000266 JMP GETEND ;BR IF NOT
27
28 ; GET INDIRECT COMMAND LINE
29
30 000572 005201 INDRCM: INC R1 ;SKIP OVER "@"
31 000574 010605 MOV SP,R5 ;SAVE SP
32 000576 005037 000046 CLR @#USRL0C ;PUT USR IN TOP OF MEMORY
33 000602 .SETTOP #SIEND+2000. ;MAKE ROOM FOR HANDLERS
34 000610 $LOCK
35 000612 .CSISPC #XBUF,#INDEX,R1;PARSE COMMAND LINE
36 000626 103004 BCC 1$ ;BR IF NO ERROR
37 000630 9$: XXX 25 ;CAN'T OPEN INDIRECT FILE
38 ; LOAD HANDLER AND OPEN INDIRECT FILE
39 000640 010506 1$: MOV R5,SP ;RESET SP
40 000642 105767 0000000 TSTB RSTS ;Running under RSTS?
41 000646 001024 BNE 2$ ; => yes, no fetches needed
42 000650 .FETCH #SIEND,#XBUF+36 ;LOAD HANDLER
43 000662 103762 BCS 9$ ;BR IF ERROR
44 000664 .LOOKUP #XAREA,#1,#XBUF+36
45 000704 103005 BCC 2$
46 000706 8$: .RELEAS #XBUF+36 ;ERROR. RELEASE HANDLER.
47 000716 000744 BR 9$
48 ; READ 1ST BLOCK OF FILE
49 000720 2$: .READW #XAREA,#1,#GENBUF,#256, #0
50 000756 103753 BCS 0$
51 ; CLOSE COMMAND FILE
52 000760 .CLOSE #1
53 000766 .RELEAS #XBUF+36
54 000776 $UNLOCK
55 ; SUPPRESS TRAILING ENDS OF CONTINUATION LINES
56 001000 012701 002432' MOV #GENBUF,R1 ;START SCANNING HERE
57 001004 010102 MOV R1,R2 ;PUT RESULT HERE

```

```
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
```

```

1      ;
2      ; WE NOW HAVE COMMAND LINE.
3      ;
4 001060 SCNCM1:
5      ;
6      ; Check in for debugging breakpoint.
7      ;
8 001060 004767 000000G          CALL    CISRTI          ;CHECK IN TO ROOT
9      ;
10     ; Do initial memory allocation.
11     ;
12 001064          $GETTOP
13 001072 010067 000000G      MOV     RO, TOPMEM          ;SAVE TOP OF MEMORY ADDRESS
14 001076 012737 014574' 000046  MOV     #SIEND,@#USRLOC      ;SET USR SWAP LOCATION
15 001104 012704 000057      MOV     #'/,R4              ;Enable the switch finder
16 001110 010103      MOV     R1,R3              ;Save off string pointer
17 001112 105767 000000G      TSTB   RSTS              ;We running under RSTS?
18 001116 001420          BEQ     16$              ; => no
19 001120 112100          13$:  MOVB   (R1)+,RO          ;Yes, if there is an equal sign, then
20 001122 001416          BEQ     16$              ; we must ignore switches to the
21 001124 004767 011742      CALL   QUOTCK              ; left of it (/MODE:n, /SI:m, etc.)
22 001130 001006          BNE    15$
23 001132 112100          17$:  MOVB   (R1)+,RO          ;Don't look within quoted strings
24 001134 001411          BEQ     16$
25 001136 120067 003526'      CMPB   RO,QUOTE
26 001142 001373          BNE    17$
27 001144 000765          BR     13$
28 001146 020027 000075          15$:  CMP     RO,#'=
29 001152 001362          BNE    13$
30 001154 012704 000040      MOV     #' ,R4              ;We have a "=", disable switch finder
31 001160 010301          16$:  MOV     R3,R1              ;Reset (possibly) the string ptr
32 001162 012702 001344'      MOV     #FILBUF,R2         ;PUT FILE SPECS HERE
33 001166 105012          CLRB   (R2)
34 001170 105067 002052'      CLRB   SWBUF
35 001174 005003          CLR    R3
36 001176 112100          3$:   MOVB   (R1)+,RO          ;GET NEXT CHAR FROM COMMAND
37 001200 004767 011666      CALL   QUOTCK              ;START OF QUOTED STRING?
38 001204 001006          BNE    10$
39 001206 110022          11$:  MOVB   RO,(R2)+          ;ACCEPT ALL CHARS IN QUOTED STRING
40 001210 112100          MOVB   (R1)+,RO          ;GET NEXT CHAR IN STRING
41 001212 001426          BEQ    4$
42 001214 120067 003526'      CMPB   RO,QUOTE          ;END OF QUOTED STRING?
43 001220 001372          BNE    11$
44 001222 120027 000040          10$:  CMPB   RO,#'
45 001226 001763          BEQ    3$
46 001230 120027 000141          CMPB   RO,#141          ;LOWER CASE 'a'?
47 001234 103405          BLD    12$
48 001236 120027 000172          CMPB   RO,#172          ;LOWER CASE 'z'?
49 001242 101002          BHI    12$
50 001244 162700 000040          SUB    #40,RO           ;CONVERT TO UPPER CASE LETTER
51 001250 120004          12$:  CMPB   RO,R4              ;START OF SWITCHES?
52 001252 001006          BNE    4$
53 001254 005703          TST    R3
54 001256 001004          BNE    4$
55 001260 105012          CLRB   (R2)              ;END FILE SPEC
56 001262 012702 002052'      MOV     #SWBUF,R2         ;GO TO SWITCH BUFFER
57 001266 005203          INC    R3

```

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

```

1          .SBTTL  PARSE SWITCHES
2          ;
3          ;  PARSE SWITCH STRING
4          ;
5 001306 005067 000000G  PARSW:  CLR  NUMKY2      ;# OF KEYS
6 001312 005067 000000G          CLR  INRLEN      ;INPUT RECORD LENGTH
7 001316 005067 000000G          CLR  RECLEN
8 001322 005067 000000G          CLR  SKPCNT      ;# OF RECORDS TO SKIP
9 001326 005067 000000G          CLR  SRTCNT      ;# RECORDS PASSED TO SORT
10 001332 005067 000002G         CLR  SRTCNT+2
11 001336 005067 000000G         CLR  T MPCNT      ;# BLOCKS WRITTEN TO TEMP FILE
12 001342 005067 000002G         CLR  T MPCNT+2
13 001346 005067 000000G         CLR  SKPVAL
14 001352 005067 000000G         CLR  RCNT1      ;ZERO RECORD COUNTERS
15 001356 005067 000000G         CLR  RCNT2
16 001362 005067 000000G         CLR  EOFBUF
17 001366 005067 000000G         CLR  FLDMAP      ;NO FIELD MAPPING
18 001372 005067 000000G         CLR  SELADR      ;NO RECORD SELECTION
19 001376 005067 000000G         CLR  EOFCOL      ;NO USER EOF CHARACTER
20 001402 005067 000000G         CLR  BLKFC      ;NO BLOCKING FACTOR YET
21 001406 105067 000000G         CLR B  EOFCHR      ;END OF FILE CHARACTER
22 001412 105067 000000G         CLR B  UEOF      ;HAVEN'T HIT USER'S EOF RECORD
23 001416 005067 000000G         CLR  NUMLN      ;NO RECORD NUMBERING WANTED
24 001422 005067 000000G         CLR  SFLAGS      ;STATE FLAGS
25 001426 005067 000000G         CLR  ITYPE      ;NO INPUT FILE TYPE
26 001432 005067 000000G         CLR  OTYPE      ;NO OUTPUT FILE TYPE
27          . IF DF RT11
28 001436 005067 000000G          CLR  CINSPC      ;CHAIN FILE NAME
29 001442 005037 000510          CLR  @#510      ;NO CHAIN MESSAGE
30 001446 005067 000000G          CLR  CHNPPN      ;Default chain PPN = current PPN
31 001452 005067 000000G          CLR  FILPPN      ;As well as the default file PPN
32          . ENDC
33 001456 012767 177777 000000G   MOV  #177777,MXREC1 ;SET TO SORT INFINITE # RECORDS
34 001464 012767 177777 000000G   MOV  #177777,MXREC2
35          ; Initialize record number counter to zero.
36 001472 012700 000060          MOV  #'0,R0
37 001476 012702 000000G         MOV  #NUMBUF,R2      ;BCD RECORD COUNTER BUFFER
38 001502 110022 2$:  MOV B  R0,(R2)+ ;SET TO ALL ZERO
39 001504 020227 000010G         CMP  R2,#NUMBUF+8. ;DONE ALL?
40 001510 103774          BLO  2$      ;BR IF NOT
41          ;

```

```

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$:   MOVB   (R1)+,R0      ;GET NEXT CHAR
6 001526 120027 000101      CMPB   R0,#'A          ;IS IT A LETTER?
7 001532 103410              BLD    1$             ;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              BLD    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

```

```

1 ;-----
2 ; /SIZE:NNN -- RECORD SIZE
3 ;
4 001620 004767 011562 SWSIZE: CALL CKCOL ;CHECK FOR COLON
5 001624 004767 011262 CALL ACRDEC ;ACCRUE SIZE
6 001630 005705 TST R5 ;MUST BE >0
7 001632 003004 BGT 1#
8 001634 XXX 2 ;INVALID RECORD SIZE
9 001644 010567 000000G 1#: MOV R5,INRLEN ;SAVE RECORD LENGTH
10 001650 111100 MOVB (R1),R0 ;GET VALUE DELIMITER
11 001652 001421 BEQ NXSJ ;END OF SWITCHES
12 001654 120027 000057 CMPB R0,#'/ ;START OF NEXT SWITCH
13 001660 001416 BEQ NXSJ ;BR IF YES
14 001662 120027 000072 CMPB R0,#': ;ADDITIONAL PARAMETER?
15 001666 001404 BEQ 2# ;BR IF YES
16 001670 3#: XXX 19 ;INVALID SYNTAX
17 001700 005201 2#: INC R1 ;NEXT PARAMETER SHOULD BE "V"
18 001702 122127 000126 CMPB (R1)+,#'V ;BR IF BAD
19 001706 001370 BNE 3#
20 001710 052767 000000G 000000G BIS #F$VLEN,SFLAGS ;REMEMBER VARIABLE LENGTH
21 001716 000167 002544 NXSJ: JMP NXTSW
22 ;
23 ;-----
24 ; /SKIP:NNN -- SKIP # OF RECORDS
25 ;
26 001722 004767 011460 SWSKIP: CALL CKCOL ;CHECK FOR COLON
27 001726 004767 011160 CALL ACRDEC ;ACCRUE VALUE
28 001732 010567 000000G MOV R5,SKPCNT ;STORE SKIP COUNT
29 001736 010567 000000G MOV R5,SKPVAL
30 001742 000765 BR NXSJ
31 ;
32 ;-----
33 ; /REC:NNN -- SET MAX # OF RECORDS TO SORT
34 ;
35 001744 004767 011436 SWREC: CALL CKCOL ;CHECK FOR COLON
36 001750 004767 011136 CALL ACRDEC ;ACCRUE VALUE
37 001754 010567 000000G MOV R5,MXREC1 ;STORE LOW ORDER PART
38 001760 016767 003454' 000000G MOV VALHI,MXREC2 ;STORE HIGH ORDER PART
39 001766 000753 BR NXSJ
40 ;
41 ;-----
42 ; /BLOCK:NN -- SET INTERNAL FILE BLOCKING FACTOR.
43 ;
44 001770 004767 011412 SWBLOK: CALL CKCOL ;CHECK FOR COLON
45 001774 004767 011112 CALL ACRDEC ;ACCRUE DECIMAL NUMBER
46 002000 010567 000000G MOV R5,BLKFC ;SET BLOCKING FACTOR
47 002004 000744 BR NXSJ
48 . IF DF RT11
49 ;
50 ;-----
51 ; /SEND:'text' -- SEND MESSAGE THROUGH CHAIN DATA AREA.
52 ;
53 002006 004767 011374 SWSND: CALL CKCOL ;CHECK FOR COLON
54 002012 012702 000510 MOV #510,R2 ;POINT TO CHAIN DATA AREA
55 002016 112100 MOVB (R1)+,R0 ;GET 1ST CHARACTER OF MESSAGE
56 002020 004767 011046 CALL QUOTCK ;IS TEXT STRING QUOTED?
57 002024 001007 BNE 2# ;BR IF NOT

```

```

58 ; TEXT STRING IS QUOTED
59 002026 112100 4$: MOVB (R1)+,R0 ; MOVE CHARS IN MESSAGE TILL QUOTE HIT
60 002030 120067 003526' CMPB RO,QUOTE ; END OF MESSAGE?
61 002034 001412 BEQ 1$ ; BR IF YES
62 002036 110022 MOVB RO,(R2)+ ; STORE TEXT CHAR IN SAVE AREA
63 002040 001372 BNE 4$ ; BR IF NOT END OF COMMAND
64 002042 000406 BR 5$ ; HIT END OF COMMAND
65 ; TEXT STRING IS NOT QUOTED
66 002044 110022 2$: MOVB RO,(R2)+ ; MOVE CHAR TO SAVE AREA
67 002046 001404 BEQ 5$ ; BR IF HIT END OF COMMAND
68 002050 112100 MOVB (R1)+,R0 ; GET NEXT TEXT CHAR
69 002052 120027 000057 CMPB RO,#'/' ; START OF NEXT SWITCH?
70 002056 001372 BNE 2$ ; BR IF NOT
71 002060 005301 5$: DEC R1 ; POINT BACK ONE
72 002062 105012 1$: CLRB (R2) ; STORE NULL AT END OF STRING
73 002064 000403 BR NXSJ1 ; GO PROCESS NEXT SWITCH
74 .ENDC
75 ;
76 ;
77 ; /BIN -- BINARY RECORDS -- DON'T LOOK FOR CTRL-Z.
78 ;
79 002066 052767 0000006 0000006 SWBIN: BIS #F$BIN,SFLAGS ; REMEMBER THIS IS A BINARY RECORD
80 002074 000167 002366 NXSJ1: JMP NXTSW
81 ;
82 ;
83 ; /EOF:CHAR:COL --- DECLARE EOF SENTINEL
84 ;
85 002100 012767 000001 0000006 SWEOF: MOV #1,EOFCOL ; SET DEFAULT EOF COLUMN NUMBER
86 002106 112767 000032 0000006 MOVB #CTRLZ,EOFCHR ; AND EOF CHAR
87 002114 121127 000072 CMPB (R1),#': ; DID USER SPECIFY VALUE?
88 002120 001026 BNE 9$ ; BR IF NOT
89 002122 004767 011260 CALL CKCOL ; CHECK FOR COLON
90 ; SEE IF CHAR SPECIFIED IN QUOTES OR AS OCTAL VALUE
91 002126 111100 MOVB (R1),R0 ; GET NEXT CHARACTER
92 002130 004767 010736 CALL QUOTCK ; IS STRING QUOTED
93 002134 001004 BNE 1$ ; BR IF NOT
94 002136 005201 INC R1 ; SKIP OVER QUOTE
95 002140 112105 MOVB (R1)+,R5 ; GET EOF CHAR
96 002142 005201 INC R1 ; SKIP OVER TRAILING QUOTE
97 002144 000402 BR 2$
98 002146 004767 011036 1$: CALL ACROCT ; ACCRUE OCTAL CHAR VALUE
99 002152 110567 0000006 2$: MOVB R5,EOFCHR ; STORE CHARACTER CODE
100 002156 121127 000072 CMPB (R1),#': ; IS COLUMN NUMBER SPECIFIED?
101 002162 001005 BNE 9$ ; BR IF NOT (USE 1)
102 002164 005201 INC R1 ; SKIP OVER COLON
103 002166 004767 010720 CALL ACRDEC ; ACCRUE DEC VALUE OF COLUMN
104 002172 010567 0000006 MOV R5,EOFCOL ; SAVE EOF COLUMN NUMBER
105 002176 000167 002264 9$: JMP NXTSW

```

```

1      . IF DF RT11
2      ;
3      ;
4      ; /CHAIN: DEV: FILE. EXT --- CHAIN TO NEXT PROGRAM
5      ;
6      ; MOVE FILE NAME TO CINFIL AND INSERT TRAILING NULL
7 002202 004767 011200 SWCHAN: CALL CKCOL ;CHECK FOR COLON
8 002206 012702 001604' MOV #CINFIL,R2 ;PUT NAME HERE
9 002212 112100 2$: MOV (R1)+,R0 ;PICK UP CHAR FROM NAME
10 002214 001410 BEQ 1$ ;BR IF HIT END
11 002216 004767 010650 CALL QUOTCK ;IGNORE QUOTES AROUND THE NAME
12 002222 001773 BEQ 2$
13 002224 120027 000057 CMPB RO,#'/' ;HIT NEXT SWITCH?
14 002230 001402 BEQ 1$ ;BR IF YES
15 002232 110022 MOVB RO,(R2)+ ;MOVE TO CINFIL
16 002234 000766 BR 2$
17 002236 105012 1$: CLRB (R2) ;PUT NULL AT END OF NAME
18 002240 005301 DEC R1 ;POINT BACK TO DELIMITER
19 002242 004767 000016 CALL GETCHN ;Set up the chain info
20 002246 103004 BCC 10$
21 002250 XXX 21
22 002260 000167 002202 10$: JMP NXTSW
23
24 ; LET . CSISPC PARSE FILE NAME
25 002264 010605 SETCHN: MOV SP,R5 ;SAVE SP
26 002266 . CSISPC #XBUF,#CHADEx,#CINFIL
27 002304 103426 BCS 5$ ;BR IF BAD NAME
28 002306 010506 MOV R5,SP ;RESTORE SP
29 ; MOVE FILE NAME TO SAVE AREA IN ROOT
30 002310 012702 001662' MOV #XBUF+36,R2
31 002314 012703 000000G MOV #CINSPC,R3
32 002320 012223 4$: MOV (R2)+,(R3)+
33 002322 020327 000010G CMP R3,#CINSPC+10
34 002326 103774 BLD 4$
35 002330 105767 000000G TSTB RSTS ;We under RSTS?
36 002334 001412 BEQ 5$ ; => no
37 002336 013702 000054 MOV @#54,R2 ;Yes, save off the special info
38 002342 062702 000400 ADD #CSPCOF+2,R2 ; (See the SVRSTS routine's comments
39 002346 012703 000000G MOV #CINNRT,R3 ; for what is happening here)
40 002352 012223 MOV (R2)+,(R3)+
41 002354 012223 MOV (R2)+,(R3)+
42 002356 012223 MOV (R2)+,(R3)+
43 002360 012223 MOV (R2)+,(R3)+
44 002362 000207 5$: RETURN
45 . ENDC

```

```

1 ;
2 ;-----
3 ; /KEY: TDS. L: TDS. L: ... -- DEFINE SORT KEYS
4 ;
5 002364 016704 000000G SWKEY: MOV NUMKY2, R4 ; GET # OF KEYS SO FAR
6 002370 062767 000002 000000G ADD #2, NUMKY2 ; INC #
7 002376 004767 011004 CALL CKCOL ; CHECK FOR COLON
8 ; GET 1ST CHAR WHICH DEFINES TYPE OF KEY
9 002402 112100 MOVB (R1)+, R0
10 002404 004767 007240 CALL ACRFLD ; PARSE FIELD SPEC
11 002410 016764 003506' 000000G MOV FLDORD, KAD(R4) ; SET ASCENDING/DESCENDING FLAG
12 002416 016764 003500' 000000G MOV FLDPOS, KSTRT(R4) ; SET STARTING POSITION OF FIELD
13 002424 016764 003502' 000000G MOV FLDSIZ, KLEN(R4) ; SET FIELD LENGTH
14 002432 016764 003504' 000000G MOV FLDTYP, KTYPE(R4) ; SET FIELD DATA TYPE
15 002440 121127 000072 CMPB (R1), #' ; IS THERE ANOTHER KEY DESCRIPTOR?
16 002444 001747 BEQ SWKEY ; YES -- GO GET IT
17 002446 000167 002014 JMP NXTSW
18 . IF DF RT11
19 ;
20 ;-----
21 ; /PAUSE -- PAUSE AFTER READING INPUT
22 ;
23 002452 052767 000000G 000000G SWPAUS: BIS #F$PAUS, SFLAGS ; REMEMBER TO PAUSE
24 002460 000167 002002 JMP NXTSW
25 . ENDC
26 ;
27 ;-----
28 ; /NUMBER: col. len -- Insert record number.
29 ;
30 002464 004767 010716 SWNMBR: CALL CKCOL ; CHECK FOR COLON
31 002470 004767 010416 CALL ACRDEC ; GET STARTING COLUMN NUMBER
32 002474 005305 DEC R5 ; CONVERT TO BYTE INDEX INTO RECORD
33 002476 010567 000000G MOV R5, NUMCOL
34 002502 122127 000056 CMPB (R1)+, #' ; PERIOD SHOULD BE DELIMITER
35 002506 001404 BEQ 1$ ; BR IF OK
36 002510 XXX 36 ; INVALID SWITCH
37 002520 004767 010366 1$: CALL ACRDEC ; GET LENGTH OF NUMBER FIELD
38 002524 010567 000000G MOV R5, NUMLEN
39 002530 000167 001732 JMP NXTSW
40 . IF DF RT11
41 ;
42 ;-----
43 ; /PPN: [chain ppn]: [file ppn] -- Define default chain and file
44 ; open request PPNs for RSTS
45 ;
46 002534 004067 000022 SWPPN: JSR R0, 5$ ; Process the first switch
47 002540 000000G . WORD CHNPPN
48 002542 121127 000072 CMPB (R1), #' ; Is there a second part?
49 002546 001003 BNE 1$ ; => no
50 002550 004067 000006 JSR R0, 5$ ; Yes, process it, also
51 002554 000000G . WORD FILPPN
52 002556 000167 001704 1$: JMP NXTSW
53 ;
54 002562 004767 010620 5$: CALL CKCOL ; Insure colon present
55 002566 121127 000072 CMPB (R1), #' ; Null entry?
56 002572 001432 BEQ 10$ ; => yes
57 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

```

```

1 ;-----
2 ; Get file type codes (RT$xxx) specified with the /ITYPE & /DTYPE switches.
3 ; On return the flags are in R2.
4 ;
5 002756 004767 010424 GETTYP: CALL CKCOL ;CHECK FOR COLON
6 002762 005002 CLR R2 ;BUILD FLAGS IN R2
7 ;
8 ; Get language type.
9 ;
10 002764 112100 MOVB (R1)+,R0 ;GET LANGUAGE CODE LETTER
11 002766 120027 000103 CMPB RO,#'C ;COBOL?
12 002772 001003 BNE 1$ ;BR IF NOT
13 002774 052702 000000G BIS #RT$CBL,R2 ;COBOL FLAG
14 003000 000435 BR GTORG
15 003002 120027 000104 1$: CMPB RO,#'D ;DIBOL?
16 003006 001003 BNE 2$ ;BR IF NOT
17 003010 052702 000000G BIS #RT$DBL,R2 ;DIBOL FLAG
18 003014 000427 BR GTORG
19 003016 120027 000106 2$: CMPB RO,#'F ;FORTRAN?
20 003022 001003 BNE 3$ ;BR IF NOT
21 003024 052702 000000G BIS #RT$FTN,R2 ;FORTRAN FLAG
22 003030 000421 BR GTORG
23 003032 120027 000130 3$: CMPB RO,#'X ;ARBITRARY DATA?
24 003036 001003 BNE 4$ ;BR IF NOT
25 003040 052702 000000G BIS #RT$X,R2 ;ARBITRARY DATA TYPE
26 003044 000413 BR GTORG
27 003046 120027 000124 4$: CMPB RO,#'T ;TEXT FILE?
28 003052 001003 BNE 5$ ;BR IF NOT
29 003054 052702 000000G BIS #RT$TXT,R2 ;TEXT FLAG
30 003060 000405 BR GTORG
31 003062 120027 000122 5$: CMPB RO,#'R ;FILES-11 FORMAT?
32 003066 001066 BNE GTERR ;NOPE, AND NONE OF ABOVE
33 003070 052702 000000G BIS #RT$F11,R2
34 ;
35 ; Get file organization type.
36 ;
37 003074 111100 GTORG: MOVB (R1),R0 ;GET ORGANIZATION CODE LETTER
38 003076 001434 BEQ 5$ ;BR IF HIT END OF COMMAND
39 003100 120027 000057 CMPB RO,#'/' ;HIT NEXT SWITCH?
40 003104 001431 BEQ 5$ ;BR IF YES
41 003106 005201 INC R1 ;ADVANCE CHAR POINTER
42 003110 120027 000123 CMPB RO,#'S ;SEQUENTIAL ORG?
43 003114 001003 BNE 1$ ;BR IF NOT
44 003116 052702 000000G BIS #RT$OS,R2 ;ORG=SEQ
45 003122 000425 BR GTRMD
46 003124 120027 000122 1$: CMPB RO,#'R ;RELATIVE ORGANIZATION?
47 003130 001003 BNE 2$ ;BR IF NOT
48 003132 052702 000000G BIS #RT$OR,R2 ;ORG=RELATIVE
49 003136 000417 BR GTRMD
50 003140 120027 000111 2$: CMPB RO,#'I ;ISAM ORG?
51 003144 001403 BEQ 3$ ;BR IF YES
52 003146 120027 000130 CMPB RO,#'X ;ISAM ORG?
53 003152 001034 BNE GTERR ;ERROR IF NONE OF ABOVE
54 003154 032702 000000G 3$: BIT #RT$CBL,R2 ;ISAM ONLY LEGAL FOR COBOL
55 003160 001431 BEQ GTERR ;BR IF NOT COBOL
56 003162 052702 000000G BIS #RT$OX,R2 ;ORG=ISAM
57 003166 000403 BR GTRMD ;GO GET RECORDING MODE

```

```

58 ; Set default organization and recording mode.
59 003170 052702 000000C 5$: BIS #RT$OS+RT$RA,R2 ;ORG=SEQUENTIAL, RECORDING MODE = ASCII
60 003174 000421 BR @TOK
61 ;
62 ; Get recording mode.
63 ;
64 003176 111100 GTRMD: MOVB (R1),R0 ;GET RECORDING MODE CODE LETTER
65 003200 001407 BEQ 1$ ;BR IF HIT END OF COMMAND
66 003202 120027 000057 CMPB RO,#'/ ;HIT NEXT SWITCH?
67 003206 001404 BEQ 1$ ;BR IF YES
68 003210 005201 INC R1 ;ADVANCE CHAR POINTER
69 003212 120027 000101 CMPB RO,#'A ;ASCII?
70 003216 001003 BNE 2$ ;BR IF NOT
71 003220 052702 000000G 1$: BIS #RT$RA,R2 ;MODE=ASCII
72 003224 000405 BR @TOK
73 003226 120027 000102 2$: CMPB RO,#'B ;BINARY?
74 003232 001004 BNE @TERR ;ERROR IF NEITHER
75 003234 052702 000000G BIS #RT$RB,R2 ;MODE=BINARY
76 ;
77 ; Finished.
78 ;
79 003240 000241 @TOK: CLC ;SIGNAL GOOD RETURN
80 003242 000207 RETURN
81 ;
82 ; Error.
83 ;
84 003244 000261 @TERR: SEC ;SIGNAL ERROR ON RETURN
85 003246 000207 RETURN

```

```

1 ; -----
2 ; /INCLUDE: (expression) -- Perform record selection.
3 ;
4 003250 004767 010132 SWINCL: CALL CKCOL ;CHECK FOR COLON
5 003254 012767 003537' 003540' MOV #OPRSTP,OPRSTK ;INIT OPERATOR STACK
6 003262 005002 CLR R2 ;NO SELECT BLOCKS YET
7 003264 105067 000000G CLRB ANDFLG ;NO "AND" OPERATORS YET
8 003270 105067 000000G CLRB ORFLG ;NO "OR" OPERATORS YET
9 ;
10 ; Get next comparison specification.
11 ;
12 003274 112100 SLCSPC: MOVB (R1)+,R0 ;GET NEXT CHARACTER
13 003276 120027 000050 CMPB R0,#'(' ;OPEN PAREN?
14 003302 001006 BNE 4$ ;BR IF NOT
15 ; Got open paren. Push operator on stack.
16 003304 005367 003540' DEC OPRSTK ;MAKE ROOM ON STACK
17 003310 112777 000004 003540' MOVB #OP$LPN,@OPRSTK ;PUT LEFT PAREN OPERATOR ON OPERATOR STACK
18 003316 000766 BR SLCSPC
19 ;
20 ; Check for "NOT" operator
21 ;
22 003320 120027 000116 4$: CMPB R0,#'N ;START OF "NOT"?
23 003324 001022 BNE 1$ ;BR IF NOT
24 003326 122127 000117 CMPB (R1)+,#'O ;CHECK SPELLING
25 003332 001013 BNE 9$
26 003334 122127 000124 CMPB (R1)+,#'T
27 003340 001010 BNE 9$
28 003342 012703 000007 MOV #OP$NOT,R3 ;SAY OPERATOR IS "NOT"
29 003346 105267 000000G INCB ANDFLG ;FORCE FULL EXPRESSION EVALUATION
30 003352 105267 000000G INCB ORFLG
31 003356 000167 000436 JMP SLOPX ;DO EXPRESSION GENERATION
32 003362 9$: XXX 29 ;INVALID SELECT SYNTAX
33 ;
34 ; This should be the start of a field specification.
35 ;
36 003372 004767 006252 1$: CALL ACRFLD ;ACCRUE FIELD SPECIFICATION
37 ; Build a skeleton selection control block.
38 003376 162767 000014 000000G SUB #S$$SZ, TOPMEM ;RESERVE ROOM FOR BLOCK
39 003404 016703 000000G MOV TOPMEM,R3 ;BUILD BLOCK HERE
40 003410 005702 TST R2 ;IS THIS THE FIRST BLOCK?
41 003412 001403 BEQ 2$ ;BR IF YES
42 003414 010362 000000 MOV R3,S$LINK(R2) ;CHAIN LAST BLOCK FORWARD TO NEW BLOCK
43 003420 000402 BR 3$
44 003422 010367 000000G 2$: MOV R3,SELADR ;REMEMBER ADDRESS OF FIRST CONTROL BLOCK
45 003426 010302 3$: MOV R3,R2 ;ADDR OF NEW BLOCK IS CARRIED IN R2
46 003430 010267 003476' MOV R2,CURSEL ;REMEMBER ADDR OF CURRENT SELECT BLOCK
47 003434 005062 000000 CLR S$LINK(R2) ;CLEAR OUR FORWARD LINK
48 003440 105062 000005 CLRB S$FLG(R2) ;CLEAR CONTROL FLAG BYTE
49 003444 112762 000000 000002 MOVB #SO$CMP,S$OPR(R2);SAY OPERATION IS FIELD COMPARISON
50 ; Store field specification information in select block.
51 003452 016762 003500' 000006 MOV FLDPOS,S$POS1(R2);STARTING BYTE POSITION
52 003460 016762 003502' 000010 MOV FLDSIZ,S$SIZ1(R2);FIELD SIZE
53 003466 116762 003504' 000003 MOVB FLDTYP,S$TYP(R2);DATA TYPE
54 ; Now get comparison operator.
55 003474 112103 SLCMOP: MOVB (R1)+,R3 ;GET 1ST CHARACTER
56 003476 112100 MOVB (R1)+,R0 ;GET 2ND CHARACTER
57 003500 000300 SWAB R0 ;PUT TOGETHER THE 2 CHARACTERS

```

```

58 003502 050300          BIS      R3,R0
59 003504 012703 000014    MOV      #NUMCD,R3      ;GET 2* NUMBER OF OPERATORS
60 003510 020063 177776'  2$:     CMP      RO,CMPOPS-2(R3) ;LOOK UP OPERATOR NAME
61 003514 001407          BEQ      1$              ;BR IF FOUND
62 003516 162703 000002    SUB      #2,R3          ;CHECK NEXT
63 003522 001372          BNE      2$              ;
64 003524                XXX      27              ;INVALID COMPARISON OPERATOR
65                ; Convert operator to 2-bit type code.
66 003534 006203          1$:     ASR      R3              ;CONVERT TO BYTE TABLE INDEX
67 003536 116362 000013' 000004    MOVB     CMPCOD-1(R3),S$BR(R2) ;SET COMPARISON CODE
68                ;
69                ; Get operand which follows comparison operator.
70                ;
71 003544 112100          SLOPND: MOVB     (R1)+,R0      ;GET 1ST LETTER OF OPERAND
72 003546 120027 000101    CMPB     RO,#'A         ;SEE IF IT IS A LETTER
73 003552 103403          BLO      2$              ;BR IF NOT
74 003554 120027 000132    CMPB     RO,#'Z         ;
75 003560 003406          BLE      1$              ;BR IF LETTER
76                ; Operand must be a literal.
77                ; Get it and store in comparison block.
78 003562 004767 006416    2$:     CALL     SLLIT        ;ACCRUE LITERAL VALUE
79 003566 152762 000001 000005    BISB     #$F$LIT,S$FLG(R2);SAY OPERAND 2 IS A LITERAL
80 003574 000415          BR       SLOPR          ;GO CHECK NEXT OPERATOR
81                ; Operand must be another field descriptor.
82                ; Get it and store info in select control block.
83 003576 004767 006046    1$:     CALL     ACRFLD       ;ACCRUE FIELD DESCRIPTOR
84 003602 016762 003500' 000012    MOV      FLDPDS,S$POS2(R2);STORE FIELD POSITION
85 003610 126762 003504' 000003    CMPB     FLDTYP,S$TYP(R2);COMPARE FIELD DATA TYPES
86 003616 001404          BEQ      SLOPR          ;BR IF THEY MATCH
87 003620                XXX      28              ;FIELD TYPES DON'T MATCH
88                ;
89                ; We have finished building a comparison control block.
90                ; See what follows.
91                ;
92 003630 112100          SLOPR:  MOVB     (R1)+,R0      ;GET NEXT CHARACTER
93 003632 001403          BEQ      1$              ;BR IF HIT END OF COMMAND LINE
94 003634 120027 000057    CMPB     RO,#'/'        ;START OF NEXT SWITCH?
95 003640 001004          BNE      2$              ;BR IF NOT
96 003642 005301          1$:     DEC      R1              ;POINT BACK TO DELIMITER
97 003644 112703 000001    MOVB     #OP$EOE,R3     ;SET END-OF-EXPRESSION OPERATOR
98 003650 000463          BR       SLOPX          ;FORCE EXPRESSION EVALUATION
99 003652 120027 000051    2$:     CMPB     RO,#')'    ;RIGHT PAREN?
100 003656 001003          BNE      3$              ;BR IF NOT
101 003660 112703 000003    MOVB     #OP$RPN,R3     ;SET RIGHT PAREN OPERATOR CODE
102 003664 000455          BR       SLOPX          ;EVALUATE EXPRESSION
103                ; Check for "AND" and "OR" logical operators.
104 003666 120027 000101    3$:     CMPB     RO,#'A'    ;START OF "AND"?
105 003672 001013          BNE      4$              ;BR IF NOT
106 003674 122127 000116    CMPB     (R1)+,#'N'     ;CHECK REST OF SPELLING
107 003700 001016          BNE      7$              ;
108 003702 122127 000104    CMPB     (R1)+,#'D'     ;
109 003706 001013          BNE      7$              ;
110 003710 112703 000006    MOVB     #OP$AND,R3     ;SET AND OPERATOR
111 003714 105267 000000G    INCB     ANDFLG         ;REMEMBER WE'VE HAD AN AND OPERATOR
112 003720 000437          BR       SLOPX          ;EVALUATE EXPRESSION
113 003722 120027 000117    4$:     CMPB     RO,#'O'    ;START OF "OR"?
114 003726 001014          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$OR,R3       ;SET COMMA-OR OPERATOR
129 004002 105267 000000G        INCB   ORFLG            ;REMEMBER WE'VE HAD AN OR OPERATION
130 004006 000404          BR     SLOPX
131 004010 012703 000011          7$:    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)     ;CLEAR 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,

```

```

172          ; generate a comparison block to hold info about next operand
173          ; in list.
174          ;
175 004152 120327 000010          CMPB   R3,#OP$COR          ; COMMA-OR?
176 004156 001405          BEQ    1$                    ; BR IF YES
177 004160 120327 000011          CMPB   R3,#OP$CND          ; COMMA-AND?
178 004164 001402          BEQ    1$                    ; BR IF YES
179 004166 000167 177102          JMP    SLCSPC              ; GO GET NEXT OPERAND
180          ; Build comparison control block for next operand in list.
181 004172 162767 000014 0000006 1$:  SUB    #$SZ, TOPMEM        ; RESERVE ROOM FOR NEXT BLOCK
182 004200 016762 0000006 0000000  MOV    TOPMEM, S$LINK(R2); CHAIN LAST BLOCK FORWARD TO NEW BLOCK
183 004206 016702 0000006          MOV    TOPMEM, R2          ; GET ADDRESS OF NEW BLOCK TO R2
184 004212 005062 000000          CLR    S$LINK(R2)        ; CLEAR OUR FORWARD LINK
185 004216 016703 003476          MOV    CURSEL, R3         ; GET ADDRESS OF LAST COMPARISON BLOCK
186 004222 016362 000006 000006  MOV    S$POS1(R3), S$POS1(R2); COPY INFO FROM LAST BLOCK
187 004230 016362 000010 000010  MOV    S$SIZ1(R3), S$SIZ1(R2)
188 004236 116362 000003 000003  MOVB   S$TYP(R3), S$TYP(R2)
189 004244 116362 000002 000002  MOVB   S$OPR(R3), S$OPR(R2)
190 004252 116362 000004 000004  MOVB   S$BR(R3), S$BR(R2)
191 004260 105062 000005          CLRB   S$FLG(R2)
192 004264 000167 177254          JMP    SLOPND              ; GO GET NEXT OPERAND IN LIST

```

```

1
2 ; /MAP: size: s. d. l: s. d. l: ... -- specify field mapping
3 ;
4 004270 004767 007112 SWMAP: CALL CKCOL ; CHECK FOR COLON
5 004274 004767 006612 CALL ACRDEC ; GET OUTPUT RECORD SIZE
6 004260 010567 000000 MOV R5, RECLN ; SAVE OUTPUT RECORD SIZE
7 004304 122127 000072 CMPB (R1)+, #' ; SIZE SHOULD BE FOLLOWED BY COLON
8 004310 001404 BEQ 1# ; BR IF OK
9 004312 XXX 31 ; INVALID RECORD SIZE SPEC WITH /MAP
10 004322 016702 000000 1#: MOV TOPMEM, R2 ; GET ADDR OF FREE MEMORY AREA
11 004326 162702 000006 SUB #M##SZ, R2 ; RESERVE ROOM FOR 1ST MAP BLOCK
12 004332 010267 000000 MOV R2, FLDMAP ; REMEMBER ADDR OF 1ST BLOCK
13 ; Parse next field descriptor.
14 004236 004767 006550 4#: CALL ACRDEC ; ACCRUE STARTING COLUMN NUMBER
15 004242 005305 DEC R5 ; CONVERT TO BYTE NUMBER
16 004244 002405 BLT 9# ; BR IF INVALID
17 004346 010562 000000 MOV R5, M#POS1(R2) ; SAVE POS OF SOURCE FIELD
18 004352 122127 000056 CMPB (R1)+, #' ; PERIOD SHOULD BE DELIMITER
19 004356 001404 BEQ 2# ; BR IF OK
20 004360 9#: XXX 32 ; INVALID /MAP FIELD SPECIFICATION
21 004370 004767 006516 2#: CALL ACRDEC ; ACCRUE DESTINATION FIELD COLUMN NUMBER
22 004374 005305 DEC R5 ; CONVERT TO BYTE NUMBER
23 004376 002770 BLT 9# ; BR IF INVALID
24 004400 010562 000002 MOV R5, M#POS2(R2) ; SAVE DESTINATION FIELD POSITION
25 004404 122127 000056 CMPB (R1)+, #' ; PERIOD SHOULD BE DELIMITER
26 004410 001363 BNE 9# ; BR IF ERROR
27 004412 004767 006474 CALL ACRDEC ; ACCRUE LENGTH SPECIFICATION
28 004416 010562 000004 MOV R5, M#SIZE(R2) ; SAVE FIELD LENGTH
29 004422 122127 000072 CMPB (R1)+, #' ; IS THERE ANOTHER FIELD SPEC?
30 004426 001003 BNE 5# ; BR IF NOT
31 004430 162702 000006 SUB #M##SZ, R2 ; MAKE ROOM FOR ANOTHER MAP BLOCK
32 004434 000740 BR 4# ; GO ACCRUE NEXT FIELD DESCRIPTOR
33 ; Hit end of /MAP switch specification
34 004436 005301 5#: DEC R1 ; POINT BACK TO DELIMITER
35 004440 010267 000000 MOV R2, MAPEND ; SAVE ADDRESS OF LAST MAP BLOCK
36 004444 010267 000000 MOV R2, TOPMEM ; SAVE NEW TOP-OF-MEMORY ADDRESS
37 004450 000167 000012 JMP NXTSW ; GO GET NEXT SWITCH
38 . IF DF RT11
39 ;
40 ;
41 ; /CR -- Append cr-lf to end of records.
42 ;
43 004454 052767 000000 000000 SWCR: BIS #F$CR, SFLAGS ; REMEMBER FLAG OCCURED
44 004462 000167 000000 JMP NXTSW ; PROCESS NEXT SWITCH
45 . ENDC
46 ;
47 ;
48 ; MOVE ON TO NEXT SWITCH
49 ;
50 004466 112100 NXTSW: MOVB (R1)+, R0 ; GET SWITCH DELIMITER
51 004470 001411 BEQ ENDSW ; BR IF HIT END OF SWITCHES
52 004472 120027 000057 CMPB R0, #' / ; START OF NEW SWITCH?
53 004476 001404 BEQ 1# ; BR IF YES
54 004500 XXX 19
55 004510 000167 175002 1#: JMP GETSW ; GO PROCESS NEXT SWITCH

```

```

1      ;
2      ;   GOT ALL SWITCHES.   CHECK THEM.
3      ;
4 004514 005767 000000G   ENDSW:  TST      INRLEN      ; DID WE GET RECORD LENGTH?
5 004520 001010          BNE      3$          ; BR IF YES
6 004522 032767 000000G 000000G   BIT      #RT$OX, ITYPE   ; IS THIS AN ISAM FILE?
7 004530 001004          BNE      3$          ; DON'T HAVE TO SPECIFY RECORD LENGTH FOR ISAM FILE
8 004532          2$:   XXX      22
9 004542 116701 000000G   3$:   MOVB     NUMKY2, R1   ; GET # OF KEYS SPECIFIED
10 004546 001771          BEQ      2$          ; ERROR IF NONE
11 004550 005002          CLR      R2
12 004552 026227 000000G 000000G 5$:   CMP      KTYPE(R2), #FT$T ; DATE TYPE KEY?
13 004560 001003          BNE      4$          ; BR IF NOT
14 004562 012762 000000G 000000G   MOV      #6, KLEN(R2)   ; SET DATE KEY LENGTH TO 6 CHARSS
15 004570 062702 000000G   4$:   ADD      #2, R2     ; CHECK NEXT KEY
16 004574 020201          CMP      R2, R1         ; CHECKED ALL?
17 004576 103765          BLD      5$          ; BR IF NOT
18      ;
19      ;   Check input and output file types.
20      ;
21 004600 016702 000000G   CKFTYP: MOV      ITYPE, R2   ; GET INPUT FILE TYPE FLAGS
22 004604 001002          BNE      1$          ; BR IF TYPE SPECIFIED
23 004606 012702 000000G   MOV      #RT$DBL+RT$OR+RT$RB, R2 ; DEFAULT TO DIBOL, RELATIVE, BINARY
24 004612 016703 000000G   1$:   MOV      OTYPE, R3   ; GET OUTPUT FILE TYPE FLAGS
25 004616 001010          BNE      2$          ; BR IF SOME SPECIFIED
26 004620 010203          MOV      R2, R3         ; DEFAULT TO SAME TYPE AS INPUT
27 004622 032702 000000G   BIT      #RT$OX, R2     ; IS INPUT FILE ORG = INDEXED?
28 004626 001404          BEQ      2$          ; BR IF NOT
29 004630 042703 000000G   BIC      #RT$OX, R3     ; DO NOT DEFAULT OUTPUT ORG TO INDEXED
30 004634 052703 000000G   BIS      #RT$OS, R3     ; DEFAULT OUTPUT ORG TO SEQUENTIAL
31      ;   Check obsolete /BIN and /SIZE:V switch settings.
32 004640 032767 000000G 000000G 2$:   BIT      #F$VLEN, SFLAGS ; VARIABLE LENGTH?
33 004646 001404          BEQ      3$          ; BR IF NOT
34 004650 052702 000000G   BIS      #RT$VLN, R2    ; SAY BOTH INPUT & OUTPUT TYPES ARE VAR LEN
35 004654 052703 000000G   BIS      #RT$VLN, R3
36 004660          3$:
37      ;   Determine if input and/or output records are to be blank stripped.
38 004660 032702 000000G   4$:   BIT      #RT$RB, R2   ; INPUT MODE BINARY?
39 004664 001002          BNE      5$          ; BR IF YES
40 004666 052702 000000G   BIS      #RT$VLN, R2    ; IS ASCII MODE THEN SET VARIABLE LENGTH FLAG
41 004672 032703 000000G   5$:   BIT      #RT$RB+RT$OR+RT$OX+RT$X, R3 ; BINARY, RELATIVE, ISAM OR "X" TYPE?
42 004676 001002          BNE      6$          ; IF ANY OF ABOVE THEN FIXED LENGTH OUTPUT
43 004700 052703 000000G   BIS      #RT$VLN, R3    ; ALLOW VARIABLE LENGTH OUTPUT RECORDS
44      ;   See if we need to manage CR-LF record terminator characters.
45 004704 032702 000000G   6$:   BIT      #RT$RB+RT$X, R2 ; INPUT BINARY OR FIXED-LENGTH?
46 004710 001010          BNE      7$          ; BR IF EITHER
47 004712 032702 000000G   BIT      #RT$FTN, R2    ; INPUT FROM FORTRAN FILE?
48 004716 001403          BEQ      8$          ; BR IF NOT
49 004720 032702 000000G   BIT      #RT$OR, R2     ; FORTRAN RELATIVE FILE?
50 004724 001002          BNE      7$          ; NO CR-LF ON FORTRAN RELATIVE FILES
51 004726 052702 000000G   8$:   BIS      #RT$CR, R2   ; CR-LF IS RECORD TERMINATOR
52 004732 032703 000000G   7$:   BIT      #RT$RB+RT$X, R3 ; OUTPUT FILE BINARY OR FIXED-LENGTH?
53 004736 001010          BNE      9$          ; BR IF EITHER
54 004740 032703 000000G   BIT      #RT$FTN, R3    ; OUTPUT TO FORTRAN DATA FILE?
55 004744 001403          BEQ      10$         ; BR IF NOT
56 004746 032703 000000G   BIT      #RT$OR, R3     ; FORTRAN RELATIVE FILE?
57 004752 001002          BNE      9$          ; BR IF YES (NO CR-LF)

```

58	004754	052703	000000G	10#:	BIS	#RT#CR,R3	;ADD CR-LF TO RECORD END
59	004760	010267	000000G	9#:	MOV	R2,ITYPE	;SAVE INPUT FILE TYPE FLAGS
60	004764	010367	000000G		MOV	R3,OTYPE	;SAVE OUTPUT FILE TYPE FLAGS

Initialize files

```

1          .SBTTL  Initialize files
2          ;
3          ; Do file initialization
4          ;
5 004770   SETFIL: #LOCK          ; LOCK USR IN MEMORY FOR SPEED
6 004772   004767   002144      CALL   INIFIL          ; DO FILE INITIALIZATION
7          ; Note: At this point R4 contains the sum of the input file sizes.
8          ;
9          ; Delete any temp files that were left around from aborted runs.
10         ;
11 004776   $ERASE #FD1,#SPC2     ; TEMP FILE 1
12 005114   $ERASE #FD1,#SPC3     ; TEMP FILE 2
13         ;
14         ; Reopen 1st input file through FD1.
15         ;
16 005232   012702   000006G     MOV    #FILES+6,R2      ; POINT TO SPC FOR 1ST INPUT FILE
17 005236   $REOPEN #FD1,(R2)    ; REOPEN 1ST INPUT FILE
18 005322   005722           TST    (R2)+          ; POINT TO ENTRY FOR NEXT FILE TO OPEN
19 005324   010267   000000G     MOV    R2,CURSAV      ; SET CURSAV TO POINT TO SPC FOR NEXT FILE
20         ;
21         ; Set up record length information.
22         ;
23 005330   005767   000000G     TST    RECLEN          ; WAS MAPPED RECORD LENGTH SPECIFIED?
24 005334   001003           BNE    2$              ; BR IF YES
25 005336   016767   000000G 000000G  MOV    INRLEN,RECLEN ; DEFAULT TO INPUT RECORD LENGTH IF NOT
26 005344   016701   000000G     2$: MOV    RECLEN,R1      ; GET SORT RECORD LENGTH
27 005350   005201           INC    R1              ; BOUND UP TO WORD BOUNDARY
28 005352   042701   000001      BIC    #1,R1
29 005356   010167   000000G     MOV    R1,RWDLEN     ; SAVE THIS AS SORT RECORD BUFFER SIZE
30         ;
31         ; Determine how much space to allocate for temp files.
32         ; (Note: R4 now contains sum of # blocks in input files)
33         ;
34 005362   032767   000000G 000000G  BIT    #RT#OX,ITYPE  ; IS INPUT FILE ORGANIZATION INDEXED?
35 005370   001430           BEQ    1$              ; BR IF NOT
36         ;
37         ; Indexed organization file.
38         ; Use the following formula to determine temp file size:
39         ; Space = 1.2*(#-input-records*output-record-length)/512. +30.
40         ;
41 005372   016704   000000G     MOV    MXREC1,R4      ; GET # INPUT RECORDS
42 005376   016705   000000G     MOV    MXREC2,R5
43 005402   016700   000000G     MOV    RECLEN,R0     ; GET OUTPUT RECORD LENGTH
44 005406   004767   000000G     CALL  MULTPY        ; MULTIPLY
45 005412   012700   000014      MOV    #12.,R0       ; NOW MULTIPLY BY 12 (1.2*10)
46 005416   004767   000000G     CALL  MULTPY
47 005422   012700   012000      MOV    #5120.,R0     ; DIVIDE BY 512.*10.
48 005426   004767   000000G     CALL  DIVIDE
49 005432   010402           MOV    R4,R2         ; GET LOW-ORDER RESULT VALUE
50 005434   062702   000036      ADD    #30.,R2       ; ADD 30 BLOCKS
51 005440   005505           ADC    R5            ; PROPOGATE CARRY AND CHECK FOR HIGH-ORDER NON-ZERO
52 005442   001433           BEQ    OPNTMP        ; BR IF NEED LESS THAN 65K BLOCKS
53 005444   012702   177777      MOV    #-1,R2        ; ASK FOR 65K BLOCKS
54 005450   000430           BR    OPNTMP
55         ;
56         ; File organization is not indexed.
57         ; Use the following formula to calculate how much temp file space to request:

```

```

58          ; Space = 1.2*(input-file-size)*(output-rec-len/input-rec-len)+30.
59          ;
60 005452 005005          1$: CLR      R5          ; CLEAR HIGH-ORDER WORD
61 005454 012700 000014  MOV     #12.,R0       ; MULTIPLY FILE SIZE BY 12.
62 005460 004767 000000G  CALL   MULTPY
63 005464 016700 000000G  MOV     RECLEN,R0     ; NOW MULTIPLY BY OUTPUT RECORD LENGTH
64 005470 004767 000000G  CALL   MULTPY
65 005474 012700 000012  MOV     #10.,R0       ; NOW DIVIDE BY 10.
66 005500 004767 000000G  CALL   DIVIDE
67 005504 016700 000000G  MOV     INRLEN,R0    ; NOW DIVIDE BY INPUT RECORD LENGTH
68 005510 004767 000000G  CALL   DIVIDE
69 005514 010402  MOV     R4,R2          ; GET LOW-ORDER PART OF RESULT
70 005516 062702 000050  ADD     #40.,R2       ; NEED 30 EXTRA BLOCKS
71 005522 005505  ADC     R5              ; PROPOGATE CARRY & TEST HIGH-ORDER PART
72 005524 001402  BEQ     OPNTMP         ; BR IF AMT NEEDED FITS IN 16-BITS
73 005526 012702 177777  MOV     #-1,R2        ; ASK FOR LARGEST POSSIBLE SPACE
74          ;
75          ; Open temp files.
76          ; At this point, R2 contains total # blocks we want for temp file space.
77          ; Try to get entire temp file space in temp file # 1.
78          ;
79 005532 005067 000000G  OPNTMP: CLR     T2SIZ      ; SAY 2ND TEMP FILE IS NOT USED YET
80 005536  $NEW    #FD2, #SPC2, R2 ; CREATE 1ST TEMP FILE
81 005672 103064  BCC     5$              ; BRANCH IF OPEN WAS OK
82          ; Error occured on the open attempt.
83          ; Try to open to largest available space.
84          $NEW    #FD2, #SPC2, #-1
85 006032 103004  BCC     5$              ; BR IF OPEN SUCCESSFUL
86 006034  9$:    XXX     9              ; COULD NOT OPEN FILE
87 006044 010067 000000G  5$:    MOV     R0,T1SIZ    ; SAVE SIZE OF TEMP FILE 1
88 006050 001771  BEQ     9$              ; ERROR IF NO SPACE IN FILE 1
89 006052 020002  TRYT2: CMP     R0,R2      ; DID WE GET ENOUGH SPACE?
90 006054 103402  BLD     2$              ; BR IF NOT
91 006056 000167 000442  JMP     6$              ; BR IF GOT ENOUGH SPACE
92 006062 160002  2$:    SUB     R0,R2      ; FIND OUT HOW MUCH WE LACK
93          ;
94          ; REQUEST REMAINING SPACE NEEDED FOR TEMP FILE # 2
95          ;
96          $NEW    #FD4, #SPC3, R2 ; TRY TO GET NEEDED SPACE
97 006220 103137  BCC     7$              ; BR IF GOT IT
98 006222  $NEW    #FD4, #SPC3, #-1 ; GET LARGEST AVAILABLE SPACE
99 006360 103402  BCS     1$              ; BR IF COULDN'T GET ANY
100 006362 005700  TST     R0              ; DID WE REALLY GET SOME SPACE?
101 006364 001055  BNE     7$              ; BR IF YES
102 006366  1$:    $PURGE #FD4          ; RELEASE 2ND TEMP FILE
103 006516 005000  CLR     R0              ; SAY ITS SIZE IS ZERO
104 006520 010067 000000G  7$:    MOV     R0,T2SIZ    ; SAVE SIZE OF TEMP FILE 2
105 006524  6$:    $UNLOCK          ; UNLOCK USR

```

```

1          .SBTTL  Generate code to do key comparisons
2          ;-----
3          ;
4          ; GENERATE CODE TO DO KEY COMPARISONS
5          ;
6          ; 1ST STEAL ENOUGH CORE FROM TOP OF MEM FOR COMPARE ROUTINE.
7 006526 005002 CODGEN: CLR R2 ; INDEX TO 1ST KEY
8 006530 016701 000000G MOV TOPMEM,R1 ; POINT TO CURRENT TOP OF MEMORY
9 006534 016203 000000G 1#: MOV KTYPE(R2),R3 ; GET TYPE OF NEXT KEY
10 006540 166301 003566' SUB CODSIZ(R3),R1 ; LEAVE ROOM FOR GENERATED CODE
11 006544 062702 000002G ADD #2,R2 ; GO CHECK NEXT KEY
12 006550 120267 000000G CMPB R2,NUMKY2 ; DONE ALL?
13 006554 103767 BLD 1# ; BR IF NOT
14 006556 162701 000036 SUB #30.,R1 ; LEAVE ROOM FOR EXTRA CODE
15          ;
16          ; STORE CODE FOR WINNER ROUTINE.
17          ;
18 006562 010167 000000G MOV R1, TOPMEM ; SAVE NEW POINTER TO TOP OF MEMORY
19 006566 005721 TST (R1)+ ; POINT TO START OF CODE AREA
20 006570 004767 002634 CALL GENWIN ; GENERATE WINNER CODE
21 006574 010167 000000G MOV R1, COMPAR ; START OF COMPARISON CODE
22 006600 012767 000001 003512' MOV #1,RSTAT ; REGS R1&R2 ARE LOADED FOR ASCENDING KEYS
23          ; GENERATE CODE TO SAVE INITIAL RECORD POINTERS
24 006606 012721 010137 MOV #010137,(R1)+ ; [MOV R1,@#REC1]
25 006612 012721 000000G MOV #REC1,(R1)+ ; [REC1]
26 006616 012721 010237 MOV #010237,(R1)+ ; [MOV R2,@#REC2]
27 006622 012721 000000G MOV #REC2,(R1)+ ; [REC2]
28          ; NOW GENERATE COMPARE CODE FOR EACH KEY.
29 006626 005002 4#: CLR R2 ; INIT KEY INDEX
30 006630 016203 000000G 3#: MOV KTYPE(R2),R3 ; GET KEY TYPE
31 006634 004773 003610' CALL @GENVEC(R3) ; GENERATE CODE FOR THIS TYPE OF KEY
32 006640 062702 000002G ADD #2,R2 ; MOVE ON TO NEXT KEY
33 006644 120267 000000G CMPB R2,NUMKY2 ; DONE ALL?
34 006650 103011 BHIS 2# ; BR IF FINISHED
35          ; SEE IF WE HAVE GOTTEN TOO FAR FOR A BR INSTRUCTION TO WORK
36 006652 010103 MOV R1,R3 ; FIND DISTANCE FROM RETURN CODE
37 006654 166703 003510' SUB RETLOC,R3
38 006660 020327 000346 CMP R3,#230. ; GOTTEN TOO FAR?
39 006664 101761 BLOS 3# ; BR IF OK
40 006666 004767 002536 CALL GENWIN ; GENERATE NEW WINNER ROUTINES
41 006672 000756 BR 3#
42          ; GENERATE [RTS PC] FOR EQUAL KEY COMPARISONS
43 006674 012721 000207 2#: MOV #207,(R1)+ ; [RTS PC]
44          ;
45          ; SEE IF WE NEED TO RESERVE ROOM FOR AN EOF RECORD.
46          ;
47 006700 005767 000000G SEOFR: TST EOFCOL ; IS EOF RECORD EXPECTED?
48 006704 001423 BEQ SETBLK ; BR IF NOT
49 006706 016704 000000G MOV TOPMEM,R4 ; GET CURRENT TOP OF MEMORY
50 006712 166704 000000G SUB RECLN,R4 ; RESERVE ROOM FOR EOF RECORD
51 006716 042704 000001G BIC #1,R4 ; MAKE SURE EVEN WORD
52 006722 010467 000000G MOV R4, EOFBUF ; PLACE EOF RECORD HERE
53 006726 005744 TST -(R4)
54 006730 010467 000000G MOV R4, TOPMEM
55          ; INITIALIZE EOF RECORD TO ALL EOF CHARS
56 006734 005724 TST (R4)+
57 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\$	

```

1          .SBTTL Determine buffer sizes
2          ;-----
3          ;
4          ; DETERMINE BUFFER SIZES
5          ;
6 006754 016704 000000G SETBLK: MOV   BLKFC,R4      ;GET FILE BLOCKING FACTOR
7 006760 001037          BNE   8$          ;BR IF USER SPECIFIED A VALUE
8 006762 016703 000000G     MOV   TOPMEM,R3      ;TOP OF FREE MEMORY
9 006766 012702 000000G     MOV   #P2TOP,R2     ;R2 -> TOP OF PASS 2
10 006772 022702 000000G    CMP   #P1TOP,R2     ;IS TOP OF PASS 1 HIGHER ??
11 006776 101402          BLOS  9$          ;BRANCH IF NOT
12 007000 012702 000000G    MOV   #P1TOP,R2     ;USE HIGHER OF THE TWO
13 007004          9$:
14 007004 160203          SUB   R2,R3          ;GET SPACE AVAILABLE TO PASS 2
15          ; MAKE SURE WE HAVE MIN SIZE
16 007006 020327 005000    CMP   R3,#<5*BUFSIZ> ;GOT ENOUGH FOR SORT?
17 007012 103004          BHS   10$         ;BR IF YES
18 007014          XXX   24          ;NOT ENOUGH MEMORY
19          ; SET BLOCKING FACTOR SO THAT THERE IS ROOM FOR AT LEAST
20          ; 5 BUFFERS.
21 007024 012702 012000 10$:  MOV   #<2*5*BUFSIZ>,R2; TRY BLOCKING FACTOR OF 2
22 007030 012704 000001    MOV   #1,R4
23 007034 020203          7$:  CMP   R2,R3          ;THIS SIZE OK
24 007036 101006          BHI   6$          ;BR IF TOO BIG
25 007040 062702 005000    ADD   #<5*BUFSIZ>,R2 ;INCREASE BLOCKING FACTOR
26 007044 005204          INC   R4
27 007046 020427 000005    CMP   R4,#5         ;GO TO MAX BLOCKING FACTOR OF 5
28 007052 103770          BLO   7$
29          ; NOW GET BUFFER SIZE.
30 007054 010467 000000G 6$:  MOV   R4,BLKFC      ;SAVE BLOCKING FACTOR
31 007060 005001          8$:  CLR   R1
32 007062 062701 001000 1$:  ADD   #BUFSIZ,R1    ;CALCULATE BUFFER SIZE
33 007066 005304          DEC   R4
34 007070 001374          BNE   1$
35 007072 010167 000000G    MOV   R1,0BUFSZ    ;SAVE BUFFER SIZE
36 007076 006201          ASR   R1            ;SAVE WORD SIZE OF BUFFER
37 007100 010167 000000G    MOV   R1,0WCNT
38          ;
39          ; ENTER PHASE 1 OF SORT
40          ;
41 007104 012767 000000G 000000G MOV   #P2TOP,LOWMEM ;ASSUME TOP OF PASS 2 IS LOWEST
42 007112          #GETBOT R2,#P1TOP ;SEE IF WE CAN HAVE TOP OF PASS 1
43 007132 010267 000000G    MOV   R2,LOWMEM   ;LOWMEM = P2TOP OR P1TOP
44 007136 000167 000000G    JMP   START1

```

```

1      .IF      DF RT11
2      .SBTTL  Set up information about file specs
3      ;-----
4      ; INIFIL is called to process the file name portion of the command line.
5      ; It parses the file specs and creates file blocks containing Rad50
6      ; file names, sizes, etc.
7      ;
8      ; Inputs:
9      ;   FILBUF = Asciz CSI file spec (i.e., out,temp=in1,in2,...)
10     ;
11     ; Outputs:
12     ;   FILES = Vector of pointers to file specs. (last pointer is zero).
13     ;   R4 = Sum of input file sizes.
14     ;
15 007142 010146 INIFIL: MOV     R1, -(SP)
16 007144 010246      MOV     R2, -(SP)
17 007146 010346      MOV     R3, -(SP)
18 007150 010546      MOV     R5, -(SP)
19     ;
20     ; Determine proper default file extensions.
21     ;
22 007152 016702 000000G      MOV     ITYPE, R2      ;GET INPUT FILE TYPE FLAGS
23 007156 004767 004244      CALL    GETDFX        ;GET DEFAULT INPUT FILE EXTENSION
24 007162 010267 000064'      MOV     R2, DEFEXT1   ;SET DEFAULT INPUT FILE EXTENSION
25 007166 016702 000000G      MOV     OTYPE, R2      ;GET OUTPUT FILE TYPE FLAGS
26 007172 004767 004230      CALL    GETDFX        ;GET DEFAULT EXTENSION
27 007176 010267 000066'      MOV     R2, DEFEXT+2  ;SET DEFAULT OUTPUT FILE EXTENSION
28     ;
29     ; Use .CSISPC to parse command line.
30     ;
31 007202 010605      MOV     SP, R5          ;PRESERVE STACK POINTER
32 007204      .CSISPC #RSPCS, #DEFEXT, #FILBUF ;PARSE COMMAND LINE
33 007222 103004      BCC     2$            ;BR IF NO ERROR
34 007224      XXX     19          ;INVALID COMMAND LINE
35 007234 010506 2$: MOV     R5, SP          ;RESET STACK POINTER
36 007236 004767 001536      CALL    SVRSTS        ;SAVE (POSSIBLY) ANY RSTS INFORMATION
37     ;
38     ; Set addresses in FILES vector to point to file specifications.
39     ;
40 007242 012701 000000G      MOV     #FILES, R1     ;POINT TO FIRST CELL IN POINTER VECTOR
41 007246 012731 000000G      MOV     #RSPCS, @(R1)+ ;OUTPUT FILE
42 007252 012731 000012G      MOV     #RSPCS+10, @(R1)+;1ST TEMP FILE
43 007256 012731 000024G      MOV     #RSPCS+20, @(R1)+;2ND TEMP FILE
44 007262 012731 000036G      MOV     #RSPCS+30, @(R1)+;1ST INPUT FILE
45 007266 012731 000046G      MOV     #RSPCS+38, @(R1)+;2ND INPUT FILE
46 007272 012731 000056G      MOV     #RSPCS+46, @(R1)+;3RD INPUT FILE
47 007276 012731 000066G      MOV     #RSPCS+54, @(R1)+;4TH INPUT FILE
48 007302 012731 000076G      MOV     #RSPCS+62, @(R1)+;5TH INPUT FILE
49 007306 012731 000106G      MOV     #RSPCS+70, @(R1)+;6TH INPUT FILE
50     ;
51     ; See if this is an in-place sort.
52     ; Specified by omitting the output file spec.
53     ;
54 007312 005777 000000G      TST     @SPC1         ;OUTPUT FILE SPECIFIED?
55 007316 001013      BNE     5$            ;BR IF YES
56 007320 032767 000000G 000000G      BIT     #RT#OX, ITYPE ;IS INPUT FILE ORGANIZATION INDEXED?
57 007326 001404      BEQ     9$            ;BR IF NOT

```

Set up information about file specs

```

58 007300          XXX      37          ;ERROR -- INPLACE SORT NOT LEGAL WITH INDEXED FILE
59 007340 052767 0000000 0000000 9#:  BIS      #F#INPL,SFLAGS ;REMEMBER THIS IS AN IN-PLACE SORT
60                ;
61                ; Set up temp file names.
62                ;
63 007346 016702 0000000          5#:  MOV      SPC2,R2      ;GET ADDRESS OF TEMP FILE BLOCK
64 007352 005712          TST      (R2)          ;DID USER SPECIFY THE TEMP FILE DEVICE?
65 007354 001002          BNE      8#           ;BR IF YES
66 007356 016712 000034'          MOV      R50DK,(R2)    ;DEFAULT TO DK:
67 007362 016703 0000000          8#:  MOV      SPC3,R3      ;POINT TO 2ND TEMP FILE SPEC AREA
68 007366 005713          TST      (R3)          ;WAS DEVICE SPECIFIED FOR 2ND TEMP FILE?
69 007370 001001          BNE      1#           ;BR IF YES
70 007372 011213          MOV      (R2),(R3)    ;USE SAME DEVICE AS FOR 1ST TEMP FILE
71 007374 016762 000036' 0000002 1#:  MOV      R50TN0,2(R2) ;SET DEFAULT TEMP FILE NAME
72 007402 016762 000040' 0000004          MOV      R50TN1,4(R2)
73 007410 016763 000036' 0000002          MOV      R50TN0,2(R3) ;SET NAME FOR 2ND TEMP FILE
74 007416 016763 000042' 0000004          MOV      R50TN2,4(R3)
75                ;
76                ; If running under TSX, generate extension of the form XXz where "z"
77                ; is a function of the line number.
78                ;
79                ; If running under RSTS, generate extension of the form XXz where "z"
80                ; is a function of the job number.
81                ;
82 007424 105767 0000000          TSTB     RSTS          ;RUNNING UNDER RSTS?
83 007430 001404          BEQ      6#           ;BR IF NOT
84 007432 113700 000404          MOVB     @#RSTSJOB,R0 ;GET RSTS JOB # TIMES 2
85 007436 006200          ASR      R0           ; TIMES 1
86 007440 000403          BR       7#           ;
87 007442 116700 0000000          6#:  MOVB     TSXFL,R0      ;GET TSX LINE NUMBER
88 007446 001407          BEQ      3#           ;BR IF RUNNING DIRECTLY UNDER RT11
89 007450 066700 000044'          7#:  ADD      R50XX,R0      ;FORM EXTENSION NAME
90 007454 010062 0000006          MOV      R0,6(R2)    ;SET AS EXTENSION OF TEMP FILE 1
91 007460 010063 0000006          MOV      R0,6(R3)    ;AND TEMP FILE 2
92 007464 000406          BR       4#           ;
93                ; Running directly under RT-11 -- Use "TMP" as temp file extension.
94 007466 016762 000046' 0000006 3#:  MOV      R50TMP,6(R2) ;SET TMP AS TEMP FILE EXTENSIONS
95 007474 016763 000046' 0000006          MOV      R50TMP,6(R3)
96                ;
97                ; Load device handlers.
98                ;
99 007502 016705 0000000          4#:  MOV      TOPMEM,R5     ;GET ADDRESS OF TOP OF FREE MEMORY
100 007506 012702 0000000          MOV      #FILES,R2    ;POINT TO FILE DESCRIPTOR POINTER VECTOR
101 007512 013201          10#:  MOV      @(R2)+,R1    ;GET ADDRESS OF RT-11 FILE SPEC
102 007514 004767 003574          CALL     LOADIT       ;LOAD DEVICE HANDLER
103 007520 020227 0000220          CMP      R2,#FILES+22 ;HAVE WE DONE ALL?
104 007524 103772          BLO     10#          ;BR IF NOT
105 007526 005745          TST      -(R5)        ;
106 007530 010567 0000000          MOV      R5,TOPMEM   ;SAVE NEW TOP OF MEMORY ADDRESS
107                ;
108                ; If this is a COBOL-Plus Indexed organization file, read in home block
109                ; and set up information about data files.
110                ;
111 007534 032767 0000000 0000000          BIT      #RT#OX,ITYPE ;IS THIS A COBOL ISAM FILE?
112 007542 001402          BEQ      OPNDA1      ;BR IF NOT
113 007544 004767 000252          CALL     INIX        ;DO INITIALIZATION FOR ISAM FILE
114                ;

```

```

115 ; Open each file, Add up number of blocks and do .SAVESTATUS
116 ;
117 007550 012702 0000060 OPNDAT: MOV #FILES+6, R2 ; POINT TO SPEC FOR 1ST INPUT FILE
118 007554 012705 0000000 MOV #SAVBUF, R5 ; DO SAVESTATUS INTO THIS AREA
119 007560 005004 CLR R4 ; ACCUMULATE FILE SIZES IN R4
120 007562 011201 3#: MOV @R2, R1 ; GET POINTER TO SPC FOR FILE
121 007564 001511 BEQ 1# ; BR IF REACHED END
122 007566 005731 TST @(R1)+ ; IS FILE SPEC NON-ZERO?
123 007570 001507 BEQ 1# ; IF ZERO THEN FILE WAS NOT SPECIFIED
124 007572 $OLD #FD1, (R2) ; TRY TO OPEN INPUT FILE
125 007714 103004 BCC 2# ; BR IF OPEN WAS SUCCESSFUL
126 007716 XXX 0 ; CANNOT OPEN INPUT FILE
127 007726 060004 2#: ADD R0, R4 ; ACCUMULATE FILE SIZES
128 007730 010532 MOV R5, @(R2)+ ; MAKE SPC POINT TO SAVESTATUS AREA
129 007732 $SAVEST #FD1 ; SAVE FILE STATUS
130 010002 062705 000012 ADD #12, R5 ; INCREMENT TO THE NEXT SAVESTATUS AREA
131 010006 000665 BR 3# ; GO OPEN REST OF INPUT FILES
132 ;
133 ; Finished: Return accumulated file size in R4.
134 ;
135 010010 012605 1#: MOV (SP)+, R5
136 010012 012603 MOV (SP)+, R3
137 010014 012602 MOV (SP)+, R2
138 010016 012601 MOV (SP)+, R1
139 010020 000207 RETURN
  
```

Set up information about file specs

```

1          ; -----
2          ; INIX is called to do initialization for COBOL-Plus Indexed organization
3          ; files. It reads the file home block and sets up the RAD50 file specs
4          ; for the data files in the RSPCS area.
5          ;
6          ; Inputs:
7          ;   TOPMEM = Top of free memory address.
8          ;
9          ; Outputs:
10         ;   TOPMEM = Set to new top of free memory address.
11         ;   MXREC1-MXREC2 = Max # of records to sort.
12         ;
13 010022 010146 INIX:  MOV    R1, -(SP)
14 010024 010246      MOV    R2, -(SP)
15 010026 010346      MOV    R3, -(SP)
16 010030 010446      MOV    R4, -(SP)
17 010032 010546      MOV    R5, -(SP)
18         ;
19         ; Make sure he only specified one input file.
20         ;
21 010034 005777 0000000  TST    @SPC5          ; IS THERE A SECOND INPUT FILE?
22 010040 001404      BEQ    10$          ; BR IF NOT
23 010042          XXX    38          ; ERROR IF YES
24         ;
25         ; Open ISAM index file and read in home block.
26         ;
27 010052          10$:  $OLD    #FD1, #SPC4          ; OPEN FIRST INPUT FILE
28 010176 103004      BCC    1$          ; BR IF OPEN OK
29 010200          XXX    8          ; CAN'T OPEN INPUT FILE
30 010210          1$:  $READ   #FD1, #GENBUF, #256, #0 ; READ HOME BLOCK
31 010370 103447      BCS    2$          ; BR IF READ ERROR
32 010372          $WAIT  #FD1          ; WAIT FOR READ TO FINISH
33 010506 103004      BCC    3$          ; BR IF READ OK
34 010510          2$:  XXX    14          ; READ ERROR
35 010520          3$:  $KEEP  #FD1          ; CLOSE INDEX FILE
36         ;
37         ; Extract information from home block.
38         ;
39         ; Set number of records in file.
40         ;
41 010566 012701 002432'  MOV    #GENBUF, R1          ; POINT TO HOME BLOCK BUFFER
42 010572 016102 000032  MOV    HB$RIU(R1), R2        ; GET LOW-ORDER NUMBER OF RECORDS IN FILE
43 010576 016103 000034  MOV    HB$RIU+2(R1), R3      ; GET HIGH-ORDER NUMBER OF RECORDS IN FILE
44 010602 020367 000000G  CMP    R3, MXREC2          ; COMPARE HIGH-ORDER WITH PREVIOUSLY SPECIFIED MAX
45 010606 103404      BLO    4$          ; BR IF WE SHOULD USE ISAM # RECORDS
46 010610 101007      BHI    5$          ; BR IF PREVIOUS MAX IS SMALLER
47 010612 020267 000000G  CMP    R2, MXREC1          ; COMPARE LOW-ORDER
48 010616 103004      BHIS   5$          ; BR IF PREVIOUS VALUE IS ONE TO USE
49 010620 010267 000000G  4$:  MOV    R2, MXREC1          ; SET # RECORDS TO BE READ
50 010624 010367 000000G  MOV    R3, MXREC2
51         ;
52         ; Set up information about record length.
53         ;
54 010630 016103 000004  5$:  MOV    HB$RSZ(R1), R3      ; GET THE LOGICAL RECORD LENGTH (BYTES)
55 010634 010367 000000G  MOV    R3, ISMARS          ; THIS IS FULL RECORD LENGTH LESS LEADING CONTROL WORD
56 010640 116100 000001  MOVB   HB$DUP(R1), R0       ; GET # OF KEYS THAT ALLOW DUPLICATES
57 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      R5OCDO,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

```

```

1          .SBTTL  RSTS oriented subroutines
2          ;
3          ; SVRSTS
4          ;
5          ; Save the RSTS oriented file parameters after a .CSISPC file
6          ; decode.
7          ;
8          ; When running under the RT11 run-time of the RSTS operating
9          ; system, there are a number of RSTS oriented file parameters
10         ; that can be present in a file specification which have no
11         ; meaning for RT11.  These include the Project-Programmer Number
12         ; (PPN), Protection Code, Mode, and Clustersize.  Fortunately
13         ; (and not documented outside of DEC), the RT11 run-time saves
14         ; these parameters in a job's "Read/Write Stack" after a .CSISPC
15         ; decode is performed.  This read/write stack is the job-dependent
16         ; data area that the RT11 run-time maintains for each job.  It
17         ; is created in the high part of the user memory allocated to
18         ; the running job, is about 3600 bytes long (for RSTS V06C),
19         ; and is at all times pointed at by the low core RMON word
20         ; (location 54).  The beginning of the saved parameters is
21         ; CSPSDF bytes into the R/W stack with each channel's information
22         ; occupying 20 bytes.
23         ;
24         ; Also present in the R/W stack are four "one shot" cells
25         ; which are active for one file oriented operation (.LOOKUP,
26         ; .ENTER, etc.).  These four words are for the PPN, protection
27         ; code, mode, and clustersize, respectively, and reside at
28         ; the immediate beginning of the R/W stack.  Since the second
29         ; through fifth words of the saved .CSISPC parameters (see
30         ; above) are also these four items, these items can be saved
31         ; off after a .CSISPC call and then loaded into the one-shot
32         ; area immediately preceding a file operation.
33         ;
34         ; This routine does the saving of the RSTS-oriented paramters,
35         ; while the SETNRT routine processes the loading of the one-
36         ; shots.
37         ;
38 011000 105767 000000G  SVRSTS: TSTB   RSTS      ;We running under RSTS?
39 011004 001436          BEQ    10$      ; => no
40 011006 010046          MOV    R0, -(SP)   ;Get some work registers
41 011010 010146          MOV    R1, -(SP)
42 011012 010246          MOV    R2, -(SP)
43 011014 010346          MOV    R3, -(SP)
44 011016 010446          MOV    R4, -(SP)
45
46 011020 013700 000054   MOV    @#RMON, R0    ;Point to the initial block
47 011024 062700 000400   ADD    #CSPCDF+2, R0
48 011030 012704 000000G  MOV    #FILES, R4    ;Point to vector of file descriptor addresses
49 011034 012702 000011   MOV    #9, R2        ;And set a counter
50
51 011040 012401          5$:   MOV    (R4)+, R1    ;Get address of next file descriptor block
52 011042 062701 000002   ADD    #2, R1        ;Point to area where we save RSTS data
53 011046 010003          MOV    R0, R3        ;Get a move from pointer
54 011050 012321          MOV    (R3)+, (R1)+   ;Save PPN
55 011052 012321          MOV    (R3)+, (R1)+   ; Protection code
56 011054 012321          MOV    (R3)+, (R1)+   ; Mode
57 011056 012321          MOV    (R3)+, (R1)+   ; and Clustersize

```

```
58 011060 062700 000020      ADD    #20,R0      ;Skip to the next block
59 011064 005302      DEC    R2          ;And continue until done
60 011066 001364      BNE    5$         ;Loop for all files
61
62 011070 012604      MOV    (SP)+,R4
63 011072 012603      MOV    (SP)+,R3      ;Restore the work registers
64 011074 012602      MOV    (SP)+,R2
65 011076 012601      MOV    (SP)+,R1
66 011100 012600      MOV    (SP)+,R0
67
68 011102 000207      10$:  RETURN      ;Return to the caller
69                      . ENDC ;RT-11
```

```

1          .SBTTL  CODE GENERATOR SUBROUTINES
2          ;
3          ;-----
4          ;  ASCII KEY
5          ;
6 011104 016203 000000G  GNASCII: MOV      KLEN(R2),R3      ;GET KEY LENGTH
7 011110 020327 000002      CMP      R3,#2          ;SEE IF SHORT KEY
8 011114 101003      BHI      2$          ;BR IF LONG
9 011116 103420      BLO      ONEASC      ;BR IF SINGLE ASCII CHARACTER
10 011120 000167 000062      JMP      TWOASC      ;BR IF 2-CHAR KEY
11          ;  GENERATE CODE FOR GENERAL ASCII STRING COMPARISON
12 011124 004767 000424 2$:  CALL      GENKPT      ;GEN CODE TO GET KEY POINTERS
13 011130 012721 012703      MOV      #012703,(R1)+      ;[MOV #LEN,R3]
14 011134 010321      MOV      R3,(R1)+      ;[#LEN]
15 011136 012721 122122      MOV      #122122,(R1)+      ;[CMPB (R1)+,(R2)+]
16 011142 004767 000446 4$:  CALL      GENBRU      ;GEN [BNE RETURN]
17 011146 012721 005303      MOV      #5303,(R1)+      ;[DEC R3]
18 011152 012721 001374      MOV      #1374,(R1)+      ;[BNE .-8.]
19 011156 000207      RETURN
20          ;
21          ;-----
22          ;  GENERATE CODE FOR SINGLE ASCII CHARACTER
23          ;
24 011160 004767 000262  ONEASC: CALL      GENRPT      ;GET RECORD POINTERS
25 011164 012721 126162      MOV      #126162,(R1)+      ;[CMPB POS(R1),POS(R2)]
26 011170 016203 000000G      MOV      KSTRT(R2),R3      ;GET POS+1
27 011174 010321      MOV      R3,(R1)+      ;[POS]
28 011176 010321      MOV      R3,(R1)+      ;[POS]
29 011200 004767 000410      CALL      GENBRU      ;GEN BNE
30 011204 000207      RETURN
31          ;
32          ;-----
33          ;  GEN CODE FOR TWO CHARACTER ASCII KEYS
34          ;
35 011206 004767 000234  TWOASC: CALL      GENRPT      ;GET RECORD POINTERS
36 011212 012721 126162      MOV      #126162,(R1)+      ;[CMPB POS(R1),POS(R2)]
37 011216 016203 000000G      MOV      KSTRT(R2),R3      ;GET COLUMN #
38 011222 010321      MOV      R3,(R1)+      ;[POS]
39 011224 010321      MOV      R3,(R1)+      ;[POS]
40 011226 004767 000362      CALL      GENBRU      ;GEN [BNE RETURN]
41 011232 012721 126162      MOV      #126162,(R1)+      ;[CMPB POS+1(R1),POS+1(R2)]
42 011236 005203      INC      R3          ;GET POS+1
43 011240 010321      MOV      R3,(R1)+      ;[POS+1]
44 011242 010321      MOV      R3,(R1)+      ;[POS+1]
45 011244 004767 000344      CALL      GENBRU      ;GEN [BNE RETURN]
46 011250 000207      RETURN
47          ;
48          ;-----
49          ;  GENERATE CODE FOR UNSIGNED 16-BIT BINARY INTEGER.
50          ;
51 011252 004767 000170  GNUSI:  CALL      GENRPT      ;GET RECORD POINTERS
52 011256 012721 026162      MOV      #026162,(R1)+      ;[CMP POS(R1),POS(R2)]
53 011262 016203 000000G      MOV      KSTRT(R2),R3      ;GET STARTING COL #
54 011266 010321      MOV      R3,(R1)+      ;[POS]
55 011270 010321      MOV      R3,(R1)+      ;[POS]
56 011272 004767 000316      CALL      GENBRU      ;GEN [BNE RETURN]
57 011276 052767 000000G 000000G  BIS      #F$BIN,SFLAGS      ;REMEMBER THIS IS A BINARY RECORD

```

```

58 011304 000207                RETURN
59                               ;-----
60                               ;
61                               ;   Comp item stored with least significant word first.
62                               ;
63 011306 012704 000000G  GNCMP:  MOV     #KYCOMP,R4      ; COMPARISON ROUTINE
64 011312 052767 000000G 000000G  BIS     #F#BIN,SFLAGS ; REMEMBER RECORD HAS COMP FIELD
65 011320 000425                BR      GNCALL
66                               ;
67                               ;   Leading separate sign.
68                               ;
69 011322 012704 000000G  GNLSS:  MOV     #KYLSS,R4      ; COMPARISON ROUTINE
70 011326 000422                BR      GNCALL
71                               ;
72                               ;   Trailing separate sign.
73                               ;
74 011330 012704 000000G  GNTSS:  MOV     #KYTSS,R4      ; COMPARISON ROUTINE
75 011334 000417                BR      GNCALL
76                               ;
77                               ;   Date fields in the form (mddy).
78                               ;
79 011336 012704 000000G  GNDATE: MOV     #KYDATE,R4      ; COMPARISON ROUTINE
80 011342 000414                BR      GNCALL
81                               ;
82                               ;   Binary values stored with the most significant word first.
83                               ;
84 011344 012704 000000G  GNBIN:  MOV     #KYBIN,R4      ; COMPARISON ROUTINE
85 011350 052767 000000G 000000G  BIS     #F#BIN,SFLAGS ; REMEMBER BINARY FIELD IN RECORD
86 011356 000406                BR      GNCALL
87                               ;
88                               ;   Signed numeric display (trailing non-separate sign).
89                               ;
90 011360 012704 000000G  GNDIS:  MOV     #KYDIS,R4      ; COMPARISON ROUTINE
91 011364 000403                BR      GNCALL
92                               ;
93                               ;   Ascii characters with lower case translated to upper case.
94                               ;
95 011366 012704 000000G  GNUCA:  MOV     #KYUCA,R4      ; COMPARISON ROUTINE
96 011372 000400                BR      GNCALL
97                               ;
98                               ;-----
99                               ;   Gen code to call a subroutine (whose address is in R4) to do a key
100                              ;   comparison.
101                              ;   On entry to the subroutine the following registers will be set up:
102                              ;     R1 = Address of key field 1.
103                              ;     R2 = Address of key field 2.
104                              ;     R3 = # of bytes in key fields.
105                              ;
106 011374 004767 000154  GNCALL: CALL    GENKPT      ; GEN CODE TO LOAD R1 & R2 WITH KEY POINTERS
107 011400 012721 012703  MOV     #012703,(R1)+ ; [MOV # bytes,R3]
108 011404 016221 000000G  MOV     KLEN(R2),(R1)+ ; [# bytes]
109 011410 012721 004737  MOV     #4737,(R1)+   ; [CALL @#routine]
110 011414 010421  MOV     R4,(R1)+     ; [Routine]
111 011416 004767 000172  CALL    GENBRU      ; [BNE return]
112 011422 105067 003512' CLR    RSTAT      ; REGISTERS ARE MESSED UP
113 011426 000207                RETURN
114                               ;

```

```

115 ; -----
116 ; GEN CODE TO SIGNAL WHICH RECORD WAS THE WINNER.
117 ;
118 011430 012721 000401 GENWIN: MOV #40J, (R1)+ ; [BR .+2]
119 011434 010167 003510' MOV R1, RETLOC ; SAVE LOCATION OF RTS PC INSTRUCTION
120 011440 012721 000207 MOV #207, (R1)+ ; [RTS PC]
121 011444 000207 RETURN
122 ;
123 ; -----
124 ; GEN CODE TO LOAD R1 & R2 WITH POINTERS TO START OF RECORDS
125 ; IF THEY ARE NOT ALREADY LOADED CORRECTLY.
126 ;
127 011446 005762 000000G GENRPT: TST KAD(R2) ; NEEDED FOR ASCENDING OR DESCENDING?
128 011452 001020 BNE 1# ; BR IF DESCENDING
129 ; GEN OF ASCENDING KEY
130 011454 026727 003512' 000001 CMP RSTAT, #1 ; ARE THEY LOADED CORRECTLY NOW?
131 011462 001413 BEQ 2# ; BR IF YES
132 011464 012767 000001 003512' MOV #1, RSTAT ; SAY THEY ARE LOADED CORRECTLY
133 011472 012721 013701 MOV #013701, (R1)+ ; [MOV @#REC1, R1]
134 011476 012721 000000G MOV #REC1, (R1)+ ; [REC1]
135 011502 012721 013702 MOV #013702, (R1)+ ; [MOV @#REC2, R2]
136 011506 012721 000000G MOV #REC2, (R1)+ ; [REC2]
137 011512 000207 2#: RETURN
138 ; GEN CODE FOR DESCENDING KEYS
139 ; (REVERSE ORDER OF RECORDS)
140 011514 026727 003512' 000002 1#: CMP RSTAT, #2 ; ARE REGS ALREADY LOADED?
141 011522 001413 BEQ 3# ; BR IF YES
142 011524 012767 000002 003512' MOV #2, RSTAT ; SAY THEY ARE LOADED
143 011532 012721 013701 MOV #013701, (R1)+ ; [MOV @#REC2, R1]
144 011536 012721 000000G MOV #REC2, (R1)+ ; [REC2]
145 011542 012721 013702 MOV #013702, (R1)+ ; [MOV @#REC1, R2]
146 011546 012721 000000G MOV #REC1, (R1)+ ; [REC1]
147 011552 000207 3#: RETURN
148 ;
149 ; -----
150 ; GEN CODE TO LOAD R1 & R2 WITH POINTERS TO START OF KEYS
151 ;
152 011554 010546 GENKPT: MOV R5, -(SP)
153 011556 004767 177664 CALL GENRPT ; LOAD R1 & R2 WITH RECORD POINTERS
154 011562 016205 000000G MOV KSTRT(R2), R5 ; GET STARTING COL #
155 011566 001406 BEQ 9# ; BR IF AT FRONT OF RECORD
156 011570 012721 062701 MOV #062701, (R1)+ ; [ADD #POS, R1]
157 011574 010521 MOV R5, (R1)+ ; [POS]
158 011576 012721 062702 MOV #062702, (R1)+ ; [ADD #POS, R2]
159 011602 010521 MOV R5, (R1)+ ; [POS]
160 011604 005067 003512' 9#: CLR RSTAT ; SAY R1-R2 ARE MESSED UP
161 011610 012605 MOV (SP)+, R5
162 011612 000207 RETURN
163 ;
164 ; -----
165 ; Generate code for BNE following a test which leaves
166 ; the condition codes correctly set for BGT/BLT/BEQ.
167 ;
168 011614 010346 GENBRU: MOV R3, -(SP)
169 011616 012721 001000 MOV #001000, (R1)+ ; [BNE]
170 011622 010103 MOV R1, R3
171 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	BIC	#177400,R3	;LEAVE ONLY DISPLACEMENT FIELD
175	011640	050361	BIS	R3,-2(R1)	;STORE DISPLACEMENT INTO BNE INSTRUCTION
176	011644	012603	MOV	(SP)+,R3	
177	011646	000207	RETURN		

1
2
3
4
5
6
7
8
9
10
11
12
13
14 011650 010546
15 011652 012705 003542'
16 011656 105715
17 011660 001004
18 011662
19 011672 120025
20 011674 001370
21 011676 116500 000011
22 011702 010067 003504'
23
24 011706 005067 003506'
25 011712 112100
26 011714 120027 000101
27 011720 001421
28 011722 120027 000104
29 011726 001003
30 011730 005267 003506'
31 011734 000413
32
33 011736 120027 000060
34 011742 103403
35 011744 120027 000071
36 011750 101404
37 011752
38 011762 005301
39
40 011764 004767 001122
41 011770 005305
42 011772 002004
43 011774
44 012004 010567 003500'
45
46 012010 122127 000056
47 012014 001005
48 012016 004767 001070
49 012022 010567 003502'
50 012026 000403
51 012030 005067 003502'
52 012034 005301
53
54
55
56 012036 016700 003504'
57 012042 020027 000006

```

.SBTTL SUBROUTINES
-----
; ACRFLD is called to accrue a sort field descriptor of the
; form tas.l where
;   t = type code (returned as FT$xxx in FLDTYP)
;   a = Ascending/Descending (FLDORD: 0==>ascending, 1==>descending)
;   s = starting column number (value-1 returned in FLDPOS)
;   l = field length (returned in FLDSIZ)
; When called, R0 must contain the 1st letter of the field descriptor
; and R1 must point to the 2nd letter.
; On return, R1 points to the delimiter hit.
;
ACRFLD: MOV     R5, -(SP)
        MOV     #KEYTC, R5          ; POINT TO DATA TYPE LETTER TABLE
2$:     TSTB    (R5)                ; REACHED END OF TABLE?
        BNE     5$                  ; BR IF NOT
        XXX     3                    ; INVALID DATA TYPE LETTER
5$:     CMPB    R0, (R5)+           ; LOOK FOR CHAR IN TABLE
        BNE     2$                  ; LOOP TILL FOUND
        MOVB   KEYCOD-KEYTC-1(R5), R0 ; GET FT$xxx TYPE CODE
        MOV     R0, FLDTYP          ; SAVE FIELD TYPE CODE
; Get second character which specifies Ascending/Descending
        CLR     FLDORD              ; ASSUME ASCENDING
        MOVB   (R1)+, R0
        CMPB   R0, #'A              ; ASCENDING?
        BEQ    7$                  ; BR IF YES
        CMPB   R0, #'D              ; DESCENDING?
        BNE     4$                  ; BR IF NOT
        INC    FLDORD              ; SET DESCENDING ORDER FLAG
        BR     7$
; If A/D was left out default to A.
4$:     CMPB   R0, #'0              ; HIT A DIGIT?
        BLD    8$                  ; BR IF NOT
        CMPB   R0, #'9
        BLOS   6$                  ; HIT START OF COLUMN # FIELD
8$:     XXX     6                    ; NOT A OR D
6$:     DEC    R1                  ; POINT TO START OF NUMBER
; Accrue field starting column number.
7$:     CALL   ACRDEC              ; ACCRUE STARTING COLUMN NUMBER
        DEC    R5                  ; CONVERT TO BYTE NUMBER
        BGE    12$                 ; BR IF VALUE VALID
        XXX    5                    ; INVALID STARTING COLUMN NUMBER
12$:    MOV    R5, FLDPOS           ; SAVE STARTING BYTE NUMBER
; Accrue field length if specified
        CMPB   (R1)+, #'.'          ; PERIOD SHOULD BE THE DELIMITER
        BNE    10$                 ; BR IF SIZE NOT SPECIFIED
        CALL   ACRDEC              ; ACCRUE SIZE VALUE
        MOV    R5, FLDSIZ          ; SAVE SIZE VALUE
        BR     AFCKV
10$:    CLR    FLDSIZ              ; SAY SIZE WAS ZERO
        DEC    R1                  ; POINT TO START OF NEXT SWITCH
;
; Check validity of field spec.
;
AFCKV:  MOV    FLDTYP, R0          ; GET FIELD DATA TYPE
        CMP    R0, #FT$T          ; DATE?

```

```

58 012046 001004          BNE      1$          ;BR IF NOT
59 012050 012767 000006 003502'  MOV      #6,FLDSIZ  ;SAY SIZE IS 6 CHARS.
60 012056 000450          BR       10$
61 012060 020027 000002          1$:  CMP      R0,#F1*B    ;B--SIGNED BINARY?
62 012064 001403          BEQ      2$          ;BR IF YES
63 012066 020027 000004          CMP      R0,#F1*U    ;U--UNSIGNED BINARY?
64 012072 001007          BNE      3$          ;BR IF NOT
65 012074 016700 003502'          2$:  MOV      FLDSIZ,R0    ;GET SPECIFIED SIZE
66 012100 001015          BNE      4$          ;BR IF IT WAS SPECIFIED
67 012102 012767 000002 003502'  MOV      #2,FLDSIZ  ;DEFAULT TO 2 BYTES
68 012110 000411          BR       4$
69 012112 020027 000010          3$:  CMP      R0,#F1*F    ;F--FLOATING?
70 012116 001030          BNE      10$         ;BR IF NOT
71 012120 016700 003502'          MOV      FLDSIZ,R0    ;GET SPECIFIED SIZE
72 012124 001003          BNE      4$          ;BR IF SIZE WAS SPECIFIED
73 012126 012767 000004 003502'  MOV      #4,FLDSIZ  ;DEFAULT TO 4 BYTES
74                                ; Common code for computational data types.
75 012134 032700 000001          4$:  BIT      #1,R0      ;IS SIZE EVEN?
76 012140 001404          BEQ      5$          ;BR IF YES
77 012142          9$:  XXX      4          ;INVALID FIELD LENGTH
78 012152 020027 000010          5$:  CMP      R0,#8      ;NOT TOO LONG?
79 012156 101371          BHI      7$          ;BR IF TOO LONG
80 012160 032767 000001 003500'  BIT      #1,FLDPOS   ;IS BYTE POSITION EVEN?
81 012166 001404          BEQ      10$         ;BR IF YES
82 012170          XXX      5          ;INVALID STARTING COLUMN
83                                ; End of accruing field.
84 012200 012605          10$:  MOV      (SP)+,R5
85 012202 000207          RETURN

```

```

1 ; -----
2 ; SLLIT is called to accrue a literal value which is part of a record
3 ; selection switch.
4 ; When called, R2 must point to the current select control block,
5 ; R0 must contain the 1st character of the literal and R1 must point
6 ; to the 2nd character of the literal.
7 ;
8 012204 010346 SLLIT: MOV R3, -(SP)
9 012206 010446 MOV R4, -(SP)
10 012210 010546 MOV R5, -(SP)
11 ; Reserve room for literal below current select control block.
12 012212 016703 000000G MOV TOPMEM, R3 ; GET FREE MEMORY POINTER
13 012216 016204 000010 MOV S#SIZ1(R2), R4 ; GET SIZE OF FIELD
14 012222 160403 SUB R4, R3 ; RESERVE ROOM FOR LITERAL
15 012224 042703 000001 BIC #1, R3 ; FORCE ADDRESS EVEN
16 012230 010362 000012 MOV R3, S#POS2(R2) ; SAVE ADDRESS OF START OF LITERAL AREA
17 012234 010367 000000G MOV R3, TOPMEM ; UPDATE FREE MEMORY POINTER
18 012240 004767 000626 CALL QUOTCK ; IS THIS A QUOTED STRING?
19 012244 001022 BNE SLLNUM ; BR IF NOT (MUST BE NUMERIC LITERAL)
20 ;
21 ; Accrue alphanumeric literal string.
22 ;
23 012246 112100 SLLALP: MOVB (R1)+, R0 ; GET NEXT CHAR FROM STRING
24 012250 001410 BEQ 3$ ; BR IF HIT END OF COMMAND STRING
25 012252 120067 003526' CMPB R0, QUOTE ; IS THIS CHAR THE CLOSING QUOTE?
26 012256 001405 BEQ 3$ ; BR IF YES
27 012260 005704 TST R4 ; HAVE WE MOVED ENOUGH CHARS?
28 012262 001771 BEQ SLLALP ; BR IF YES
29 012264 110023 MOVB R0, (R3)+ ; MOVE CHAR TO HOLDING AREA
30 012266 005304 DEC R4 ; COUNT CHARS MOVES
31 012270 000766 BR SLLALP ; GO MOVE MORE
32 ; Reached end of string.
33 ; See if we need to blank fill the end.
34 012272 005704 3$: TST R4 ; DID WE COMPLETELY FILL FIELD?
35 012274 001404 BEQ 4$ ; BR IF YES
36 012276 112723 000040 MOVB #' , (R3)+ ; BLANK FILL REST OF FIELD
37 012302 005304 DEC R4
38 012304 000772 BR 3$
39 012306 000167 000550 4$: JMP SLLEND
40 ;
41 ; Accrue a numeric literal.
42 ;
43 ; First move characters in literal to temp holding buffer.
44 012312 105067 003527' SLLNUM: CLRB NEGVAL ; ASSUME VALUE IS NOT TO BE NEGATED
45 012316 105067 003530' CLRB DECP ; ASSUME VALUE HAS NO DEC PT
46 012322 012703 001770' MOV #SLBUF, R3 ; POINT TO TEMP HOLDING BUFFER
47 012326 120027 000060 6$: CMPB R0, #'0 ; DETERMINE WHAT TYPE OF CHAR THIS IS
48 012332 103017 BHIS 1$ ; BR IF POSSIBLY DIGIT
49 012334 120027 000055 CMPB R0, #'-' ; MINUS SIGN?
50 012340 001003 BNE 2$ ; BR IF NOT
51 012342 105267 003527' INCB NEGVAL ; REMEMBER TO NEGATE VALUE
52 012346 000415 BR 3$
53 012350 120027 000053 2$: CMPB R0, #'+' ; PLUS SIGN?
54 012354 001412 BEQ 3$ ; IGNORE IF YES
55 012356 120027 000056 CMPB R0, #'.' ; DEC POINT?
56 012362 001011 BNE 4$ ; BR IF NOT
57 012364 105267 003530' INCB DECP ; REMEMBER WE HIT DEC PT

```

```

58 012370 000403          BR      5$
59 012372 120027 000071  1$:    CMPB   RO,#'9      ; IS CHAR A DIGIT?
60 012376 101003          BHI     4$              ; BR IF NOT (HIT END OF LITERAL)
61 012400 110023          5$:    MOVB   RO,(R3)+      ; MOVE CHAR TO HOLDING BUFFER
62 012402 112100          3$:    MOVB   (R1)+,RO      ; 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),RO    ; GET USAGE TYPE CODE
79 012444 020027 000002  CMP    RO,#FT$B       ; COMP?
80 012450 001520          BEQ     SLCOMP          ; BR IF YES
81 012452 020027 000004  CMP    RO,#FT$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)+,RO      ; GET NEXT DIGIT FROM STRING
101 012536 001402          BEQ     5$              ; BR IF END HIT
102 012540 110025          MOVB   RO,(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),RO    ; GET SIGN TYPE CODE
109 012550 020027 000012  CMP    RO,#FT$D       ; TRAILING NON-SEPARATE SIGN?
110 012554 001423          BEQ     SLTNS          ; BR IF YES
111 012556 020027 000016  CMP    RO,#FT$I       ; LEADING NON-SEPARATE SIGN?
112 012562 001430          BEQ     SLTSS          ; BR IF YES
113 012564 020027 000020  CMP    RO,#FT$J       ; TRAILING NON-SEPARATE SIGN?
114 012570 001437          BEQ     SLTSS          ; BR IF YES

```

```

115 012572 020027 000000          CMP      RO,#FT#C          ; CHARACTER (UNSIGNED NUMERIC DISPLAY)?
116 012576 001412          BEQ      SLTNS            ; BR IF YES
117 012600 020027 000014          CMP      RO,#FT#A          ; UPPER CASE CHARACTER?
118 012604 001407          BEQ      SLTNS            ; BR IF YES
119 012606 020027 000006          CMP      RO,#FT#T          ; DATE?
120 012612 001404          BEQ      SLTNS            ; 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),RO         ; GET LAST DIGIT OF VALUE
128 012634 062700 000100          ADD      #64.,RO         ; MARK AS NEGATIVE
129 012640 110015          MOVB     RO,(R5)         ; PUT BACK INTO NUMBER
130 012642 000506          BR       SLNEND
131
132          ; Leading separate sign.
133
134 012644 012700 000053        SLTSS:  MOV      #'+,RO         ; ASSUME POSITIVE VALUE
135 012650 105767 003527'        TSTB     NEGVAL          ; NEGATIVE?
136 012654 001402          BEQ      1$             ; BR IF NOT
137 012656 012700 000055          MOV      #'-,RO         ; GET NEGATIVE SIGN CHAR
138 012662 110077 000000G        1$:     MOVB     RO,@TOPMEM     ; STORE AS 1ST CHAR OF NUMBER
139 012666 000474          BR       SLNEND
140
141          ; Trailing separate sign.
142
143 012670 012700 000053        SLTSS:  MOV      #'+,RO         ; ASSUME POSITIVE VALUE
144 012674 105767 003527'        TSTB     NEGVAL          ; NEGATIVE?
145 012700 001402          BEQ      1$             ; BR IF NOT
146 012702 012700 000055          MOV      #'-,RO         ; GET NEGATIVE SIGN CHAR
147 012706 110015          1$:     MOVB     RO,(R5)         ; STORE AS LAST DIGIT OF NUMBER
148 012710 000463          BR       SLNEND
149
150          ; Convert value to comp.
151
152 012712 016201 000010        SLCOMP: MOV      S$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$:     MOVB     (R4)+,RO         ; GET NEXT DIGIT FROM NUMBER STRING
161 012740 001432          BEQ      4$             ; BR IF END HIT
162 012742 162700 000060          SUB      #'0,RO          ; 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),RO         ; 2*VALUE + NEW DIGIT

```

```

172 012776 005515          ADC      (R5)          ;PROPOGATE CARRY
173 013000 006313          ASL      (R3)          ;4*VALUE
174 013002 006145          ROL      -(R5)
175 013004 006313          ASL      (R3)          ;8*VALUE
176 013006 006115          ROL      (R5)
177 013010 060023          ADD      R0,(R3)+      ;10.*VALUE + DIGIT
178 013012 005525          ADC      (R5)+
179 013014 061545          ADD      (R5),-(R5)    ;ADD HIGH ORDER PARTS
180 013016 011500          MOV      (R5),R0       ;COPY HIGH-ORDER FOR NEXT DIGIT
181 013020 005301          DEC      R1            ;DONE?
182 013022 003360          BGT      2$            ;BR IF MORE WORDS TO DO
183                               ; Go check next digit from number string.
184 013024 000744          BR       3$
185                               ; See if we need to negate the comp value we just accrued.
186 013026 105767 003527'  4$:      TSTB     NEGVAL    ;NEGATIVE?
187 013032 001412          BEQ      SLNEND        ;BR IF NOT
188 013034 016703 000000G  MOV      TOPMEM,R3     ;POINT TO VALUE
189 013040 016205 000010  MOV      S$SIZ1(R2),R5 ;GET # BYTES IN VALUE
190 013044 006205          ASR      R5            ;GET # WORDS
191 013046 000401          BR       5$
192 013050 005513          6$:      ADC      (R3)          ;PROPOGATE CARRY
193 013052 005423          5$:      NEG      (R3)+      ;NEGATE A WORD OF THE VALUE
194 013054 005305          DEC      R5            ;MORE TO DO?
195 013056 003374          BGT      6$            ;BR IF YES
196                               ;
197 013060 012601          SLNEND: MOV      (SP)+,R1
198                               ;
199                               ; finished accruing literal value.
200                               ;
201 013062 012605          SLEND:  MOV      (SP)+,R5
202 013064 012604          MOV      (SP)+,R4
203 013066 012603          MOV      (SP)+,R3
204 013070 000207          RETURN
205                               ;
206                               ;-----
207                               ; QUOTCK is called to see if the character in R0 is a quote or
208                               ; apostrophe character.
209                               ; On return, the condition codes are set for BEQ/BNE.
210                               ; The character in R0 is stored in a call called QUOTE.
211                               ;
212 013072 110067 003526'  QUOTCK: MOVB     R0,QUOTE    ;STORE THE CHARACTER
213 013076 120027 000042  CMPB     R0,#'"          ;IS IT QUOTE?
214 013102 001402          BEQ      1$            ;BR IF YES
215 013104 120027 000047  CMPB     R0,#''          ;APOSTROPHE?
216 013110 000207          1$:      RETURN          ;RETURN WITH CONDITION CODE SET

```

```

1
2 ; ACRDEC IS CALLED TO ACCRUE A DECIMAL VALUE.
3 ; WHEN CALLED R1 MUST POINT TO THE 1ST DIGIT OF THE NUMBER.
4 ; ON RETURN THE LOW-ORDER 16 BITS OF THE NUMBER ARE RETURNED
5 ; IN R5. THE HIGH-ORDER 16 BITS ARE STORED IN VALHI.
6 ; ALL OTHER REGISTERS ARE PRESERVED.
7 ;
8 013112 010046 ACRDEC: MOV R0, -(SP)
9 013114 010446 MOV R4, -(SP)
10 013116 005004 CLR R4
11 013120 005005 CLR R5
12 013122 112100 1#: MOV (R1)+, R0 ; PICK UP NEXT CHAR
13 013124 162700 000060 SUB #'0, R0 ; SEE IF DIGIT & CVT TO BINARY
14 013130 002421 BLT 7# ; BR IF NOT DIGIT
15 013132 020027 000011 CMP R0, #9. ; IS THIS A DIGIT?
16 013136 101016 BHI 7# ; BR IF NOT
17 ; MULTIPLY PREVIOUS VALUE BY 10.
18 013140 006305 ASL R5 ; LOW ORDER *2
19 013142 006104 ROL R4 ; HIGH ORDER *2
20 013144 010446 MOV R4, -(SP)
21 013146 010546 MOV R5, -(SP)
22 013150 006305 ASL R5 ; *4
23 013152 006104 ROL R4
24 013154 006305 ASL R5 ; *8.
25 013156 006104 ROL R4
26 013160 062605 ADD (SP)+, R5
27 013162 005504 ADC R4
28 013164 062604 ADD (SP)+, R4
29 ; ADD IN VALUE OF NEW DIGIT
30 013166 060005 ADD R0, R5
31 013170 005504 ADC R4
32 013172 000753 BR 1# ; GO GET NEXT CHAR
33 ; HIT END OF NUMBER
34 013174 010467 003454 ' 9#: MOV R4, VALHI ; SAVE HIGH ORDER PART
35 013200 005301 DEC R1 ; POINT BACK TO DELIMITER
36 013202 012604 MOV (SP)+, R4
37 013204 012600 MOV (SP)+, R0
38 013206 000207 RETURN

```

```

1 ; -----
2 ; ACROCT IS CALLED TO ACCRUE AN OCTAL VALUE.
3 ; WHEN CALLED R1 MUST POINT TO START OF NUMBER.
4 ; ON RETURN RESULT IS IN R5.
5 ; ALL OTHER REGISTERS ARE PRESERVED.
6 ;
7 013210 010046 ACROCT: MOV R0, -(SP)
8 013212 005005 CLR R5
9 013214 112100 2$: MOVB (R1)+, R0 ; GET NEXT CHAR
10 013216 162700 000060 SUB #'0, R0 ; CVT TO BINARY
11 013222 002410 BLT 9$ ; BR IF HIT END OF NUMBER
12 013224 020027 000007 CMP R0, #7 ; MAKE SURE ITS OCTAL
13 013230 101005 BHI 9$ ; BR IF HIT END OF NUMBER
14 013232 006305 1$: ASL R5 ; MULTIPLY PREVIOUS VALUE BY 8
15 013234 006305 ASL R5
16 013236 006305 ASL R5
17 013240 060005 ADD R0, R5 ; ADD IN NEW DIGIT
18 013242 000764 BR 2$ ; GO GET NEXT CHAR
19 ; HIT END OF NUMBER
20 013244 005301 9$: DEC R1 ; POINT BACK TO DELIMITER
21 013246 012600 MOV (SP)+, R0
22 013250 000207 RETURN

```

```

1
2 ;
3 ; GTCPRM IS CALLED TO ACCRUE A SWITCH PARAMETER OF
4 ; THE FORM ":CHAR:COL".
5 ; ON RETURN, THE CHARACTER CODE IS IN R2 AND THE
6 ; COLUMN NUMBER IS IN R5. R1 POINTS PAST THE PARAM.
7 ; ALL OTHER REGS ARE PRESERVED.
8 ;
9 GTCPRM: CALL CKCOL ; CHECK FOR COLON
10 ; CMPB (R1),#' ' ; QUOTED CHAR?
11 ; BNE 1# ; BR IF NOT
12 ; INC R1 ; SKIP OVER QUOTE
13 ; MOVB (R1)+,R2 ; GET THE CHAR
14 ; INC R1 ; SKIP TRAILING QUOTE
15 ; BR 2#
16 ;
17 1#: CALL ACROCT ; ACCRUE OCTAL CHARACTER CODE
18 ; MOV R5,R2
19 ;
20 2#: CALL CKCOL ; CHECK FOR COLON
21 ; CALL ACRDEC ; ACCRUE DEC COLUMN #
22 ; RETURN
23 ; IF DF RT11
24 ;
25 ;
26 ;
27 ;
28 ;
29 ; LOADIT IS CALLED TO LOAD A DEVICE HANDLER INTO THE
30 ; TOP OF MEMORY. WHEN CALLED R1 MUST POINT TO A CELL
31 ; CONTAINING THE RAD50 DEVICE NAME AND R5 MUST POINT TO
32 ; THE TOP OF FREE MEMORY.
33 ; ON RETURN R5 POINTS TO THE NEW TOP OF MEMROY.
34 ;
35 LOADIT: TST (R1) ; IS THERE A DEVICE NAME SPECIFIED?
36 ; BEQ 9#
37 ; MOV R0, -(SP)
38 ;
39 ; CHECK STATUS OF DEVICE
40 ; .SERR
41 ; .DSTAT #CBLK,R1
42 ; BCC 1# ; BR IF DEVICE RECOGNIZED
43 ; XXX 7 ; DON'T RECOGNIZE DEVICE
44 ;
45 1#: TST CBLK+4 ; IS HANDLER RESIDENT NOW?
46 ; BNE 2# ; BR IF YES
47 ; SUB <CBLK+2>,R5 ; RESERVE ROOM FOR HANDLER
48 ; .FETCH R5,R1 ; LOAD HANDLER
49 ; BCS 4# ; BR IF CAN'T GET IT
50 ;
51 2#: .HERR
52 ; MOV (SP)+,R0
53 ;
54 9#: RETURN
55 ; .ENDC
56 ;
57 ;
58 ;
59 ;
60 ;
61 ; CKCOL IS CALLED TO CHECK THAT THE NEXT CHARACTER IS
62 ; A COLON. IF NOT AN ERROR ABORT OCCURS.
63 ; IF YES THE COLON IS SKIPPED OVER.
64 ;
65 CKCOL: CMPB (R1)+, #' : ' ; IS NEXT CHAR A COLON?
66 ; BEQ 1# ; BR IF YES
67 ; XXX 19 ; SYNTAX ERROR IF NOT
68 ;
69 1#: RETURN

```

```

1 ; -----
2 ; Subroutine to determine the correct default file extension based on
3 ; the file type.
4 ; When called, R2 must contain the file type flags (RT$xxx).
5 ; On return, R2 contains the Rad-50 default file extension.
6 ;
7 013428 032702 000000G GETDFX: BIT #RT$CBL,R2 ; COBOL FILE?
8 013432 001417 BEQ 1$ ; BR IF NOT
9 ; COBOL file.
10 013434 032702 000000G BIT #RT$OX,R2 ; ORGANIZATION = INDEXED?
11 013440 001403 BEQ 2$ ; BR IF NOT
12 013442 016702 000060' MOV R5OCIO,R2 ; EXT = CIO
13 013446 000207 RETURN
14 013450 032702 000000G 2$: BIT #RT$RB,R2 ; MODE = BINARY?
15 013454 001403 BEQ 3$ ; BR IF NOT
16 013456 016702 000056' MOV R5OCDB,R2 ; EXT = CDB
17 013462 000207 RETURN
18 013464 016702 000054' 3$: MOV R5OCDA,R2 ; EXT = CDA
19 013470 000207 RETURN
20 013472 032702 000000G 1$: BIT #RT$DBL,R2 ; DIBOL FILE?
21 013476 001403 BEQ 4$ ; BR IF NOT
22 ; DIBOL file.
23 013500 016702 000050' MOV R5ODDF,R2 ; EXT = DDF
24 013504 000207 RETURN
25 013506 016702 000052' 4$: MOV R5ODAT,R2 ; SET EXTENTION TO "DAT"
26 013512 000207 RETURN

```

```

1          .SBTTL  PRTNUM -- Print a decimal number
2          ;-----
3          ; PRTNUM is called to convert a 32-bit binary value to ascii and print
4          ; the resulting string on the terminal without trailing Cr-Lf.
5          ;
6          ; Inputs:
7          ;   R4 = low-order 16-bits of value
8          ;   R5 = High-order 16-bits of value
9          ;
10         013514 010146 PRTNUM: MOV     R1, -(SP)
11         ;
12         ; Convert the value to an asciz string.
13         ;
14         013516 004767 000016          CALL     CVTNUM          ; CONVERT VALUE TO ASCIZ STRING
15         ;
16         ; Print the result.
17         ;
18         013522 112100 1$:      MOVB   (R1)+, R0          ; GET NEXT DIGIT FROM STRING
19         013524 001403          BEQ     2$          ; BR IF ASCIZ NULL HIT
20         013526          .TTYOUT          ; PRINT THE DIGIT
21         013532 000773          BR     1$
22         ;
23         ; Finished
24         ;
25         013534 012601 2$:      MOV     (SP)+, R1
26         013536 000207          RETURN
27         .SBTTL  CVTNUM -- Convert binary value to asciz string
28         ;-----
29         ; CVTNUM is called to convert a 32-bit binary value to an ascii digit string.
30         ;
31         ; Inputs:
32         ;   R4 = Low-order 16-bits of value
33         ;   R5 = High-order 16-bits of value
34         ;
35         ; Outputs:
36         ;   R1 = Address of start of asciz string.
37         ;
38         013540 010446 CVTNUM: MOV     R4, -(SP)
39         013542 010546          MOV     R5, -(SP)
40         ;
41         ; Get pointer to end of asciz result buffer.
42         ;
43         013544 012701 011702'          MOV     #CVTBUF+9, R1 ; POINT TO NULL BYTE AT END OF CONVERT BUFFER
44         ;
45         ; Split off low order digit.
46         ;
47         013550 012700 000012 1$:      MOV     #10, R0          ; SET DIVIDE BY 10.
48         013554 004767 000000G          CALL   DIVIDE          ; DIVIDE R4-R5 BY 10.
49         013560 062700 000060          ADD     #'0, R0          ; CONVERT REMAINDER TO ASCII DIGIT
50         013564 110041          MOVB   R0, -(R1)          ; MOVE TO RESULT BUFFER
51         013566 005704          TST    R4          ; MORE TO CONVERT?
52         013570 001367          BNE   1$          ; BR IF YES
53         013572 005705          TST    R5          ; CHECK HIGH-ORDER PART OF QUOTIENT
54         013574 001365          BNE   1$
55         ;
56         ; Finished
57         ;

```

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

```

1
2
3
4
5
6 013604 010067 000136'
7 013610 012706 000000G
8
9 013614
10 013662
11 014012
12 014060 005767 000000G
13 014064 001454
14 014066
15 014216
16
17
18 014216 105767 000000G
19 014222 001123
20 014224
21 014232 012701 000000G
22 014236 005711
23 014240 001407
24 014242 005731
25 014244 001774
26 014246 005741
27 014250
28 014256 000767
29 014260
30
31
32
33
34 014260 105767 000000G
35 014264 002022
36 014266 016767 000000G 000142'
37 014274 016767 000002G 000140'
38 014302 012700 000126'
39 014306 104375
40 014310 103006
41 014312 105067 000000G
42 014316
43 014326 000167 163636
44
45
46 014332 016701 000136'
47 014336 001034
48 014340 152737 000001 000053
49
50 014346 105767 000000G
51 014352 001040
52
53 014354
54 014362 016704 000000G
55 014366 016705 000000G
56 014372 004767 177116
57 014376
    
```

```

.SBTTL EXECUTION TERMINATION
-----
; TERMINATION ROUTINE.
; ENTER WITH ERROR CODE IN RO.
;
TERM:  MOV     RO,ABTBUF
        MOV     #STACK,SP      ;RESET SP
; MAKE SURE ALL CHANNELS ARE CLOSED
        $KEEP  #FD1
        $PURGE #FD2
        $KEEP  #FD3
        TST    T2SIZ           ;DID WE USE TEMP FILE 2?
        BEQ    7$              ;BR IF NOT
        $PURGE #FD4
7$:
; IF DF RT11
; RELEASE DEVICE HANDLERS
12$:   TSTB    SPCFL6           ;Special chain mode?
        BNE    13$             ; => yes
        .SERR
        MOV    #FILES,R1      ;R1 -> TABLE OF -> FILE SPECS
11$:   TST    (R1)             ;(R1) == 0 IF AT END OF TABLE
        BEQ    8$              ;EXIT IF END
        TST    @(R1)+          ;CHECK DEVICE NAME
        BEQ    11$             ;IGNORE BLANK DEVICE
        TST    -(R1)           ;REVERSE AUTO-INCREMENT FROM ABOVE
        .RELEAS (R1)+         ;RELEASE HANDLER
        BR     11$
8$:
; .SETTOP #SIEND              ;RESET TOP OF MEMORY
;
; IF WE WERE CALLED IN MESSAGE MODE SEND BACK A RESPONSE
;
        TSTB    MODE           ;MESSAGE MODE?
        BGE    1$              ;BR IF NOT
        MOV    SRTCNT,MRC1     ;SET SORTED RECORD COUNT
        MOV    SRTCNT+2,MRC2
        MOV    #SNDBLK,RO      ;SEND REPLY
        EMT    375
        BCC    10$             ;BR IF NO ERROR
        CLRB   MODE           ;GET OUT OF MESSAGE MODE
        XXX    26
10$:   JMP     SORTR           ;WAIT FOR ANOTHER MESSAGE
;
; SEE IF ANY ERROR OCCURED.
1$:   MOV    ABTBUF,R1         ;GET ERROR CODE
        BNE    2$              ;BR IF ERROR OCCURED
        BISB   #NOERR,@#USERRB ;SET SUCCESS FLAG
; ONLY PRINT "SORT COMPLETED" IF IN DIRECT RUN MODE.
        TSTB   MODE
        BNE    3$              ;BR IF IN CHAIN MODE
; Print number of records read and sorted.
        .PRINT #TXNRR         ;NUMBER OF RECORDS READ:
        MOV    RCNT1,R4        ;GET LOW-ORDER VALUE
        MOV    RCNT2,R5        ;GET HIGH-ORDER VALUE
        CALL   PRNUM           ;PRINT THE VALUE
        .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     PRNUM          ;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
    
```

ABTBUF	000136R	002	EM20	004730R	002	F\$INPL=	***** G	KLEN	=	***** G	DWCNT	=	***** 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		
ACRDOT	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\$LD	=	000002 G	EM30	005502R	002	GETSW	001516R	005	LOADIT	013314R	005	GLEN	=	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	
CHND	=	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	EDFBUF=	***** G		GTOK	003240R	005	M\$\$SZ	=	000006 G		RT\$CBL=	***** G	
CMPTMP	003442R	002	EDFCHR=	***** G		GTORG	003074R	005	NEGVAL	003527R	002	RT\$CR	=	***** G	
CODGEN	006526R	005	EDFCOL=	***** 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		NUMCD	=	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	
CSPCDF=	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	DBUFSZ=	***** 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\$EDE=	000001		R50TMP	000046R	002	
EM14	004470R	002	FT\$F	=	000010	KAD	=	***** G	OP\$LPN=	000004		R50TNO	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	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

SELADR= ***** G	SLPSOP 004142R	005 SWBUF 002052R	002 TMPCNT= ***** G	XXX21 = 002664R	005
SEDFR 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 SWEDF 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	SWDTYP 002722R	005 T2SIZ = ***** G	XXX30 = 012614R	005
SKPCNT= ***** G	SO\$OR = 000004 G	SWPAUS 002452R	005 UEDF = ***** G	XXX31 = 004312R	005
SKPVAL= ***** G	SPCFLG= ***** G	SWPPN 002534R	005 USERRB= 000053	XXX32 = 004360R	005
SLBUF 001770R	002 SPC1 = ***** G	SWREC 001744R	005 USRLDC= 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
SLEND 013062R	005 SRTFHD 011607R	002 S\$LINK= 000000 G	WRDS = 000004	XXX5 = 012170R	005
SLLIT 012204R	005 SRTINI 000000R	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 SWBLK 001770R	005 TERM 013604R	005 XXX20 = 001600R	005	

. ABS. 000000 000 (RW, I, GBL, ABS, DVR)
 000000 001 (RW, I, LCL, REL, CON)
 SORTID 011704 002 (RW, D, LCL, REL, CON)
 SWNAM 000042 003 (RW, I, GBL, REL, DVR)
 SWBRV 000042 004 (RW, I, GBL, REL, DVR)
 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, CRF=DK: CBSRTH, CBSRTI/C

SLCOMP	30-80	30-82	30-152#																		
SLCSPC	19-12#	19-18	19-179																		
SLLALP	30-23#	30-28	30-31																		
SLEND	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	19-80	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	3-53#	36-65																			
SRTINI	2-54	9-6#																			
SRTNAM	4-25	3-52#																			
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							

\$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	1-449#	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	1-676#													
\$PURGE	1-465#	22-102	36-10	36-14										
\$READ	1-492#	26-30												
\$REOPE	1-309#	22-17												
\$SAVES	1-289#	25-129												
\$SPC	1-159#													
\$SYDON	1-674#													
\$SYINI	1-589#	7-7												
\$SYSTE	1-171#													
\$UNLOC	1-338#	11-54	22-105											
\$WAIT	1-558#	26-32												
\$WRITE	1-525#													
... CM0	7-7	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	7-7	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	1-58#	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#	7-37												
.GVAL	1-100#													
.HERR	1-97#	9-7	22-11	22-12	22-102	33-42	36-10	36-14						
.LOCK	1-56#	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#	7-7												
.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					

.SETTO	1-97#	11-33	12-12											
.TTYIN	1-99#	9-51	9-56											
.TTYOU	1-98#	7-50	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
		36-11	36-14											36-10
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
		36-11	36-14											36-10
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
		17-87	17-117	19-147	20-9	20-20	20-54	21-8	22-86	24-18	25-34	25-58	25-126	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	

Table of contents

4-	1	ALLOCATE CORE SPACE
5-	1	MAIN SORT ROUTINE
6-	1	I/O ROUTINES

```

1          .TITLE  SORT1  SORT -- PHASE 1
2 000000   .CSECT  SORT1
3          .ENABL  LC
4          000000' P1BASE =
5          ;
6          ; THIS SECTION OF SORT CREATES THE INITIAL SORTED RUNS.
7          ; IT USES A REPLACEMENT SELECTION ALGORITHM FOR THIS PROCESS.
8          ; SEE KNUTH 'ART OF COMPUTER PROGRAMMING' VOLUME 3 SECT. 5.4.1
9          ; FOR A DISCUSSION OF THIS TECHNIQUE. THE SYMBOLS AND LABELS USED IN
10         ; THIS PROGRAM AGREE WITH THE CONVENTIONS USED BY KNUTH.
11         ;
12         ; Copyright (c) 1980, 1981.
13         ; S&H Computer Systems, Inc.
14         ; Nashville, Tennessee USA
15         ;
16         ; All rights reserved.
17         ; This software may not be sold, distributed or otherwise made
18         ; available for use except as licensed by S&H Computer Systems, Inc.
19         ;
20         ; COBOL-Plus, SORT-Plus, RTSORT, TSX and TSX-Plus are registered
21         ; trademarks of S&H Computer Systems, Inc.
22         ;
23         ;
24         ;
25         ; GLOBAL DEFINITIONS.
26         ;
27         .GLOBL  SORTP1,P1TOP,P1BASE,P1SIZ
28         ;
29         ; GLOBAL REFERENCES.
30         ;
31         .GLOBL  KTYPE,FD1,SETNRT
32         .GLOBL  TOPMEM,RECLN,FT%D,SO$AND,SO$NOT
33         .GLOBL  UEQF
34         .GLOBL  FILRN,FILST,FILFL,NUMKY2
35         .GLOBL  RWDLEN,MXFILS,MXFIL2,FILNB
36         .GLOBL  SKPCNT,INFLEN,OWCNT,BLKFC
37         .GLOBL  OBUF1,OBUF2,OBUFSZ,EOFBUF
38         .GLOBL  OBFEND,MXREC1,MXREC2
39         .GLOBL  KYDATE,KYLSS,KYTSS
40         .GLOBL  KYBIN,ITYPE,SRTCNT,TMPCNT
41         .GLOBL  RT$CBL,RT$DBL,RT$FTN,RT$TXT
42         .GLOBL  RT$OS,RT$OR,RT$OX,RT$F11
43         .GLOBL  RT$RA,RT$RB,RT$VLN,RT$CR
44         .GLOBL  NUMBUF,NUMCOL,NUMLEN
45         .GLOBL  RCNT1,RCNT2,EOFCOL,EOFCHR,ABORT
46         .GLOBL  NXTBLK,ORPOS,FLUSH,COMPAR
47         .GLOBL  P1TERM,LOWMEM,FLBLK,ISMARS
48         .GLOBL  OUTWRT,IFPNT,CUROFL,KSTRT,FT$T
49         .GLOBL  FLDMAP,INRLEN,MAPEND,SELADR
50         .GLOBL  S$OPR,SO$CMP,SO$OR,S$POS1,S$POS2
51         .GLOBL  S$FLG,SF$LIT,S$SIZ1,S$TYP,BT$LO
52         .GLOBL  BT$HI,BT$REV,S$LINK,KYCOMP,KYDIS
53         .GLOBL  KYUCA,M$POS1,M$POS2,M$SIZE,M$SZ
54         .GLOBL  S$BR,ORFLG,ANDFLG,CISR11
55         .GLOBL  CURSAV,SKPVAL

```

```

1          ;
2          000620      MAXREC =          400.      ; MAX # OF RECORDS TO PLACE IN CORE
3          ;
4          ; RECORD DESCRIPTOR TABLES.
5 000000      LOC:      .BLKW  MAXREC      ; ADDRESS OF RECORD(I).
6 001440      LOSER:   .BLKW  MAXREC      ; INDEX OF LOSER BELOW THIS NODE.
7 003100      FE:      .BLKW  MAXREC      ; EXTERNAL BLOCK FORWARD LINK.
8 004540      FI:      .BLKW  MAXREC      ; INTERNAL BLOCK FORWARD LINK.
9 006200      RN:      .BLKW  MAXREC      ; RUN # THIS RECORD WILL GO IN.
10         ; GENERAL DATA AREA.
11         CR          =          15      ; ASCII CARRIAGE RETURN
12         LF          =          12      ; ASCII LINE FEED
13         CTRLZ       =          32      ; CTRL-Z EOF CHARACTER
14 007640 000000      NUMREC: .WORD  0      ; # OF RECORDS WHICH WILL FIT IN CORE.
15 007642 000000      IBUF1:  .WORD  0      ; ADDRESS OF PRIMARY INPUT BUFFER.
16 007644 000000      IBUF2:  .WORD  0      ; ADDRESS OF SECONDARY INPUT BUFFER.
17 007646 000000      RQ:      .WORD  0      ; RUN # RECORD(Q) WILL GO IN.
18 007650 000000      RC:      .WORD  0      ; CURRENT RUN NUMBER.
19 007652 000000      RMAX:   .WORD  0      ; # OF HIGHEST RUN IN FILE.
20 007654 000000      Q:      .WORD  0      ; INDEX OF NEW RECORD.
21 007656 000000      LSTKEY: .WORD  0      ; ADDRESS OF LAST WINNER RECORD.
22 007660 000000      IBFEND: .WORD  0      ; ADDR OF END OF INPUT BUF.
23 007662 000000      IRPOS:  .WORD  0      ; POINTER TO NEXT INPUT REC.
24 007664 000000      INRDSZ: .WORD  0
25 007666 000000      OFLINK: .WORD  0
26 007670 000000      MAPBUF: .WORD  0
27 007672 000000      RECPtr: .WORD  0
28 007674 000000      SELREC: .WORD  0
29 007676 000000      LSTACK: .WORD  0
30         ; BYTE DATA ITEMS.
31 007700      000      EOFFLG: .BYTE  0
32         ; Logical operation temp stack.
33 007701          .BLKB  6.
34 007707      LSTKTP:
35         ;
36         .EVEN
37         ;
38         ; MACRO TO CAUSE ABORT
39         ;
40         .MACRO  XXX      CODE
41         XXX 'CODE      =
42         MOV      #CODE, R0
43         CALL    ABORT
44         .ENDM   XXX
45
46

```

```

1          .SBTTL  ALLOCATE CORE SPACE
2          ;
3          ; Check in for debugging breakpoint.
4          ;
5 007710 004767 000000G  SORTP1: CALL  CISRT1          ;CHECK IN TO ROOT
6          ;
7          ; ALLOCATE AVAILABLE CORE SPACE.
8          ;
9 007714          $GETBOT R1,#P1TOP
10         ; Allocate room for temp record buffer if field mapping is wanted.
11 007734 005767 000000G  TST  FLDMAP          ;IS FIELD MAPPING WANTED?
12 007740 001407          BEQ  6#          ;BR IF NOT
13 007742 010167 177722  MOV  R1,MAPBUF        ;SAVE ADDRESS OF START OF BUFFER
14 007746 066701 000000G  ADD  INRLEN,R1       ;ADD LENGTH OF INPUT LOGICAL RECORDS
15 007752 005201          INC  R1          ;BOUND UP TO WORD BOUNDARY
16 007754 042701 000001  BIC  #1,R1
17         ; ALLOCATE FILE I/O BUFFERS.
18 007760 010167 177656  6#:  MOV  R1,IBUF1        ;INPUT BUFFER 1.
19 007764 010167 177672  MOV  R1,IRPOS        ;ADDRESS OF 1ST. REC.
20 007770 010167 177664  MOV  R1,IBFEND       ;FORCE SETUP OF 1ST BUFFER.
21 007774 066701 000000G  ADD  OBUFSZ,R1
22 010000 010167 177640  MOV  R1,IBUF2        ;INPUT BUFFER 2.
23 010004 066701 000000G  ADD  OBUFSZ,R1
24 010010 010167 000000G  MOV  R1,OBUF1        ;OUTPUT BUFFER 1.
25 010014 066701 000000G  ADD  OBUFSZ,R1
26 010020 010167 000000G  MOV  R1,OBUF2        ;OUTPUT BUFFER 2.
27 010024 066701 000000G  ADD  OBUFSZ,R1
28         ; SET UP RECORD LOCATION TABLE.
29 010030 010167 177622  MOV  R1,LSTKEY       ;BUFFER FOR LAST WINNER
30 010034 066701 000000G  ADD  RWDLEN,R1
31 010040 005002          CLR  R2
32 010042 016703 000000G  MOV  TOPMEM,R3       ;GET HIGHEST AVAILABLE MEMORY ADDRESS.
33 010046 166703 000000G  SUB  RWDLEN,R3       ;LEAVE ROOM FOR A RECORD.
34 010052 012704 000000'  MOV  #LOC,R4        ;GET ADDRESS OF LOCATION TABLE.
35 010056 020103 2#:  CMP  R1,R3          ;IS THERE ROOM FOR ANOTHER RECORD?
36 010060 101010          BHI  1#          ;BRANCH IF NOT.
37 010062 010124          MOV  R1,(R4)+        ;SET ADDRESS OF RECORD.
38 010064 005202          INC  R2
39 010066 020227 000620  CMP  R2,#MAXREC     ;ALLOCATED MAX # OF RECORDS?
40 010072 103003          BHI  1#          ;BRANCH IF YES.
41 010074 066701 000000G  ADD  RWDLEN,R1       ;MOVE ON TO NEXT RECORD.
42 010100 000766          BR   2#
43 010102 010267 177532  1#:  MOV  R2,NUMREC     ;SAVE # OF RECORDS TO MERGE.
44 010106 020227 000004  CMP  R2,#4          ;CHECK FOR REASONABLE MINIMUM
45 010112 103004          BHI  5#
46 010114          XXX  24.          ;ERROR IF NOT ENOUGH CORE
47 010124 010205 5#:  MOV  R2,R5
48 010126 005302          DEC  R2          ;START J=P-1.
49 010130 010503 3#:  MOV  R5,R3          ;GET P.
50 010132 060203          ADD  R2,R3          ;P+J.
51 010134 006203          ASR  R3          ;(P+J)/2
52 010136 006303          ASL  R3          ;CONVERT TO WORD TABLE INDEX.
53 010140 010204          MOV  R2,R4          ;GET J.
54 010142 006204          ASR  R4          ;J/2
55 010144 006304          ASL  R4          ;CONVERT TO WORD TABLE INDEX.
56 010146 006302          ASL  R2          ;CONVERT J TO WORD TBL INDEX.
57 010150 010362 003100'  MOV  R3,FE(R2)     ;EXTERNAL NODE FLINK.

```

```

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      CLR    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
  
```

```

1          .SBTTL  MAIN SORT ROUTINE
2          ;
3          ; ENTER MAIN SORT ROUTINE
4          ;
5 010532 016701 177112 SR2:  MOV  RC,R1      ;GET CURRENT RUN NUMBER.
6 010536 020167 177104    CMP  R1,RQ      ;DOES THIS RECORD GO IN THIS RUN?
7 010542 001471          BEQ  SR3      ;BRANCH IF THIS RECORD IS IN THIS RUN.
8          ;
9          ; WIND UP PROCESSING ON RUN NUMBER RC.
10         ;
11 010544 005767 177100    TST  RC          ;IS THIS THE INITIALIZATION RUN?
12 010550 001463          BEQ  ENDINI      ;BRANCH IF INITIALIZATION RUN.
13 010552 016702 000000G RUNEND: MOV  CUROFL,R2    ;GET # OF CURRENT OUTPUT FILE.
14 010556 005262 000000G    INC  FILRN(R2)  ;INCREMENT # OF RUNS IN THIS FILE.
15 010562 016703 000000G    MOV  OBUF1,R3    ;GET ADDRESS OF CURRENT OUTPUT BUFFER.
16 010566 052713 100000    BIS  #^D100000,(R3) ;SET END OF RUN FLAG IN BUFFER.
17 010572 016767 177070 000000G MOV  OFLINK,FLBLK ;LINK FORWARD TO PREV RUNS IN FILE
18 010600 016701 000000G    MOV  ORPOS,R1    ;CALCULATE # OF CHARS IN BUFFER
19 010604 160301          SUB  R3,R1
20 010606 010163 000002G    MOV  R1,2(R3)    ;SET # OF CHARS IN BUFFER
21 010612 162701 000004G    SUB  #4.,R1      ;SUBTRACT # CONTROL BYTES IN BUFFER
22 010616 060167 000000G    ADD  R1,TMPCNT   ;ACCUMULATE # DATA BYTES BEING WRITTEN
23 010622 005567 000002G    ADC  TMPCNT+2   ;TO TEMP FILE
24 010626 004767 000000G    CALL FLUSH      ;WRITE OUT LAST BLK
25 010632 005567 000002G    ADC  TMPCNT+2   ;PROPOGATE CARRY
26         ;
27         ; ADVANCE TO NEXT OUTPUT RUN
28         ;
29 010636 062702 000002G    ADD  #2,R2      ;ADVANCE TO NEXT FILE NUMBER
30 010642 020227 000000G    CMP  R2,#MXFIL2 ;HAVE WE PASSED LAST FILE?
31 010646 103401          BLO  2$        ;BRANCH IF NOT.
32 010650 005002          CLR  R2        ;RECYCLE TO FILE # 0.
33 010652 010267 000000G 2$:  MOV  R2,CUROFL
34 010656 026767 176764 176766  CMP  RQ,RMAX    ;FINISHED WITH LAST RUN?
35 010664 101402          BLOS 3$        ;BR IF NOT
36 010666 000167 004134    JMP  S1STOP     ;FINISH PHASE 1
37 010672 016267 000000G 176766 3$:  MOV  FILST(R2),OFLINK ;SAVE POINTER TO REST OF FILE
38 010700 016703 000000G    MOV  NXTBLK,R3  ;ASSIGN IT A STARTING BLOCK
39 010704 010362 000000G    MOV  R3,FILST(R2)
40 010710 010367 000000G    MOV  R3,FILFL
41 010714 005267 000000G    INC  NXTBLK
42 010720 016767 176722 176722 ENDINI: MOV  RQ,RC      ;ADVANCE RUN NUMBER
43         ;
44         ; OUTPUT TOP OF TREE. (Q NOW POINTS TO THE CHAMPION AND RQ IS ITS
45         ; RUN NUMBER)
46         ;
47 010726 005767 176714 SR3:  TST  RQ          ;IS THIS INITIALIZATION RUN?
48 010732 001466          BEQ  SR4          ;BRANCH IF YES.
49         ; MOVE RECORD(Q) TO OUTPUT BUFFER.
50 010734 016702 176714    MOV  Q,R2       ;GET INDEX TO WINNER
51 010740 016203 000000'    MOV  LOC(R2),R3 ;GET ADDRESS OF WINNER RECORD
52 010744 016762 176706 000000' MOV  LSTKEY,LOC(R2) ;LAST WINNER STAYS IN 'LSTKEY' BUFFER
53 010752 010367 176700    MOV  R3,LSTKEY
54 010756 016702 000000G    MOV  ORPOS,R2   ;POSITION IN OUTPUT BUFFER
55 010762 016705 000000G    MOV  OBFEND,R5  ;ADDR OF END OF BUFFER
56 010766 010304          MOV  R3,R4       ;POINT TO END OF RECORD
57 010770 066704 000000G    ADD  RECLN,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.

```

MAIN SORT ROUTINE

```

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 011250 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

```

```

1          .SBTTL  I/O ROUTINES
2          ;
3          ;-----
4          ; GETREC is called to obtain the next logical record from the
5          ; input file.  GETREC is responsible for reading records,
6          ; doing record selection, doing field mapping and checking
7          ; for end of file.
8          ; When called, R1 must contain the address of the buffer where
9          ; the record is to be stored.
10         ; On return, the C-flag is set if End of file is hit.
11         ;
12 011356 105767 176316  GETREC: TSTB    EOFFL0      ;HAVE WE HIT END OF FILE PREVIOUSLY?
13 011362 001402          BEQ      1$          ;BR IF NOT
14 011364 000261          SEC              ;SIGNAL END OF FILE AGAIN
15 011366 000207          RETURN
16 011370 010167 176276 1$:      MOV      R1,RECPT  ;SAVE ADDRESS OF RECORD BUFFER
17         ;
18         ; If field mapping is wanted, move the record to a logical buffer
19         ; first.
20         ;
21 011374 005767 000000G          TST      FLDMAP      ;IS FIELD MAPPING WANTED?
22 011400 001402          BEQ      GRREAD      ;BR IF NOT
23 011402 016701 176262          MOV      MAPBUF,R1   ;GET ADDRESS OF HOLDING BUFFER
24         ;
25         ; See if we have already read max allowed number of records.
26         ;
27 011406 026767 000000G 000000G GRREAD: CMP      RCNT1,MXREC1  ;READ MAX ALLOWED # RECORDS?
28 011414 001004          BNE      1$          ;BR IF NOT
29 011416 026767 000000G 000000G  CMP      RCNT2,MXREC2  ;CHECK HIGH ORDER PART OF NUMBER
30 011424 001506          BEQ      GREOF      ;READ MAX # OF RECORDS -- SIGNAL EOF
31         ;
32         ; Read a record from input file into buffer pointed to by R1.
33         ;
34 011426 004767 000234 1$:      CALL     READIN      ;READ A RECORD
35 011432 103503          BCS      GREOF      ;BR IF END OF FILE
36         ;
37         ; Check for control-z as DIBOL end-of-file sentinel.
38         ;
39 011434 032767 000000G 000000G  BIT      #RT$DBL,ITYPE  ;IS THIS A DIBOL DATA FILE?
40 011442 001403          BEQ      3$          ;BR IF NOT
41 011444 121127 000032          CMPB    (R1),#CTRLZ   ;IS CONTROL-Z THE 1ST CHAR IN RECORD?
42 011450 001474          BEQ      GREOF      ;IF YES THEN THIS IS EOF
43         ;
44         ; Count number of records read.
45         ;
46 011452 005267 000000G 3$:      INC      RCNT1      ;INCREMENT LOW ORDER COUNTER
47 011456 001002          BNE      2$          ;BR IF NO OVERFLOW
48 011460 005267 000000G          INC      RCNT2      ;INCREMENT HIGH ORDER COUNTER
49         ;
50         ; See if we need to skip over some records at front of file.
51         ;
52 011464 005767 000000G 2$:      TST      SKPCNT      ;IS SKIPPING WANTED?
53 011470 001403          BEQ      GRCKEF      ;BR IF NOT
54 011472 005367 000000G          DEC      SKPCNT      ;DEC # LEFT TO SKIP
55 011476 000743          BR       GRREAD      ;GO READ NEXT RECORD
56         ; See if this record contains user specified EOF character.
57 011500 016700 000000G GRCKEF: MOV      EOFCOL,RO   ;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    EOFLG          ;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       GREFD         ;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    DMAP          ;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$:      TST     NUMLN         ;IS RECORD NUMBER WANTED?
91 011614 001402          BEQ     7$          ;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$:      CLC              ;SAY NOT END OF FILE
103 011640 000207          RETURN
104
105         ; We just hit the end of the input file.
106         ; See if there are more input files to process.
107
108 011642 004767 001536    GREFD:  CALL    NEXTIN        ;TRY TO SWITCH TO NEXT INPUT FILE
109 011646 103403          BCS    1$          ;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.

```

```
115 ; Return with C-flag set.  
116 ;  
117 011656 105267 176016 1#: INCB EOFL0 ;REMEMBER WE HAVE HIT EOF  
118 011662 000261 SEC ;SIGNAL END OF FILE  
119 011664 000207 RETURN
```

```

1
2
3
4
5
6
7
8 011666 105767 176006 READIN: TSTB EOFFLG ; ARE WE PAST END OF FILE?
9 011672 001402 BEQ 5$ ; BR IF NOT
10 011674 000261 SEC ; SIGNAL END OF FILE
11 011675 000207 RETURN
12 011700 010246 5$: MOV R2, -(SP)
13 011702 010346 MOV R3, -(SP)
14 011704 010446 MOV R4, -(SP)
15 011705 010546 MOV R5, -(SP)
16 011710 010146 MOV R1, -(SP)
17 011712 016702 000000G MOV INRLEN, R2 ; GET DESIRED LENGTH OF INPUT RECORDS
18 011716 016704 175740 MOV IRPOS, R4 ; GET POINTER INTO CURRENT FILE BUFFER
19 011722 016705 175732 MOV IBFEND, R5 ; GET POINTER PAST END OF CURRENT BUFFER
20
21 ; Determine what type of record this is.
22
23 011726 016700 000000G MOV ITYPE, R0 ; GET RECORD TYPE FLAGS
24 011732 032700 000000G BIT #RT#CBL, R0 ; COBOL RECORD?
25 011736 001416 BEQ 1$ ; BR IF NOT
26 011740 032700 000000G BIT #RT#OX, R0 ; INDEXED ORGANIZATION FILE?
27 011744 001402 BEQ 3$ ; BR IF NOT
28 011746 000167 000656 JMP R1CX ; GO GET COBOL ISAM FILE RECORD
29 011752 032700 000000G 3$: BIT #RT#RB, R0 ; COBOL BINARY RECORD?
30 011756 001402 BEQ 2$ ; BR IF NOT
31 011760 000167 000514 JMP R1CB ; GO HANDLE BINARY RECORD
32 011764 032700 000000G 2$: BIT #RT#DR, R0 ; COBOL ASCII RELATIVE RECORD?
33 011770 001113 BNE RICAR ; BR IF YES
34 011772 000436 BR R1CA ; COBOL ASCII SEQUENTIAL RECORD
35 011774 032700 000000G 1$: BIT #RT#CR, R0 ; IS CARRIAGE-RETURN THE RECORD TERMINATOR?
36 012000 001033 BNE R1CA ; BR IF YES
37 012002 032700 000000G BIT #RT#F11, R0 ; FILES-11 SEQUENTIAL??
38 012006 001403 BEQ R1FL ; NOT FILES-11
39 012010 032700 000000G BIT #RT#OS, R0 ; SEQUENTIAL?
40 012014 001145 BNE R1RS ; BRANCH IF SO
41
42 ; Fixed length input record.
43
44 012016 004767 002530 R1FL: CALL INCNUM ; INCREMENT RECORD COUNT
45 012022 160405 3$: SUB R4, R5 ; CALC # BYTES LEFT IN INPUT BUFFER
46 012024 010203 MOV R2, R3 ; GET # BYTES WANTED FOR RECORD
47 012026 020305 CMP R3, R5 ; GOT ENOUGH BYTES LEFT IN BUFFER?
48 012030 101402 BLOS 1$ ; BR IF YES
49 012032 010503 MOV R5, R3 ; MOVE THIS MANY BYTES THIS TIME
50 012034 001410 BEQ 4$ ; BR IF BUFFER EMPTY
51 012036 160302 1$: SUB R3, R2 ; THIS MANY BYTES NEEDED AFTER THIS MOVE
52 012040 112421 2$: MOVB (R4)+, (R1)+ ; MOVE BYTES FROM INPUT BUFFER TO RECORD BUFFER
53 012042 005303 DEC R3 ; NEED MORE?
54 012044 003375 BGT 2$ ; BR IF YES
55 012046 005702 TST R2 ; DO WE NEED TO GET MORE OUT OF NEXT BUFFER?
56 012050 001002 BNE 4$ ; BR IF YES
57 012052 000167 000374 JMP RIEND

```

I/O ROUTINES

58	012055	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

```

1          ;
2          ; Variable length ASCII records terminated by carriage-return.
3          ;
4 012070 004767 002456 RICA: CALL INCNUM ; INCREMENT RECORD COUNT
5 012074 020405 6$: CMP R4,R5 ; ARE WE BEYOND END OF INPUT BUFFER?
6 012075 103403 BLD 1$ ; BR IF NOT
7 012100 004767 000704 CALL RIBUF ; READ NEXT INPUT BUFFER FULL
8 012104 103573 BCS RIEOF ; BR IF END OF FILE HIT
9 012106 112400 1$: MOVB (R4)+,R0 ; GET NEXT CHAR FROM INPUT BUFFER
10 012110 001005 BNE 2$ ; BR IF NOT NULL CHAR
11 012112 032767 0000006 0000006 BIT #RT#CBL, ITYPE ; FILE = COBOL SEQUENTIAL ORGANIZATION?
12 012120 001165 BNE RIEOF ; IF YES THEN NULL SIGNALS END OF FILE
13 012122 000764 BR 6$ ; IGNORE NULLS OTHERWISE
14 012124 020027 000032 2$: CMP R0,#32 ; IS THIS A CONTROL CHARACTER?
15 012130 101012 BHI 4$ ; BR IF NOT
16 012132 001560 BEQ RIEOF ; CONTROL-Z ==> END OF FILE
17 012134 120027 000015 CMPB R0,#15 ; CARRIAGE-RETURN?
18 012140 001417 BEQ 5$ ; BR IF GOT END OF RECORD
19 012142 120027 000012 CMPB R0,#12 ; LINE FEED?
20 012146 001752 BEQ 6$ ; IGNORE LINE FEEDS
21 012150 120027 000014 CMPB R0,#14 ; FORM FEED?
22 012154 001747 BEQ 6$ ; IGNORE FORM FEEDS
23 012156 005702 4$: TST R2 ; ROOM IN RECORD BUFFER FOR ANOTHER CHAR?
24 012160 003403 BLE 8$ ; BR IF NOT
25 012162 110021 MOVB R0,(R1)+ ; MOVE CHAR TO RECORD BUFFER
26 012164 005302 DEC R2 ; COUNT # CHARS LEFT TO GET
27 012166 000742 BR 6$ ; KEEP LOOKING FOR END OF RECORD
28 012170 8$: XXX 33. ; RECORD READ IS LONGER THAN INPUT BUFFER
29          ; Found end of record.
30          ; Pad rest of buffer with blanks.
31 012200 005702 5$: TST R2 ; DID WE COMPLETELY FILL BUFFER?
32 012202 003523 BLE RIEND ; BR IF YES
33 012204 012703 000040 MOV #' ,R3 ; GET FILLER SPACE
34 012210 110321 7$: MOVB R3,(R1)+ ; FILL RECORD AREA WITH BLANKS
35 012212 005302 DEC R2
36 012214 003375 BGT 7$
37 012216 000515 BR RIEND
  
```

```

1          ;
2          ; COBOL ascii record in relative organization file.
3          ;
4 012220 004767 002326 RICAR: CALL INCNUM ; INCREMENT RECORD COUNTER
5 012224 020405        CMP R4,R5 ; IS BUFFER EMPTY?
6 012226 103403        BLO 1# ; BR IF NOT
7 012230 004767 000554 CALL RIBUF ; READ IN ANOTHER BUFFER FULL
8 012234 103517        BCS RIEOF ; BR IF EOF HIT
9 012236 105724        1#: TSTB (R4)+ ; DOES RECORD EXIST?
10 012240 001014       BNE 4# ; BR IF YES
11         ; Record does not exist. Skip up to next one.
12 012242 010203       MOV R2,R3 ; GET RECORD LENGTH
13 012244 005203       INC R3 ; COUNT CR AT END OF RECORD
14 012246 060403       3#: ADD R4,R3 ; GET POINTER TO START OF NEXT RECORD
15 012250 020305       CMP R3,R5 ; PAST END OF CURRENT BUFFER?
16 012252 103405       BLO 2# ; BR IF NOT
17 012254 160503       SUB R5,R3 ; COUNT # CHARS SKIPPED OVER IN THIS BUFFER
18 012256 004767 000526 CALL RIBUF ; READ IN NEW BUFFER FULL
19 012262 103504       BCS RIEOF ; BR IF EOF HIT
20 012264 000770       BR 3# ; CONTINUE SKIPPING
21 012266 010304       2#: MOV R3,R4 ; GET POINTER TO START OF NEXT RECORD
22 012270 000753       BR RICAR ; SEE IF THAT RECORD EXISTS
23         ; Record exists -- Move to record buffer.
24 012272 020405       4#: CMP R4,R5 ; BUFFER EMPTY?
25 012274 103403       BLO 5# ; BR IF NOT
26 012276 004767 000506 CALL RIBUF ; READ NEW BUFFER LOAD
27 012302 103474       BCS RIEOF ; BR IF EOF HIT
28 012304 112421       5#: MOVB (R4)+,(R1)+ ; MOVE TO RECORD BUFFER
29 012306 005302       DEC R2 ; MOVE FULL RECORD SIZE
30 012310 003370       BGT 4#
31         ; Skip carriage-return at end of record.
32 012312 020405       CMP R4,R5 ; PAST END OF BUFFER?
33 012314 103403       BLO 6# ; BR IF NOT
34 012316 004767 000466 CALL RIBUF ; READ NEW BUFFER FULL
35 012322 103464       BCS RIEOF ; BR IF END OF FILE HIT
36 012324 005204       6#: INC R4 ; SKIP OVER CR
37 012326 000451       BR RIEND
  
```

```

1          ;
2          ; FILES-11 sequential
3          ; (First word contains record length)
4          ;
5 012330   RIRS:
6 012330 005204      INC      R4          ;ROUND R4 UP TO NEAREST WORD
7 012332 042704 000001  BIC      #1,R4
8 012336 020405      CMP      R4,R5          ; IS INPUT BUFFER EMPTY?
9 012340 103403      BLO      4$          ; BR IF NOT
10 012342 004767 000442  CALL    RIBUF        ; READ NEW BUFFER FULL
11 012346 103452      BCS      RIEOF        ; BR IF END OF FILE HIT
12          ; Get record control word.
13 012350 004767 002176 4$:      CALL    INCNUM        ; INCREMENT RECORD COUNTER
14 012354 012403      MOV      (R4)+,R3      ; GET RECORD SIZE
15 012356 001417      BEQ      6$          ; BR IF RECORD LENGTH = 0
16 012360 160302      SUB      R3,R2          ; COMPARE RECORD LENGTH WITH BUFFER SIZE
17 012362 002004      BGE      1$          ; BR IF IT WILL FIT
18 012364          XXX      33.          ; RECORD IS LONGER THAN BUFFER
19 012374 006203      1$:      ASR      R3          ; GET # WORDS IN RECORD
20 012376 020405      3$:      CMP      R4,R5          ; PASSED END OF INPUT BUFFER?
21 012400 103403      BLO      2$          ; BR IF NOT
22 012402 004767 000402  CALL    RIBUF        ; READ NEW BUFFER FULL
23 012406 103432      BCS      RIEOF        ; BR IF END OF FILE HIT
24 012410 012421      2$:      MOV      (R4)+,(R1)+      ; MOVE WORD FROM FILE BUFFER TO RECORD BUFFER
25 012412 005303      DEC      R3          ; MORE TO MOVE?
26 012414 003370      BGT      3$          ; BR IF YES
27          ; See if we need to pad with trailing nulls.
28 012416          6$:
29 012416 005003      CLR      R3          ; R3 = NULL, FOR PADDING END
30 012420 036767 000000G 000000G  BIT    RT#RB,ITYPE      ; BINARY RECORDS?
31 012426 001002      BNE      7$          ; BRANCH IF SO
32 012430 112703 000040  MOVB   #' ,R3          ; MUST BE ASCII, USE BLANK, NOT NULL
33 012434          7$:
34 012434 005702      TST      R2          ; DO WE NEED ANY PADDING NULLS AT END?
35 012436 003405      BLE      RIEOF        ; BR IF NOT
36 012440 110321      5$:      MOVB   R3,(R1)+      ; PUT IN PADDING NULLS
37 012442 005302      DEC      R2
38 012444 003375      BGT      5$
39 012446 000167 000000  JMP     RIEOF

```

```
1 ;  
2 ; Finished getting record.  
3 ;  
4 012452 000241 RIEND: CLC ; SIGNAL NO EOF  
5 012454 010467 175202 RIFIN: MOV R4, IRPOS ; SAVE POINTER INTO FILE BUFFER  
6 012460 012601 MOV (SP)+, R1  
7 012462 012605 MOV (SP)+, R5  
8 012464 012604 MOV (SP)+, R4  
9 012466 012603 MOV (SP)+, R3  
10 012470 012602 MOV (SP)+, R2  
11 012472 000207 RETURN  
12 ;  
13 ; End of file hit.  
14 ;  
15 012474 000261 RIEOF: SEC ; SIGNAL END OF FILE  
16 012476 000766 BR RIFIN ; RETURN
```

```

1      ;
2      ; COBOL binary records.
3      ; (First word contains record-exists flag & record length)
4      ;
5 012500 020405 RICB:  CMP      R4,R5      ; IS INPUT BUFFER EMPTY?
6 012502 103403      BLO      RSBGCW     ; BR IF NOT
7 012504 004767 000300 CALL    RIBUF      ; READ NEW BUFFER FULL
8 012510 103771      BCS      RIEOF      ; BR IF END OF FILE HIT
9      ; Get record control word.
10 012512 004767 002034 RSBGCW: CALL    INCNUM    ; INCREMENT RECORD COUNTER
11 012516 012403      MOV      (R4)+,R3    ; GET RECORD CONTROL WORD
12 012520 002413      BLT      RSBMOV     ; BR IF RECORD EXISTS
13 012522 001764      BEQ      RIEOF      ; BR IF END OF FILE HIT
14      ; Record is marked as deleted. Skip to start of next record.
15 012524 060403 1$:  ADD      R4,R3      ; GET ADDRESS OF START OF NEXT RECORD
16 012526 020305      CMP      R3,R5      ; BEYOND END OF CURRENT INPUT BUFFER?
17 012530 103405      BLO      2$        ; BR IF NOT
18 012532 160503      SUB      R5,R3      ; SUBTRACT AMT IN BUFFER WE ARE SKIPPING OVER
19 012534 004767 000250 CALL    RIBUF      ; READ NEXT BUFFER FULL
20 012540 103755      BCS      RIEOF      ; BR IF EOF HIT
21 012542 000770      BR       1$        ; CONTINUE SKIPPING
22 012544 010304 2$:  MOV      R3,R4      ; GET ADDRESS OF START OF NEXT RECORD
23 012546 000761      BR       RSBGCW     ; GO CHECK NEXT RECORD
24      ; Record exists.
25      ; Move it to record buffer.
26 012550 042703 100000 RSBMOV: BIC      #100000,R3    ; CLEAR RECORD-EXISTS FLAG
27 012554 001417      BEQ      6$        ; BR IF RECORD LENGTH = 0
28 012556 160302      SUB      R3,R2      ; COMPARE RECORD LENGTH WITH BUFFER SIZE
29 012560 002004      BGE      1$        ; BR IF IT WILL FIT
30 012562      XXX      33.        ; RECORD IS LONGER THAN BUFFER
31 012572 006203 1$:  ASR      R3        ; GET # WORDS IN RECORD
32 012574 020405 3$:  CMP      R4,R5      ; PASSED END OF INPUT BUFFER?
33 012576 103403      BLO      2$        ; BR IF NOT
34 012600 004767 000204 CALL    RIBUF      ; READ NEW BUFFER FULL
35 012604 103733      BCS      RIEOF      ; BR IF END OF FILE HIT
36 012606 012421 2$:  MOV      (R4)+,(R1)+    ; MOVE WORD FROM FILE BUFFER TO RECORD BUFFER
37 012610 005303      DEC      R3        ; MORE TO MOVE?
38 012612 003370      BGT      3$        ; BR IF YES
39      ; See if we need to pad with trailing nulls.
40 012614 005702 6$:  TST      R2        ; DO WE NEED ANY PADDING NULLS AT END?
41 012616 003715      BLE      RIEND      ; BR IF NOT
42 012620 105021 5$:  CLRB    (R1)+      ; PUT IN PADDING NULLS
43 012622 005302      DEC      R2        ;
44 012624 003375      BGT      5$        ;
45 012626 000711      BR       RIEND      ;

```

```

1          ;
2          ; COROL Indexed organization (ISAM) file record.
3          ;
4 012630 020405 R1CX:  CMP      R4,R5      ; IS BUFFER EMPTY?
5 012632 103403      BLD      3#       ; BR IF NOT
6 012634 004767 000132 CALL   RDFILE    ; READ IN NEW BUFFER FULL
7 012640 103715      BCS      RIEOF     ; BR IF END OF INPUT HIT
8          ; See if record exists.
9 012642 005724 3#:   TST      (R4)+     ; TEST RECORD CONTROL WORD
10 012644 001002      BNE      4#       ; BR IF RECORD IS DELETED
11 012646 005714      TST      (R4)     ; TEST FOR END OF DELETE CHAIN
12 012650 001006      BNE      RCXE     ; BR IF RECORD EXISTS
13          ; Record is deleted. Skip to start of next record.
14 012652 016703 000000G 4#:   MOV      ISMARS,R3    ; GET LENGTH OF RECORD TO SKIP OVER
15 012656 004767 000056      CALL   SKPXRC    ; SKIP OVER THE RECORD
16 012662 103704      BCS      RIEOF     ; BR IF END OF INPUT HIT
17 012664 000766      BR       3#       ; GO CHECK NEXT RECORD
18          ; Record exists. Move it to internal record buffer.
19 012666 016703 000000G RCXE:  MOV      INRLEN,R3    ; GET LOGICAL RECORD LENGTH (BYTES)
20 012672 006203      ASR      R3       ; CONVERT TO # WORDS
21 012674 020405 3#:   CMP      R4,R5     ; IS INPUT BUFFER EMPTY?
22 012676 103403      BLD      2#       ; BR IF NOT
23 012700 004767 000066      CALL   RDFILE    ; READ IN A NEW BUFFER FULL
24 012704 103673      BCS      RIEOF     ; BR IF END OF FILE HIT
25 012706 012421 2#:   MOV      (R4)+,(R1)+ ; MOVE RECORD TO INTERNAL BUFFER
26 012710 005303      DEC      R3       ; MORE TO MOVE?
27 012712 003370      BGT      3#       ; BR IF YES
28          ; Skip over any accession numbers at end of record.
29 012714 016703 000000G      MOV      ISMARS,R3    ; GET TOTAL RECORD LENGTH
30 012720 166703 000000G      SUB      INRLEN,R3    ; SUBTRACT LOGICAL RECORD LENGTH
31 012724 001402      BEQ      4#       ; BR IF NO ACCESSION NUMBERS
32 012726 004767 000006      CALL   SKPXRC    ; SKIP OVER ACCESSION NUMBERS
33          ; Increment record counter.
34 012732 004767 001614 4#:   CALL   INCNUM    ; INCREMENT RECORD COUNTER
35          ; Finished
36 012736 000645      BR       RIEND
  
```

```

1      ;-----
2      ; SKPXRC is called to skip over a part of an ISAM record.
3      ;
4      ; Inputs:
5      ;   R3 = Number of bytes to skip over.
6      ;   R4 = Pointer to start of data to be skipped over.
7      ;   R5 = Pointer to end of buffer.
8      ;
9      ; Outputs:
10     ;   R4 = Pointer to start of data following skipped data.
11     ;   R5 = Pointer to end of buffer.
12     ;   C-flag set on return if end of input hit.
13     ;
14 012740 010346 SKPXRC: MOV     R3, -(SP)
15 012742 060403 1$:   ADD     R4, R3           ; POINT TO START OF NEXT RECORD
16 012744 020305           CMP     R3, R5           ; IS IT PAST END OF THE BUFFER?
17 012746 103405           BLD     2$,           ; BR IF NOT
18 012750 160503           SUB     R5, R3           ; SUBTRACT AMT WE ARE SKIPPING OVER IN THIS BUFFER
19 012752 004767 000014   CALL   RDFILE          ; READ IN NEW BUFFER FULL
20 012756 103403           BCS    4$,           ; BR IF END OF INPUT HIT
21 012760 000770           BR     1$,           ; SEE IF WE HAVE REACHED DESIRED POINT
22 012762 010304 2$:   MOV     R3, R4           ; RETURN RECORD POINTER IN R4
23 012764 000241           CLC                    ; SIGNAL SUCCESS
24 012766 012603 4$:   MOV     (SP)+, R3
25 012770 000207           RETURN
  
```

```
1 ;-----  
2 ; RDFILE is called to read the next buffer full of data.  
3 ; If end of file is hit, NEXTIN is called to try to switch to the  
4 ; next input file.  
5 ;  
6 ; Outputs:  
7 ; R4 = Pointer to start of buffer read.  
8 ; R5 = Pointer to end of buffer.  
9 ; C-flag set on return if end of all input was hit.  
10 ;  
11 012772 004767 000012 RDFILE: CALL RIBUF ;TRY TO READ NEXT BUFFER FULL  
12 012776 103003 BCC 1$ ;BR IF GOT ONE  
13 ; Hit end of file.  
14 ; Call NEXTIN to switch to next input file.  
15 013000 005003 CLR R3 ;ZERO ANY REMAINING SKIP BYTES  
16 013002 004767 000376 CALL NEXTIN ;TRY TO SWITCH TO NEXT INPUT FILE  
17 013006 000207 1$: RETURN
```

```

1 ; -----
2 ; Read the next input buffer full.
3 ; On return R4 = Address of start of buffer, R5 = Address of end of buffer.
4 ; The c-flag is set on return if end of file is hit.
5 ;
6 013010 RIBUF: $WAIT #FD1 ;WAIT FOR PREVIOUS READ TO FINISH
7 013124 103006 BCC 6$ ;BR IF NO ERROR
8 013126 105700 TSTB R0 ;REAL ERROR?
9 013130 001404 BEQ 6$ ;BR IF NOT
10 013132 XXX 14. ;ERROR ON READ
11 ; Switch buffers and start next read.
12 013142 6$:
13 013142 042700 000377 BIC #377,R0 ;CLEAR GARBAGE
14 013146 006300 ASL R0 ;CVT # WORDS READ TO # BYTES READ
15 013150 010067 174510 MOV R0,INRDSZ ;SAVE # BYTES ACQUIRED BY THIS READ
16 013154 066767 0000000 0000000 ADD BLKFC,IFPNT ;ADVANCE FILE BLOCK #
17 013162 016704 174456 MOV IBUF2,R4 ;ADDRESS OF BUFFER WE JUST READ INTO
18 013166 016767 174450 174450 MOV IBUF1,IBUF2 ;BUFFER WE WILL START NEW READ INTO
19 013174 010467 174442 MOV R4,IBUF1
20 013200 016705 174460 MOV INRDSZ,R5 ;GET # BYTES READ BY LAST READ
21 013204 001475 BEQ 3$ ;BR IF END OF FILE HIT
22 013206 060405 ADD R4,R5 ;GET ADDRESS PAST END OF BUFFER
23 013210 010567 174444 MOV R5,IBFEND ;SAVE ADDRESS OF END OF BUFFER
24 013214 1$: $READ #FD1,IBUF2,OWCNT,IFPNT
25 013374 000241 2$: CLC ;SIGNAL NO END OF FILE
26 013376 000207 9$: RETURN
27 ; End of file hit.
28 013400 000261 3$: SEC ;SIGNAL END OF FILE
29 013402 000775 BR 9$

```

```

1 ; -----
2 ; NEXTIN is called when the end of an input file is reached to try to
3 ; switch to the next input file.
4 ; If there is another input file, it is opened and the C-flag is cleared
5 ; on return. R4 = Address of buffer start, R5 = Address of buffer end.
6 ; If there are no more input files the C-flag is set on return.
7 ;
8 013404 NEXTIN: PUSH <R1,R2>
9 ; Close last input file.
10 013410 $KEEP #FD1
11 ; See if there is another input file.
12 013456 016702 000000G MOV CURSAV,R2 ;R2 -> TABLE OF -> SPC BLOCKS
13 013462 012201 MOV (R2)+,R1 ;R1 -> NEXT FILE SPEC FOR INPUT
14 013464 001546 BEQ 9# ;EXIT IF NULL POINTER
15 013466 010267 000000G MOV R2,CURSAV ;SAVE -> TABLE ENTRY -> NEXT INPUT FILE SPEC
16 013472 011102 MOV (R1),R2 ;R2 -> REAL FILE SPECIFICATION
17 013474 005712 TST (R2) ;CHECK 1ST WORD OF FILE NAME
18 013476 001541 BEQ 9# ;EXIT ON BLANK NAME
19 ;
20 ; There is another input file.
21 ; Open it.
22 ;
23 013500 $REOPEN #FD1,R1 ;REOPEN NEXT FILE
24 ;
25 ; Start reads from new file.
26 ;
27 013564 005067 000000G CLR IFPNT ;START WITH BLOCK 0 OF FILE
28 013570 105067 174104 1#: CLRB EOFLG ;NO EOF YET
29 013574 $READ #FD1,IBUF2,OWCNT,IFPNT
30 013754 004767 177030 CALL RIBUF ;START NEXT READ-AHEAD AND GET BUFFER POINTERS
31 013760 010467 173676 MOV R4,IRPOS ;POINTER TO START OF BUFFER
32 013764 010567 173670 MOV R5,IBFEND ;POINTER TO END OF BUFFER
33 013770 016767 000000G 000000G MOV SKPVAL,SKPCNT ;RESET RECORD-SKIP COUNTER
34 ; Return with C-flag cleared.
35 013776 000241 CLC ;SIGNAL ANOTHER FILE AVAILABLE
36 014000 000404 BR 10#
37 ;
38 ; No more input files. Return with C-flag set.
39 ;
40 014002 162767 000002 000000G 9#: SUB #2,CURSAV ;BACKUP FILE POINTER
41 014010 000261 SEC ;SIGNAL NO MORE FILES AVAILABLE
42 014012 10#: POP <R2,R1>
43 014016 000207 RETURN

```

```

1      ;
2      ;
3      ; SELECT is called to determine if the current record should be
4      ; included in the sort.
5      ; When called, R1 must point to the start of the buffer containing
6      ; the record.
7      ; On return, the C-flag is set if the record is to be rejected.
8      ;
9 014020 010167 173650 SELECT: MOV R1,SELREC ;REMEMBER ADDRESS OF BUFFER
10 014024 010246      MOV R2,-(SP)
11 014026 010346      MOV R3,-(SP)
12 014030 010446      MOV R4,-(SP)
13 014032 010546      MOV R5,-(SP)
14 014034 016705 000000G MOV SELADR,R5 ;POINT TO 1ST SELECT CONTROL BLOCK
15 014040 012767 007707' 173630 MOV #LSTKTP,LSTACK ;INIT LOGIC STACK
16      ;
17      ; Interpret operation for next select control block.
18      ;
19 014046 116500 000000G SLINTP: MOVB S%OPR(R5),RO ;GET NEXT OPERATION FUNCTION CODE
20 014052 020027 000000G      CMP RO,#S0%CMP ;FIELD COMPARISION?
21 014056 001434      BEQ SLCMP ;BR IF YES
22 014060 020027 000000G      CMP RO,#S0%OR ;LOGICAL "OR"?
23 014064 001422      BEQ SLOR ;BR IF YES
24 014066 020027 000000G      CMP RO,#S0%AND ;LOGICAL AND?
25 014072 001407      BEQ SLAND ;BR IF YES
26      ;
27      ; Perform logical NOT function
28      ;
29 014074 105177 173576 SLNOT: COMB @LSTACK ;COMPLEMENT LAST LOGIC VALUE
30 014100 142777 000376 173570      BICB #376,@LSTACK ;CLEAR ALL BUT LOW ORDER BIT
31 014106 000167 000212      SLJNXD: JMP SLJNXD
32      ;
33      ; Perform logical AND operation.
34      ;
35 014112 117700 173560 SLAND: MOVB @LSTACK,RO ;GET LOGICAL VALUE ON TOP OF STACK
36 014116 005267 173554      INC LSTACK ;POP STACK
37 014122 005100      COM RO ;PERFORM LOGICAL AND OPERATION WITH
38 014124 140077 173546      BICB RO,@LSTACK ;LOGICAL VALUE NEXT LOWER ON THE STACK
39 014130 000766      BR SLJNXD ;GO PROCESS NEXT CONTROL BLOCK
40      ;
41      ; Perform logical OR operation.
42      ;
43 014132 117700 173540 SLOR: MOVB @LSTACK,RO ;GET LOGICAL VALUE ON TOP OF STACK
44 014136 005267 173534      INC LSTACK ;POP STACK
45 014142 150077 173530      BISB RO,@LSTACK ;OR VALUE WITH THAT NEXT LOWER ON STACK
46 014146 000757      BR SLJNXD
47      ;
48      ; Perform field comparison.
49      ;
50 014150 016701 173520 SLCMP: MOV SELREC,R1 ;POINT TO BUFFER WITH RECORD
51 014154 066501 000000G      ADD S%POS1(R5),R1 ;POINT TO FIELD WITHIN RECORD
52 014160 016502 000000G      MOV S%POS2(R5),R2 ;POINT TO 2ND FIELD
53 014164 132765 000000G 000000G      BITB #SF$LIT,S%FLG(R5); IS 2ND FIELD A LITERAL OR ANOTHER FIELD?
54 014172 001002      BNE 1$ ;BR IF IT IS A LITERAL
55 014174 066702 173474      ADD SELREC,R2 ;GET ADDRESS OF 2ND FIELD WITHIN RECORD
56 014200 016503 000000G 1$: MOV S%SZ1(R5),R3 ;GET SIZE OF FIELD
57 014204 116500 000000G      MOVB S%TYP(R5),RO ;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'
116 014372 000000G
117 014374 000000
118 014376 000000G
119 014400 000000G
120 014402 000000G
121 014404 000000G
122 014406 000000G
123 014410 000000G

SELVEC: .WORD KYASCI
.WORD KYCOMP
.WORD 0
.WORD KYDATE
.WORD KYBIN
.WORD KYDIS
.WORD KYUCA
.WORD KYLSS
.WORD KYTSS

;FT\$C --- ASCII CHARACTERS
;FT\$B --- SIGNED BINARY INTEGER
;FT\$U --- UNSIGNED BINARY INTEGER
;FT\$T --- DATE (MMDDYY)
;FT\$F --- FLOATING
;FT\$D --- DIBOL INTEGER
;FT\$A --- UPPER CASE ASCII CHARACTERS
;FT\$I --- LEADING SEPARATE SIGN
;FT\$J --- TRAILING SEPARATE SIGN

```
1 ;  
2 ;-----  
3 ; KYASCII is called to compare two ascii character strings.  
4 ; When called, R1 must point to the 1st string and R2 must point  
5 ; to the 2nd string. R3 must contain the number of characters.  
6 ; On return, the conditon code is set for BHI/BLO/BEQ test.  
7 ;  
8 014412 122122 KYASCII: CMPB (R1)+,(R2)+ ;COMPARE CHARACTERS  
9 014414 001002 BNE 1$ ;BR IF NOT EQUAL  
10 014416 005303 DEC R3 ;MORE TO CHECK?  
11 014420 003374 BGT KYASCII ;BR IF YES  
12 014422 000207 1$: RETURN ;RETURN WITH CONDITION CODE SET
```

```

1      ;
2      ;-----
3      ;  DOMAP is called to perform the field mapping function.
4      ;  When called, MAPBUF must point to the start of the buffer containing
5      ;  the input source record and RECPTR must point to the buffer where
6      ;  the destination record is to be stored.  The length of the resulting
7      ;  record must be stored in RECLen.
8      ;
9      014424 010146  DOMAP:  MOV     R1, -(SP)
10     014426 010246      MOV     R2, -(SP)
11     014430 010346      MOV     R3, -(SP)
12     014432 010446      MOV     R4, -(SP)
13     ;
14     ;  Blank or null fill the result buffer.
15     ;
16     014434 012700 000040      MOV     #' ,R0      ; ASSUME SPACE FILL WANTED
17     014440 032767 000000G 000000G  BIT     #RT#RB, ITYPE ; IS FILE RECORDING MODE BINARY?
18     014446 001401      BEQ     1#          ; BR IF ASCII
19     014450 105000      CLRB   R0          ; NULL FILL
20     014452 016703 000000G 1#:   MOV     RECLen, R3      ; GET LENGTH OF RESULT BUFFE
21     014456 016702 173210      MOV     RECPTR, R2      ; POINT TO RESULT BUFFER
22     014462 110022 2#:   MOVB   R0, (R2)+      ; FILL WITH SPACES OR NULLS
23     014464 005303      DEC     R3            ; COUNT # BYTES MOVED
24     014466 001375      BNE    2#            ; BR IF MORE TO DO
25     ;
26     ;  Now move source fields to destination record.
27     ;
28     014470 016704 000000G      MOV     FLDMap, R4      ; POINT TO 1ST MAP CONTROL BLOCK
29     ;  Move next field.
30     014474 016401 000000G  MAPMOV: MOV     M#POS1(R4), R1      ; GET POSITION OF SOURCE FIELD
31     014500 066701 173164      ADD     MAPBUF, R1      ; GET ABSOLUTE ADDRESS
32     014504 016402 000000G      MOV     M#POS2(R4), R2      ; GET POSITION OF DESTINATION FIELD
33     014510 066702 173156      ADD     RECPTR, R2      ; GET ABSOLUTE ADDRESS
34     014514 016403 000000G      MOV     M#SIZE(R4), R3      ; GET # BYTES TO MOVE
35     014520 112122 4#:   MOVB   (R1)+, (R2)+      ; MOVE THE FIELD
36     014522 005303      DEC     R3            ; MORE TO MOVE?
37     014524 003375      BGT    4#            ; BR IF YES
38     ;  Move on to next map field descriptor.
39     014526 162704 000000G 5#:   SUB     #M#SZ, R4      ; POINT TO NEXT DESCRIPTOR
40     014532 020467 000000G      CMP     R4, MAPEND      ; DONE ALL?
41     014536 103356      BHS    MAPMOV         ; BR IF MORE TO DO
42     ;
43     ;  Mapping is finished
44     ;
45     014540 012604  MAPFIN: MOV     (SP)+, R4
46     014542 012603      MOV     (SP)+, R3
47     014544 012602      MOV     (SP)+, R2
48     014546 012601      MOV     (SP)+, R1
49     014550 000207      RETURN

```

```

1 ;-----
2 ; INCNUM is called to increment the BCD record number counter.
3 ;
4 014552 005767 000000G INCNUM: TST    NUMLEN    ; IS RECORD NUMBERING WANTED?
5 014556 001415          BEQ    9$          ; GET OUT FAST IF NOT
6 014560 010146          MOV    R1, -(SP)
7 014562 012701 000010G      MOV    #NUMBUF+8, R1 ; POINT PAST END OF NUMBER
8 014566 114100          2$:  MOVB  -(R1), R0    ; GET A DIGIT
9 014570 005200          INC    R0          ; INCREMENT IT
10 014572 120027 000072     CMPB  R0, #72       ; DID WE JUST INCREMENT PAST 9?
11 014576 103403          BLO   1$          ; BR IF NOT
12 014600 112711 000060     MOVB  #'0, (R1)    ; REPLACE DIGIT WITH ZERO
13 014604 000770          BR    2$          ; AND GO INCREMENT NEXT DIGIT TO LEFT
14 014606 110011          1$:  MOVB  R0, (R1)    ; INSERT INCREMENTED DIGIT
15 014610 012601          MOV    (SP)+, R1
16 014612 000207          9$:  RETURN
17 ;-----
18 ;
19 ; INSNUM is called to insert the BCD record number into the current record.
20 ; Inputs:
21 ;   RECPTR = Pointer to start of current record.
22 ;   NUMCOL = Column (byte) number where number insert is to begin.
23 ;   NUMLEN = Number of columns to be used by number.
24 ;   NUMBUF = Buffer where 8 digit BCD number is stored.
25 ;
26 014614 010146          INSNUM: MOV    R1, -(SP)
27 014616 010246          MOV    R2, -(SP)
28 014620 010346          MOV    R3, -(SP)
29 014622 016701 173044     MOV    RECPTR, R1   ; ADDRESS OF START OF RECORD
30 014626 066701 000000G   ADD    NUMCOL, R1   ; ADDRESS OF START OF NUMBER INSERT AREA
31 014632 016703 000000G   MOV    NUMLEN, R3   ; LENGTH OF NUMBER FIELD
32 014636 060301          ADD    R3, R1       ; ADDRESS PAST END OF NUMBER INSERT AREA
33 014640 012702 000010G   MOV    #NUMBUF+8, R2 ; POINT PAST END OF NUMBER COUNTER FIELD
34 014644 010300          MOV    R3, R0       ; GET DESIRED NUMBER LENGTH
35 014646 020027 000010   CMP    R0, #8       ; ONLY GOT 8 DIGITS IN BUFFER
36 014652 101402          BLOS  1$          ; BR IF OK
37 014654 012700 000010   MOV    #8, R0       ; ONLY MOVE 8 DIGITS
38 014660 160003          1$:  SUB    R0, R3       ; # FILLER ZEROES NEEDED
39 014662 114241          2$:  MOVB  -(R2), -(R1) ; MOVE RECORD # TO RECORD
40 014664 005300          DEC    R0
41 014666 003375          BGT   2$
42 014670 005703          TST   R3            ; DO WE NEED TO INSERT EXTRA ZEROS
43 014672 001405          BEQ   4$          ; BR IF NOT
44 014674 112700 000060   MOVB  #'0, R0       ; GET ZERO CHARACTER
45 014700 110041          3$:  MOVB  R0, -(R1)    ; INSERT FILLER ZEROES
46 014702 005303          DEC    R3
47 014704 003375          BGT   3$
48 014706 012603          4$:  MOV    (SP)+, R3
49 014710 012602          MOV    (SP)+, R2
50 014712 012601          MOV    (SP)+, R1
51 014714 000207          RETURN

```

```

1          ; -----
2          ; STRIP is called to strip trailing spaces off the ends of ascii records.
3          ; Inputs:
4          ;   RECPTR = Address of start of current record.
5          ;   RECLEN = Length of current record.
6          ;
7 014716 010146 STRIP:  MOV     R1, -(SP)
8 014720 010246      MOV     R2, -(SP)
9 014722 016701 172744      MOV     RECPTR, R1      ; GET ADDRESS OF FRONT OF RECORD
10 014726 016702 000000G      MOV     RECLEN, R2      ; GET LENGTH OF RECORD
11 014732 060201      ADD     R2, R1      ; GET ADDRESS OF END OF RECORD
12 014734 112700 000040      MOVB   #' ', R0      ; GET SPACE CHAR FOR COMPARISON
13 014740 124100 1#:  CMPB   -(R1), R0      ; IS NEXT CHAR A SPACE?
14 014742 001003      BNE    2#      ; QUIT IF NOT
15 014744 105011      CLRB   (R1)      ; REPLACE WITH A NULL
16 014746 005302      DEC    R2      ; MORE TO CHECK?
17 014750 003373      BGT    1#      ; BR IF YES
18 014752 012602 2#:  MOV     (SP)+, R2
19 014754 012601      MOV     (SP)+, R1
20 014756 000207      RETURN
  
```

```

1      ;
2      ;
3      ; EOFSAV is called to save a user EOF record in a holding buffer
4      ; so that it can be written to the end of the output file.
5      ; When called R1 must contain the address of the buffer containing
6      ; the EOF record.
7      ;
8 014760 010146 EOFSAV: MOV      R1, -(SP)
9 014762 010346      MOV      R3, -(SP)
10 014764 010446      MOV      R4, -(SP)
11 014766 016703 000000G      MOV      RECLEN, R3      ; GET LENGTH OF RECORD
12 014772 016704 000000G      MOV      EOFBUF, R4      ; GET ADDRESS OF EOF HOLDING BUFFER
13 014776 112124      1$:      MOVVB   (R1)+, (R4)+      ; MOVE RECORD TO HOLDING BUFFER
14 015000 001004      BNE      2$      ; BR IF NON-NULL CHARACTER
15 015002 032767 000000G 000000G      BIT      #RT$VLN, ITYPE      ; IS THIS A VARIABLE LENGTH RECORD?
16 015010 001002      BNE      9$      ; BR IF YES -- WE JUST HIT END OF IT
17 015012 005303      2$:      DEC      R3      ; NEED TO MOVE MORE?
18 015014 003370      BGT      1$      ; BR IF YES
19 015016 012604      9$:      MOV      (SP)+, R4
20 015020 012603      MOV      (SP)+, R3
21 015022 012601      MOV      (SP)+, R1
22 015024 000207      RETURN

```

```
1 ;  
2 ;-----  
3 ;  
4 ; PASS1 TERMINATION ROUTINE.  
5 ;  
6 015026 016703 0000009 S1STOP: MOV NXTBLK,R3 ;GET NXT BLOCK #.  
7 015032 062703 000002 ADD #2,R3 ;LEAVE A LITTLE EXTRA ROOM.  
8 015036 010367 0000009 MOV R3,NXTBLK  
9 015042 004767 0000009 CALL OUTWRT ;WRITE TO LAST BLK SO MONITOR WILL  
10 015046 000167 0000009 JMP P1TERM ;RESERVE ROOM IN FILE.  
11  
12 ;  
13 015054' P1TOP = .+2  
14 015054 P1SIZ = P1TOP-P1BASE  
15 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 SLJNXD 014106R	002
BIGEST= 177777	HB\$DIP= 000030	MXREC1= ***** G	RIX 012630R	002 SLNOT 014074R	002
BLKFC = ***** G	HB\$DUP= 000001	MXREC2= ***** G	RIEND 012452R	002 SLNXP 014324R	002
BT\$HI = ***** G	HB\$RIU= 000032	M\$POS1= ***** G	RIEDF 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
CHND = 000000	IBUF1 007642R	002 NDERR = 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	INFLN= ***** G	NUMREC 007640R	002 RSTSJB= 000404	SR2 010532R	002
CURDFL= ***** 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	RECLN= ***** 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, DVR)
 000000 001 (RW, I, LCL, REL, CON)
 SORT1 015052 002 (RW, I, GBL, REL, DVR)

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 15783 Words (52 Pages)

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

RWDLEN	2-35	4-30	4-33	4-41						
S#BR	2-34	18-70								
S#FLQ	2-31	18-53								
S#LINK	2-32	18-72								
S#OPR	2-50	18-19								
S#POS1	2-50	18-51								
S#POS2	2-50	18-52								
S#SIZ1	2-51	18-56								
S#TYP	2-51	18-57								
S1STOP	5-56	24-6#								
SELADR	2-47	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	2-51	18-53								
SKPCNT	2-36	6-52	6-54*	17-33*						
SKPVAL	2-55	17-33								
SKPXRC	13-15	13-32	14-14#							
SLACPT	18-36	18-100	18-105#							
SLAND	18-25	18-35#								
SLCMP	18-21	18-50#								
SLFALS	18-75	13-79#								
SLINTP	18-17#	18-94								
SLJNXD	18-31#	18-39	18-46							
SLNOT	18-29#									
SLNXOP	18-31	18-83	18-92#							
SLOR	18-23	18-43#								
SLREJT	18-60	18-102#								
SLRTN	18-103	18-106#								
SLTRUE	18-74	18-77	18-85#							
SO#AND	2-32	18-24								
SO#CMP	2-30	18-20								
SO#NOT	2-32									
SO#OR	2-50	18-22								
SORTP1	2-27	4-5#								
SR2	5-5#	5-137								
SR3	5-7	5-47#								
SR4	5-48	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#							

.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						