SORT2 SORT -- PHASE 2 MACRO V05.01b Monday 06-May-85 17:46
Table of contents

```
                                        .TITLE  SORT2   SORT -- PHASE 2
000000                                  .CSECT  SORT2
                                        .ENABL  LC
                000000'         P2BASE  =       .
                                ;
                                ;  THIS SECTION OF SORT REPEATEDLY MERGES THE SORTED
                                ;  STRINGS PRODUCED BY PASS 1 UNTIL ONLY A SINGLE SORTED
                                ;  STRING REMAINS.  THE FINAL MERGE PASS WRITES THIS STRING
                                ;  TO THE OUTPUT FILE.
                                ;
                                ;  Copyright (c) 1980, 1981.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee USA
                                ;
                                ;  All rights reserved.
                                ;  This software may not be sold, distributed or otherwise made
                                ;  available for use except as licensed by S&H Computer Systems, Inc.
                                ;
                                ;  COBOL-Plus, SORT-Plus, RTSORT, TSX and TSX-Plus are registered
                                ;  trademarks of S&H Computer Systems, Inc.
                                ;
                                ;
                                ;
                                ;  MACRO TO CAUSE ABORT
                                ;
                                        .MACRO  XXX     COD
                                XXX'COD =       .
                                        MOV     #COD'.,R0
                                        CALL    ABORT
                                        .ENDM   XXX
                                ;
                                ;  GLOBAL DEFINITIONS.
                                ;
                                        .GLOBL          SORTP2,P2TOP,P2SIZ
                                ;
                                ;  GLOBAL REFERENCES.
                                ;
                                        .GLOBL  RWDLEN,ABORT,FD3,CISRT2
                                        .GLOBL  KTYPE,TOPMEM,RECLEN,EOFBUF
                                        .GLOBL  FILRN,FILST,MXFILS,MXFIL2
                                        .GLOBL  FILNB,OBFEND,OBUFSZ
                                        .GLOBL  NUMKY2,OBUF1,OBUF2,OBUFSZ
                                        .GLOBL  NXTBLK,ORPOS,FLUSH,COMPAR
                                        .GLOBL  DSKCVT,FLBLK,FILFL
                                        .GLOBL  CUROFL,FT$T,KSTRT
                                        .GLOBL  BLKFC,P2TERM,OWCNT
                                        .GLOBL  ITYPE,RT$CR,RT$VLN
                                        .GLOBL  RT$CBL,RT$DBL,RT$FTN,RT$TXT
                                        .GLOBL  RT$OS,RT$OR,RT$OX,RT$F11
                                        .GLOBL  RT$RA,RT$RB
                                        .GLOBL  OTYPE,F$INPL
                                        .GLOBL  UEOF,SFLAGS
                                ;
                                ;
                                ;  FILE MANAGEMENT TABLES.
                                ;
000050          000050          MXMRG   =       40.             ;ABSOLUTE MAX # OF FILES TO BE MERGED.
```

```
                        000120          MXMRG2  =       2*MXMRG
                                        ;
000000                  MRGLST: .BLKW   MXMRG           ;LIST OF FILES TO MERGE
000120                  LOC:    .BLKW   MXMRG           ;ADDRESS OF RECORD(J).
000240                  LOSER:  .BLKW   MXMRG           ;INDEX TO LOSER BELOW THIS NODE.
000360                  FE:     .BLKW   MXMRG           ;EXTERNAL BLOCK FORWARD LINK.
000500                  FI:     .BLKW   MXMRG           ;INTERNAL BLOCK FORWARD LINK.
000620                  RN:     .BLKW   MXMRG           ;RUN # THIS RECORD WILL GO IN.
000740                  IRPOS:  .BLKW   MXMRG           ;CUR POINTER INTO BUFFER
001060                  MRGBA:  .BLKW   MXMRG           ;ADDRESS OF FILE BUFFER
                                        ;
                                        ;   GENERAL DATA.
                        000015          CR      =       15              ;ASCII CARRIAGE RETURN
                        000012          LF      =       12              ;ASCII LINE FEED
                        000032          CTRLZ   =       32              ;CTRL-Z DIBOL EOF CHARACTER
001200  000000          GRNTBL: .WORD   0               ;ADDRESS OF START OF GRANULE TBL.
001202  000000          GRNEND: .WORD   0               ;END OF GRANULE TABLE.
001204  000000          NFILE2: .WORD   0               ;MAX # OF FILES WHICH CAN BE MERGED.
001206  000000          NUMMRG: .WORD   0               ;# OF FILES TO MERGE NEXT PASS.
001210                  EMTBLK: .BLKW   10              ;SPACE FOR EMT ARGS
001230  000000          RC:     .WORD   0               ;RUN # OF CURRENT RUN.
001232  000000          Q:      .WORD   0               ;INDEX TO CURRENT RECORD.
001234  000000          RQ:     .WORD   0               ;RUN # CURRENT RECORD WILL GO IN.
001236  000000          NXOBLK: .WORD   0               ;ADDRESS OF NXT BLOCK TO OUTPUT.
001240  000000          OFLINK: .WORD   0               ;LINK TO PREV RUNS IN FILE
                                        ;
001242  000             FINFLG: .BYTE   0               ;1==>THIS IS FINAL MERGE.
                                .EVEN
```

```
                                        .SBTTL  PASS 2 INITIALIZATION
                                ;
                                ;  ALLOCATE AVAILABLE CORE SPACE.
                                ;
001244  004767  000000G      SORTP2: CALL    CISRT2          ;CHECK IN FOR OVERLAY BREAKPOINT
001250  012701  006424'              MOV     #P2TOP,R1       ;GET START OF FREE MEMORY SPACE.
                                ;  SET UP GRANULE BIT TABLE.
                                ;  A 1 IN THIS TABLE INDICATES THE CORRESPONDING
                                ;  DISK GRANULE IS FREE.
001254  010167  177720                  MOV     R1,GRNTBL       ;SET START OF TABLE ADDRESS.
001260  016702  000000G                 MOV     NXTBLK,R2       ;GET # OF DISK BLKS TO ALLOCATE.
001264  062702  000017                  ADD     #15.,R2         ;CONVERT # OF BITS TO # OF WORDS.
001270  006202                          ASR     R2
001272  006202                          ASR     R2
001274  006202                          ASR     R2
001276  060102                          ADD     R1,R2           ;GET ADDRESS OF END OF TABLE.
001300  010267  177676                  MOV     R2,GRNEND
001304  005021                  1$:     CLR     (R1)+           ;MARK ALL GRANULES AS BUSY.
001306  020102                          CMP     R1,R2
001310  103775                          BLO     1$
                                ;  THE LAST 2 GRANULES ARE ACTUALLY FREE
001312  012702  000002                  MOV     #2,R2
001316  016704  000000G                 MOV     NXTBLK,R4
001322  004767  004662          6$:     CALL    RELGRN          ;FREE THOSE GRANULES.
001326  005304                          DEC     R4
001330  005302                          DEC     R2
001332  003373                          BGT     6$
                                ;  ALLOCATE FILE I/O BUFFERS.
001334  016703  000000G                 MOV     OBUFSZ,R3       ;GET SIZE OF TEMP FILE BUFFER.
001340  010167  000000G          3$:     MOV     R1,OBUF1        ;ALLOCATE OUTPUT BUFFERS.
001344  060301                          ADD     R3,R1
001346  010167  000000G                 MOV     R1,OBFEND       ;END OF BUFF1
001352  010167  000000G                 MOV     R1,OBUF2
001356  060301                          ADD     R3,R1
                                ;
                                ;  ALLOCATE FILE INPUT BUFFERS
                                ;
001360  005002                          CLR     R2              ;COUNT # OF FILES ALLOCATED
001362  010162  000120'          4$:     MOV     R1,LOC(R2)      ;ALLOCATE RECORD BUFFER
001366  066701  000000G                 ADD     RWDLEN,R1
001372  010162  001060'                 MOV     R1,MRGBA(R2)    ;ALLOCATE FILE BUFFER
001376  066701  000000G                 ADD     OBUFSZ,R1
001402  020167  000000G                 CMP     R1,TOPMEM       ;HIT TOP OF MEMORY?
001406  101005                          BHI     2$              ;BR IF YES
001410  062702  000002                  ADD     #2,R2           ;COUNT ANOTHER FILE ALLOCATED
001414  020227  000120                  CMP     R2,#MXMRG2      ;ALLOCATED MAX ALLOWED?
001420  103760                          BLO     4$              ;BR IF NOT
001422  006202                  2$:     ASR     R2              ;GET # OF FILES
001424  010267  177554                  MOV     R2,NFILE2       ;SAVE # OF FILES ALLOCATED
001430  005067  177600                  CLR     RQ              ;INIT MISC PARAMETERS
001434  005067  177570                  CLR     RC
001440  005067  177566                  CLR     Q
001444  005067  177566                  CLR     NXOBLK
001450  105067  177566                  CLRB    FINFLG
```

```
                        .SBTTL  DECIDE STRATEGY FOR NEXT MERGE PASS
                ;
                ;  THIS SECTION OF SORT DETERMINES THE STRATEGY
                ;  TO BE CARRIED OUT BY THE NEXT MERGE PASS.
                ;  ON ENTRY TO DOMRG THE CELL NUMMRG CONTAINS
                ;  THE NUMBER OF FILES TO MERGE.  FINFLG IS SET
                ;  TO 1 IF THIS IS THE FINAL MERGE.
                ;
001454  005002          STRAT:  CLR     R2              ;WILL GET MAX # OF RUNS IN ANY FILE
001456  005003                  CLR     R3              ;WILL GET SUM OF RUNS IN ALL FILES.
001460  012704  000000G         MOV     #FILRN,R4       ;TABLE WITH # OF RUNS IN EACH FILE.
001464  010405                  MOV     R4,R5
001466  062705  000000G         ADD     #MXFIL2,R5      ;GET ADDRESS OF END OF TABLE.
001472  012401          2$:     MOV     (R4)+,R1        ;GET # OF RUNS IN THIS FILE.
001474  020201                  CMP     R2,R1           ;SAVE HIGHEST NUMBER.
001476  103001                  BHIS    1$
001500  010102                  MOV     R1,R2
001502  060103          1$:     ADD     R1,R3           ;GET TOTAL NUMBER OF RUNS.
001504  020405                  CMP     R4,R5           ;FINISHED YET?
001506  103771                  BLO     2$
001510  020227  000002          CMP     R2,#2           ;MAX # OF RUNS > 2?
001514  101030                  BHI     MAJMRG          ;BRANCH IF YES.
001516  001413                  BEQ     PENMRG
001520  020367  177460          CMP     R3,NFILE2       ;CAN WE DO FINAL MERGE?
001524  101010                  BHI     PENMRG          ;BRANCH IF NOT.
                ;  THIS IS FINAL MERGE.
001526  105267  177510          INCB    FINFLG          ;SET FINAL MERGE FLAG.
001532  010367  177450          MOV     R3,NUMMRG       ;SET # OF FILES TO MERGE.
001536  012767  000000G 000000G  MOV     #MXFILS,CUROFL
001544  000442                  BR      STRFIN          ;GO PERFORM MERGE.
                ;  SEE IF THIS IS NEXT TO LAST MERGE.
001546  016701  177432  PENMRG:  MOV     NFILE2,R1
001552  006301                  ASL     R1              ;GET 2 TIMES # OF FILES WE CAN MERGE.
001554  005301                  DEC     R1              ;MINUS 1.
001556  020301                  CMP     R3,R1           ;WILL NEXT PASS FINISH FILE?
001560  103006                  BHIS    MAJMRG          ;BRANCH IF NOT.
                ;  THIS IS THE NEXT TO LAST MERGE.
                ;  MERGE (SUM(RN)-NFILE2+1) FILES.
001562  166703  177416          SUB     NFILE2,R3
001566  005203                  INC     R3
001570  010367  177412          MOV     R3,NUMMRG
001574  000426                  BR      STRFIN          ;GO DO THE MERGE.
                ;  MERGE AS MANY FILES AS POSSIBLE ON THIS PASS.
001576  005001          MAJMRG: CLR     R1
001600  012704  000000G         MOV     #FILRN,R4       ;TABLE OF RUNS PER FILE.
001604  010405                  MOV     R4,R5
001606  062705  000000G         ADD     #MXFIL2,R5      ;ADDRESS OF END OF TABLE.
001612  005724          2$:     TST     (R4)+           ;DOES THIS FILE HAVE SOME RUNS?
001614  001401                  BEQ     1$              ;BRANCH IF IT'S EMPTY.
001616  005201                  INC     R1              ;COUNT # OF FILES THAT AREN'T EMPTY.
001620  020405          1$:     CMP     R4,R5           ;FINISHED?
001622  103773                  BLO     2$
001624  020127  000000G         CMP     R1,#MXFILS      ;MAX MERGE OF (MXFILS-1).
001630  103401                  BLO     3$
001632  005301                  DEC     R1
001634  020167  177344          CMP     R1,NFILE2       ;MAX MERGE OF (NFILE2).
001640  101402                  BLOS    4$
```

DECIDE STRATEGY FOR NEXT MERGE PASS

```
     58 001642  016701  177336                    MOV     NFILE2,R1
     59 001646  010167  177334          4$:       MOV     R1,NUMMRG       ;SET # OF FILES TO BE MERGED.
     60 001652  006367  177330          STRFIN:   ASL     NUMMRG          ;SET 2* # OF FILES TO MERGE
```

```
                                        .SBTTL  INITIALIZE FOR NEXT MERGE PASS
                                ;
                                ;  THIS SECTION OF SORT USES THE INFORMATION SET UP BY
                                ;  THE STRATEGY DECISION MAKER TO INITIALIZE TABLES WHICH
                                ;  ARE USED TO DO THE ACTUAL MERGE.
                                ;
001656  105767  177360          DOMRG:  TSTB    FINFLG          ;FINAL MERGE?
001662  001023                          BNE     9$              ;BRANCH IF YES.
                                ;  FIND WHICH FILE TO USE FOR OUTPUT.
                                ;  USE THE FILE WITH THE SMALLEST NUMBER OF RUNS IN IT.
001664  012701  001747                  MOV     #999.,R1
001670  005002                          CLR     R2
001672  020162  000000G         2$:     CMP     R1,FILRN(R2)
001676  101404                          BLOS    1$
001700  010205                          MOV     R2,R5           ;SAVE INDEX TO FILE
001702  016201  000000G                 MOV     FILRN(R2),R1    ;GET # OF SORTED RUNS IN FILE
001706  001405                          BEQ     10$             ;CAN'T BEAT ZERO
001710  062702  000002          1$:     ADD     #2,R2           ;MOVE ON TO NEXT FILE
001714  020227  000000G                 CMP     R2,#MXFIL2      ;CHECKED ALL?
001720  103764                          BLO     2$
001722  010567  000000G         10$:    MOV     R5,CUROFL       ;SAVE INDEX TO OUTPUT FILE.
001726  005265  000000G                 INC     FILRN(R5)       ;ANOTHER RUN GOES TO THIS FILE.
001732  016701  000000G         9$:     MOV     OBUF1,R1        ;GET ADDRESS OF 1ST OUTPUT BUF.
001736  105767  177300                  TSTB    FINFLG          ;IS THIS FINAL MERGE?
001742  001002                          BNE     8$              ;BRANCH IF YES.
001744  005021                          CLR     (R1)+           ;ZERO FLINK FIELD.
001746  005021                          CLR     (R1)+           ;ZERO # OF BYTES IN BLOCK.
001750  010167  000000G         8$:     MOV     R1,ORPOS        ;SAVE ADDRESS WHERE NXT REC GOES.
                                ;  FIND THE NUMMRG/2 FILES WITH THE LARGEST NUMBER OF RUNS
                                ;  TO USE AS INPUT.
001754  005001                          CLR     R1              ;# OF FILES WE HAVE GOTTEN.
001756  005002          7$:     CLR     R2              ;# OF FILE TO CHECK NEXT.
001760  005003                          CLR     R3
001762  012700  177777                  MOV     #177777,R0      ;LOOK FOR SMALLEST # OF BLOCKS
001766  020362  000000G         6$:     CMP     R3,FILRN(R2)    ;SAVE LARGEST NUMBER OF RUNS.
001772  101027                          BHI     3$
001774  103405                          BLO     11$             ;TAKE IT IF MORE RUNS
001776  005703                          TST     R3              ;IS THIS FIRST TIME THROUGH?
002000  001424                          BEQ     3$              ;BR IF YES
002002  020062  000000G                 CMP     R0,FILNB(R2)    ;SELECT FILE WITH SMALLEST # BLOCKS
002006  101421                          BLOS    3$
002010  020267  000000G         11$:    CMP     R2,CUROFL       ;IS THIS THE OUTPUT FILE?
002014  001416                          BEQ     3$              ;IF YES THEN CAN'T USE FOR INPUT.
002016  005004                          CLR     R4              ;SEE IF FILE IS ALREADY IN IN LIST.
002020  020401          4$:     CMP     R4,R1           ;AT END OF LIST?
002022  001406                          BEQ     5$              ;BRANCH IF YES.
002024  020264  000000'                 CMP     R2,MRGLST(R4)   ;SEE IF IN LIST.
002030  001410                          BEQ     3$              ;IF YES THEN DON'T GET IT AGAIN.
002032  062704  000002                  ADD     #2,R4
002036  000770                          BR      4$
002040  016203  000000G         5$:     MOV     FILRN(R2),R3    ;SAVE # OF RUNS IN THIS FILE.
002044  016200  000000G                 MOV     FILNB(R2),R0    ;GET # OF BLOCKS IN FILE
002050  010205                          MOV     R2,R5           ;SAVE INDEX TO FILE.
002052  062702  000002          3$:     ADD     #2,R2           ;MOVE ON TO NEXT FILE
002056  020227  000000G                 CMP     R2,#MXFIL2      ;CHECKED ALL FILES?
002062  103741                          BLO     6$              ;BRANCH IF NOT.
002064  010561  000000'                 MOV     R5,MRGLST(R1)   ;ADD FILE TO INPUT LIST.
```

```
INITIALIZE FOR NEXT MERGE PASS

     58 002070  062701  000002                  ADD     #2,R1           ;COUNT UP 1 MORE FILE
     59 002074  020167  177106                  CMP     R1,NUMMRG       ;GOT ALL WE NEED?
     60 002100  103726                          BLO     7$              ;BRANCH IF NOT.
     61                                 ;
     62                                 ;  AT THIS POINT WE ARE ABOUT TO DO A MERGE FROM NUMMRG
     63                                 ;  FILES WHOSE INDEX NUMBERS ARE STORED IN MRGLST TO
     64                                 ;  THE OUTPUT FILE WHOSE INDEX NUMBER IS IN CUROFL.
     65                                 ;  FINFLG IS NON-ZERO IF THIS IS THE FINAL MERGE.
     66                                 ;
     67                                 ;  SET UP MERGE TABLES.
     68 002102  016702  177100          MRGSET: MOV     NUMMRG,R2       ;GET # OF FILES TO BE MERGED.
     69 002106  006202                          ASR     R2              ;GET 1* # OF FILES TO BE MERGED
     70 002110  010205                          MOV     R2,R5
     71 002112  005302                          DEC     R2
     72 002114  010503                  3$:     MOV     R5,R3           ;GET P ( NUMBER OF FILES TO MERGE).
     73 002116  060203                          ADD     R2,R3           ;GET P+J (J=INDEX TO FILE).
     74 002120  006203                          ASR     R3              ;(P+J)/2
     75 002122  006303                          ASL     R3              ;CONVERT TO WORD TABLE INDEX.
     76 002124  010204                          MOV     R2,R4           ;GET J.
     77 002126  006204                          ASR     R4              ;GET J/2.
     78 002130  006304                          ASL     R4              ;CONVERT TO WORD TABLE INDEX.
     79 002132  006302                          ASL     R2              ;CONVERT J TO WORD TABLE INDEX.
     80 002134  010362  000360'                 MOV     R3,FE(R2)       ;SET UP EXTERNAL FLINK.
     81 002140  010462  000500'                 MOV     R4,FI(R2)       ;SET UP INTERNAL FLINK.
     82 002144  005062  000620'                 CLR     RN(R2)          ;ZERO RUN # THIS REC WILL GO IN.
     83 002150  010262  000240'                 MOV     R2,LOSER(R2)
     84 002154  006202                          ASR     R2
     85 002156  005302                          DEC     R2
     86 002160  002355                          BGE     3$
     87                                 ;  INITIALIZE FILE BUFFERS.
     88 002162  005001                          CLR     R1
     89 002164  016103  000000'          9$:     MOV     MRGLST(R1),R3   ;GET FILE NUMBER.
     90 002170  016104  001060'                 MOV     MRGBA(R1),R4    ;GET ADDRESS OF BUFFER FOR FILE.
     91 002174  010461  000740'                 MOV     R4,IRPOS(R1)    ;INITIALIZE RECORD POINTER
     92 002200  016324  000000G                 MOV     FILST(R3),(R4)+ ;SET UP STARTING DISK ADDRESS.
     93 002204  005014                          CLR     (R4)            ;ZERO # OF BYTES IN BLOCK.
     94 002206  062701  000002                  ADD     #2,R1
     95 002212  020167  176770                  CMP     R1,NUMMRG       ;INITIALIZED ALL FILES?
     96 002216  002762                          BLT     9$              ;BRANCH IF NOT.
     97 002220  005067  177004                  CLR     RC
     98 002224  005067  177002                  CLR     Q
     99 002230  005067  177000                  CLR     RQ
    100                                 ;
    101                                 ;  SET UP OUTPUT FILE
    102                                 ;
    103 002234  105767  177002                  TSTB    FINFLG          ;IS THIS FINAL MERGE?
    104 002240  001013                          BNE     SR2             ;BR IF YES
    105 002242  016702  000000G                 MOV     CUROFL,R2       ;GET FILE INDEX #
    106 002246  016267  000000G 176764          MOV     FILST(R2),OFLINK;SAVE POINTER TO OTHER RUNS IN FILE
    107 002254  004767  004022                  CALL    GETGRN          ;GET A FREE CLUSTER
    108 002260  010162  000000G                 MOV     R1,FILST(R2)    ;SET AS 1ST BLOCK OF FILE
    109 002264  010167  000000G                 MOV     R1,FILFL
```

```
                                        .SBTTL  DO MERGE PASS
                                ;
                                ;  ENTER ACTUAL MERGE ROUTINE.
                                ;
002270  016701  176734        SR2:    MOV     RC,R1           ;DOES CURRENT RECORD GO
002274  020167  176734                CMP     R1,RQ           ;IN THIS RUN?
002300  001413                        BEQ     SR3             ;BRANCH IF YES.
002302  026727  176726  000001        CMP     RQ,#1           ;HAVE WE HIT THE END OF FILE?
002310  101002                        BHI     2$              ;BR IF YES
002312  005701                        TST     R1              ;IS THE INITIALIZATION RUN?
002314  001402                        BEQ     1$              ;BRANCH IF YES
002316  000167  001006        2$:     JMP     SRFIN           ;NO -- WE'RE FINISHED
002322  016767  176706  176700  1$:     MOV     RQ,RC           ;ADVANCE TO NEXT RUN.
002330  005767  176700        SR3:    TST     RQ              ;INITIALIZATION RUN?
002334  001002                        BNE     1$              ;BRANCH IF NOT
002336  000167  000144                JMP     SR4
                                ;    OUTPUT RECORD(Q).
002342  105767  176674        1$:     TSTB    FINFLG          ;FINAL OUTPUT MERGE?
002346  001051                        BNE     FINMOV          ;BRANCH IF YES.
                                ;
                                ;  WE ARE MERGING TO ANOTHER TEMP FILE
                                ;  MOVE RECORD TO OUTPUT BUFFER.
                                ;
002350  016702  176656        TMPMOV: MOV     Q,R2            ;GET MERGE LIST FILE INDEX #
002354  016203  000120'               MOV     LOC(R2),R3      ;GET ADDR OF WINNER RECORD
002360  016702  000000G               MOV     ORPOS,R2        ;GET POINTER INTO OUTPUT BUFFER
002364  016705  000000G               MOV     OBFEND,R5       ;GET ADDRESS OF END OF BUFFER
002370  010304                        MOV     R3,R4           ;GET ADDRESS OF END OF RECORD
002372  066704  000000G               ADD     RECLEN,R4
002376  020304                2$:     CMP     R3,R4           ;MOVED ALL?
002400  103031                        BHIS    4$              ;BR IF FINISHED MOVE
002402  020205                        CMP     R2,R5           ;HIT END OF BUFFER?
002404  103421                        BLO     1$              ;BR IF NOT
                                ;  OUTPUT BUFFER IS FULL -- WRITE IT OUT
002406  016700  000000G               MOV     OBUF1,R0        ;GET ADDRESS OF BUFFER
002412  160002                        SUB     R0,R2           ;CALC # OF CHARS IN BUFFER
002414  010260  000002                MOV     R2,2(R0)        ;STORE # BYTES IN BUFFER HEADER
002420  004767  003656                CALL    GETGRN          ;GET FREE DISK BLOCK
002424  010167  000000G               MOV     R1,FLBLK        ;SET FILE FLINK
002430  004767  000000G               CALL    FLUSH           ;WRITE OUT BUFFER
002434  016702  000000G               MOV     OBUF1,R2        ;POINT TO START OF NEW RECORD
002440  062702  000004                ADD     #4,R2           ;SKIP OVER BUFFER HEADER WORDS
002444  016705  000000G               MOV     OBFEND,R5       ;GET ADDRESS OF END OF BUFFER
                                ;  CONTINUE MOVING RECORD
002450  112322                1$:     MOVB    (R3)+,(R2)+     ;MOVE RECORD TO FILE BUFFER
002452  001351                        BNE     2$              ;BR IF NON-NULL CHAR
002454  032767  000000G 000000G       BIT     #RT$VLN,ITYPE   ;ARE RECORDS VARIABLE LENGTH?
002462  001745                        BEQ     2$              ;BR IF NOT
002464  010267  000000G        4$:     MOV     R2,ORPOS        ;SAVE OUTPUT BUFFER POINTER
002470  000406                        BR      3R4
                                ;
                                ;  MOVE RECORD TO FINAL OUTPUT BUFFER
                                ;
002472  016702  176534        FINMOV: MOV     Q,R2            ;GET MERGE LIST INDEX #
002476  016201  000120'               MOV     LOC(R2),R1      ;GET ADDRESS OF WINNER RECORD
002502  004767  002260                CALL    OUTREC          ;MOVE RECORD TO OUTPUT BUFFER
                                ;
```

```
                                        ;  GET NEXT RECORD FROM INPUT FILE.
                                        ;
003506  016700  176520          SR4:      MOV     @,R0              ;GET MERGE LIST FILE INDEX #
003512  016001  000120'                   MOV     LOC(R0),R1        ;GET ADDRESS OF RECORD BUFFER
003516  016002  000740'                   MOV     IRPOS(R0),R2      ;GET POINTER INTO INPUT BUFFER
003522  016003  001060'                   MOV     MRGBA(R0),R3      ;GET POINTER TO INPUT BUFFER
003526  066303  000002                   ADD     2(R3),R3          ;GET POINTER TO END OF BUFFER
003532  010105                           MOV     R1,R5             ;GET ADDR OF END OF RECORD
003534  066705  000000G                  ADD     RECLEN,R5
003540  020105                  3$:      CMP     R1,R5             ;MOVED ALL OF RECORD?
003542  103173                           BHIS    6$                ;BR IF YES
003544  020203                  4$:      CMP     R2,R3             ;HIT END OF BUFFER?
003546  103557                           BLO     2$                ;BR IF NOT
                                        ;  HIT END OF BUFFER -- SEE IF THERE ARE MORE RECORDS IN SORTED RUN
003550  016700  176456                   MOV     @,R0              ;GET MERGE LIST FILE INDEX #
003554  016002  001060'                   MOV     MRGBA(R0),R2      ;POINT TO FILE BUFFER
003560  011203                           MOV     @R2,R3            ;GET FILE FLINK FROM HEADER
003562  100570                           BMI     ENDRUN            ;BR IF END OF SORTED RUN
                                        ;  READ IN NEXT BLOCK FROM FILE
003564  010304                           MOV     R3,R4             ;RELEASE NEW GRANULE
003566  004767  003416                   CALL    RELGRN
003572  016004  000000'                   MOV     MRGLST(R0),R4     ;GET REAL FILE INDEX #
003576  005364  000000G                  DEC     FILNB(R4)         ;SAY 1 LESS BLOCK IN FILE
003602  004767  000000G                  CALL    DSKCVT            ;CVT CLUSTER # TO FILE & BLOCK #
003606                                   $READ   R4,R2,QWCNT,R3    ;START READ FOR NEXT BLOCK
003760                                   $WAIT   R4                ;WAIT FOR COMPLETION
003772  010203                           MOV     R2,R3             ;GET ADDRESS OF BUFFER
003774  066203  000002                   ADD     2(R2),R3          ;GET ADDRESS OF END OF BUFFER
004000  062702  000004                   ADD     #4,R2             ;GET ADDR OF START OF DATA
004004  000617                           BR      4$
                                        ;  CONTINUE MOVING RECORD
004006  112221                  2$:      MOVB    (R2)+,(R1)+
004010  001213                           BNE     3$                ;BR IF NON-NULL RECORD
004012  032767  000000G 000000G          BIT     #RT$VLN,ITYPE     ;ARE RECORDS VARIABLE LENGTH?
004020  001607                           BEQ     3$                ;BR IF NOT
                                        ;  PAD END OF VARIABLE LENGTH RECORDS WITH NULLS
004022  020105                  7$:      CMP     R1,R5             ;HIT END OF RECORD?
004024  103002                           BHIS    6$                ;BR IF YES
004026  105021                           CLRB    (R1)+             ;FILL WITH NULLS
004030  000774                           BR      7$
                                        ;  FINISHED RECORD MOVE
004032  016700  176074          6$:      MOV     @,R0              ;GET MERGE LIST INDEX #
004036  010260  000740'                   MOV     R2,IRPOS(R0)      ;SAVE POINTER INTO BUFFER
004042  000414                           BR      SR4L
                                        ;
                                        ;  HIT END OF SORTED RUN
                                        ;
004044  012767  000002  176062  ENDRUN:  MOV     #2,RQ             ;SAY NO MORE RECORDS FROM FILE
004052  016004  000000'                   MOV     MRGLST(R0),R4     ;GET FILE INDEX NUMBER
004056  042703  100000                   BIC     #100000,R3        ;CLEAR EOF FLAG
004062  010364  000000G                  MOV     R3,FILST(R4)      ;SAVE NEW FILE START ADDRESS
004066  005364  000000G                  DEC     FILRN(R4)         ;SAY ONE LESS RUN IN FILE
004072  000403                           BR      SR5
                                        ;
                                        ;  PUT THIS RECORD IN RUN #1
004074  012767  000001  176032  SR4L:    MOV     #1,RQ
                                        ;  COMPARE NEW RECORD WITH PREVIOUS ONES.
```

DO MERGE PASS

```
    115 003262  016701  176024          SR5:    MOV     Q,R1            ;GET INDEX TO NEW RECORD.
    116 003266  016104  000360'                  MOV     FE(R1),R4       ;GET INDEX TO FATHER (T).
    117 003272  005704                  SR6:    TST     R4              ;AT HEAD OF TREE?
    118 003274  001002                          BNE     2$              ;BRANCH IF NOT
    119 003276  000167  177046                  JMP     SR7
    120 003302  016703  176006          2$:     MOV     RQ,R3           ;IS THIS INITIALIZATION RUN?
    121 003306  001435                          BEQ     SR7             ;BRANCH IF YES.
    122 003310  020364  000620'                  CMP     R3,RN(R4)       ;RQ:RN(T).
    123 003314  101014                          BHI     1$              ;BR IF RQ>RN(T).
    124 003316  001031                          BNE     SR7
    125 003320  016401  000240'                  MOV     LOSER(R4),R1    ;GET LOSER(T).
    126 003324  016101  000120'                  MOV     LOC(R1),R1      ;GET LOC(LOSER(T)).
    127 003330  016703  175756                  MOV     Q,R3
    128 003334  016302  000120'                  MOV     LOC(R3),R2      ;GET LOC(Q).
    129 003340  004777  0000006                  CALL    @COMPAR         ;LOSER(T):Q.
    130 003344  002016                          BGE     SR7             ;BR IF LOSER(T)>=Q.
    131                                 ;  RECORD(Q) BECOMES NEW LOSER AT THIS LEVEL.
    132 003346  016701  175742          1$:     MOV     RQ,R1
    133 003352  016467  000620' 175734          MOV     RN(R4),RQ       ;RN(T)-->RQ.
    134 003360  010164  000620'                  MOV     R1,RN(R4)       ;RQ-->RN(T).
    135 003364  016703  175722                  MOV     Q,R3
    136 003370  016467  000240' 175714          MOV     LOSER(R4),Q     ;LOSER(T)-->Q.
    137 003376  010364  000240'                  MOV     R3,LOSER(R4)    ;Q-->LOSER(T).
    138                                 ;  MOVE UP TREE.  (T IS ADVANCED TO NEXT HIGHER NODE)
    139 003402  016404  000500'          SR7:    MOV     FI(R4),R4       ;FI(T)-->T.
    140 003406  000731                          BR      SR6
```

```
                                        ;
                                        ;  END OF MERGE.
                                        ;
003330  105767  175706          SRFIN:  TSTB    FINFLG          ;WAS THIS THE LAST MERGE?
003334  001020                          BNE     FINCLS          ;BRANCH IF IT WAS.
                                        ;  CLOSE OUT TEMP FILE.
003336  016701  000000G                 MOV     OBUF1,R1        ;SET END OF RUN FLAG.
003342  016702  000000G                 MOV     ORPOS,R2        ;GET CURRENT BUFFER INSERT POINTER
003346  160102                          SUB     R1,R2           ;CALCULATE # OF BYTES IN BUFFER
003350  010261  000002                  MOV     R2,2(R1)        ;STORE BYTE COUNT INTO BUFFER HEADER
003354  052711  100000                  BIS     #<^O100000>,(R1)
003360  016767  175654  000000G         MOV     OFLINK,FLBLK    ;CHAIN TO PREVIOUS RUNS IN FILE
003366  004767  000000G                 CALL    FLUSH           ;WRITE OUT LAST BLOCK.
003372  000167  176056                  JMP     STRAT           ;DETERMINE FUTURE STRATEGY.
                                        ;
                                        ;  Close out the final output (result) file.
                                        ;
003376  016704  000000G         FINCLS: MOV     ORPOS,R4        ;GET CURRENT OUTPUT BUFFER POINTER
003402  016705  000000G                 MOV     OBFEND,R5       ;ADDRESS OF END OF OUTPUT BUFFER
                                        ;
                                        ;  See if we need to insert line-feed after last record.
                                        ;
003406  016700  000000G                 MOV     OTYPE,R0        ;GET OUTPUT FILE TYPE FLAGS
003412  032700  000000C                 BIT     #RT$CBL+RT$FTN,R0 ;COBOL OR FORTRAN FILE?
003416  001412                          BEQ     2$              ;BR IF NOT
003420  032700  000000G                 BIT     #RT$RA,R0       ;RECORDING MODE ASCII?
003424  001407                          BEQ     2$              ;BR IF NOT
003426  032700  000000G                 BIT     #RT$OS,R0       ;ORGANIZATION SEQUENTIAL?
003432  001404                          BEQ     2$              ;BR IF NOT
003434  112703  000012                  MOVB    #LF,R3          ;INSERT LF
003440  004767  002502                  CALL    OUTCHR
                                        ;
                                        ;  See if we should insert control-z at end of file.
                                        ;
003444  016700  000000G         2$:     MOV     OTYPE,R0        ;GET FLAGS
003450  032700  000000G                 BIT     #RT$DBL,R0      ;DIBOL FILE?
003454  001413                          BEQ     FILLST          ;BR IF NOT
003456  032700  000000G                 BIT     #RT$OS,R0       ;ORGANIZATION SEQUENTIAL?
003462  001004                          BNE     3$              ;BR IF YES
003464  032767  000000G 000000G         BIT     #F$INPL,SFLAGS  ;INPLACE RANDOM FILE?
003472  001004                          BNE     FILLST          ;NO CONTROL-Z IF YES
003474  112703  000032          3$:     MOVB    #CTRLZ,R3       ;INSERT CONTROL-Z
003500  004767  002442                  CALL    OUTCHR
```

```
                                ;
                                ;  Determine how to fill out end of current buffer.
                                ;
003504  016703  000000G        FILL$1: MOV     OTYPE,R3        ;GET OUTPUT FILE TYPE FLAGS
003510  032703  000000G                BIT     #RT$OS,R3       ;ORGANIZATION = SEQUENTIAL?
003514  001560                        BEQ     2$              ;BR IF NOT
                                ;
                                ;  Output organization is sequential.
                                ;
003516  004767  002016          4$:     CALL    OUTBUF          ;WRITE CURRENT BUFFER TO FILE
                                ;  If this is an inplace sort, fill rest of file with nulls.
                                ;
003522  032767  000000G 000000G        BIT     #F$INPL,SFLAGS  ;INPLACE SORT?
003530  001002                        BNE     1$              ;BRANCH AROUND IF INPLACE
003532  000167  001110                JMP     CLSOUT          ;JUMP IF NOT INPLACE SORT
                                ;  Write nulls to remainder of file.
003536  005024                  1$:     CLR     (R4)+           ;FILL BUFFER WITH NULLS
003540  020405                        CMP     R4,R5           ;FINISHED?
003542  103775                        BLO     1$              ;BR IF NOT
003544                          3$:     $WRITE  #FD3,OBUF1,OWCNT,NXOBLK
003724                                  $WAIT   #FD3
004040  103002                        BCC     31$
004042  000167  000600                JMP     CLSOUT          ;JUMP IF END OF FILE REACHED
004046                          31$:
004046  066767  000000G 175162        ADD     BLKFC,NXOBLK    ;ADVANCE BLOCK #
004054  000633                        BR      3$              ;CONTINUE WRITING NULLS
                                ;
                                ;  Output organization is Relative.
                                ;
004056  016703  000000G          2$:     MOV     OTYPE,R3        ;GET OUTPUT FILE TYPE FLAGS
004062  032703  000000G                  BIT     #RT$DBL,R3      ;DIBOL FILE?
004066  001033                          BNE     10$             ;BR IF YES
004070  032703  000000G                  BIT     #RT$RA,R3       ;RECORDING MODE = ASCII?
004074  001210                          BNE     4$              ;IF YES THEN FILL REST OF FILE WITH NULLS
004076  032703  000000G                  BIT     #RT$CBL,R3      ;COBOL BINARY RELATIVE FILE?
004102  001605                          BEQ     4$              ;BR IF NOT
                                ;
                                ;  Fill COBOL relative organization binary file with deleted records.
                                ;
004104  016703  000000G          5$:     MOV     RECLEN,R3       ;GET RECORD LENGTH
004110  020405                          CMP     R4,R5           ;IS BUFFER FULL?
004112  103405                          BLO     6$              ;BR IF NOT
004114  004767  001440                  CALL    RLOUT           ;DUMP BUFFER
004120  103002                          BCC     6$
004122  000167  000520                  JMP     CLSOUT          ;JUMP IF END OF FILE REACHED
004126  010324                  6$:     MOV     R3,(R4)+        ;STORE DELETED-RECORD RECORD CONTROL WORD
004130  020405                  7$:     CMP     R4,R5           ;IS BUFFER FULL?
004132  103405                          BLO     8$              ;BR IF NOT
004134  004767  001420                  CALL    RLOUT           ;DUMP BUFFER
004140  103002                          BCC     8$
004142  000167  000500                  JMP     CLSOUT          ;JUMP IF END OF FILE HIT
004146  005024                  8$:     CLR     (R4)+           ;FILL RECORD WITH NULLS
004150  005303                          DEC     R3
004152  003366                          BGT     7$
004154  000753                          BR      5$              ;GO DO NEXT RECORD
                                ;
                                ;  This is a DIBOL random access file.
```

```
                                ;  If this is an inplace sort, preserve the contents of the end of the
                                ;  file beyond the sort output.
                                ;
004156  032767  000000G 000000G 10$:    BIT     #F$INPL,SFLAGS  ;INPLACE SORT?
004164  001002                         BNE     14$
004166  000167  000450                 JMP     13$             ;JUMP IF NOT
004172                          14$:
                                ;  Read in block from file and pad with original data.
004172  010403                         MOV     R4,R3           ;GET OFFSET WITHIN CURRENT BUFFER
004174  166703  000000G                SUB     OBUF1,R3
004200  032703  000777                 BIT     #777,R3         ;ARE WE ON A BLOCK BOUNDARY NOW?
004204  001002                         BNE     15$
004206  000167  000430                 JMP     13$             ;JUMP IF YES (DON'T NEED TO FILL THIS BLOCK)
004212                          15$:    $WAIT   #FD3
004326                                  $READ   #FD3,OBUF2,OWCNT,NXOBLK
004506                                  $WAIT   #FD3
004622  010302                         MOV     R3,R2           ;GET POINTER INTO NEW BUFFER CORRESPONDING TO
004624  066702  000000G                ADD     OBUF2,R2        ;OFFSET IN OLD BUFFER
004630  112224                  12$:    MOVB    (R2)+,(R4)+     ;FILL WITH ORIGINAL FILE DATA
004632  005203                         INC     R3
004634  032703  000777                 BIT     #777,R3         ;REACHED BLOCK BOUNDARY YET?
004640  001373                         BNE     12$             ;BR IF NOT
                                ;  Now write out the last buffer load.
004642  004767  000672          13$:    CALL    OUTBUF
                                ;
                                ;  Close output file and terminate sort.
                                ;
004646                          CLSOUT: $WAIT   #FD3            ;WAIT FOR LAST WRITE TO FINISH
004742  000167  000000G                JMP     P2TERM          ;TERMINATE THE SORT
```

```
                                ;*******************************************************************************
                                ;  OUTREC is called to move the record pointed to by R1 to the
                                ;  final output file buffer.
                                ;
004766  010146                  OUTREC: MOV     R1,-(SP)
004770  010246                          MOV     R2,-(SP)
004772  010346                          MOV     R3,-(SP)
004774  010446                          MOV     R4,-(SP)
004776  010546                          MOV     R5,-(SP)
005000  016704  000000G                 MOV     ORPOS,R4        ;GET CURRENT POINTER INTO OUTPUT BUFFER
005004  016705  000000G                 MOV     OBFEND,R5       ;GET POINTER TO END OF BUFFER
005010  016702  000000G                 MOV     RECLEN,R2       ;GET SORT RECORD LENGTH
                                ;
                                ;  Determine record type.
                                ;
005014  016700  000000G                 MOV     OTYPE,R0        ;GET OUTPUT FILE TYPE FLAGS
005020  032700  000000G                 BIT     #RT$F11,R0      ;FILES-11??
005024  001404                          BEQ     3$              ;  BRANCH IF NOT
005026  032700  000000G                 BIT     #RT$OS,R0       ;SEQUENTIAL?
005032  001133                          BNE     ORRS            ;BRANCH TO FILES-11 SEQUENTIAL
005034  000422                          BR      ORFL            ;BRANCH TO FILES-11 RELATIVE
005036  032700  000000G         3$:     BIT     #RT$CBL,R0      ;COBOL FILE?
005042  001404                          BEQ     1$              ;BR IF NOT
005044  032700  000000G                 BIT     #RT$RA,R0       ;COBOL ASCII FILE?
005050  001066                          BNE     ORCA            ;BR IF YES
005052  000564                          BR      ORCB            ;COBOL BINARY RECORD
005054  032700  000000G         1$:     BIT     #RT$FTN,R0      ;FILE TYPE = FORTRAN?
005060  001404                          BEQ     2$              ;BR IF NOT
005062  032700  000000C                 BIT     #RT$OR+RT$RB,R0 ;RELATIVE OR BINARY?
005066  001001                          BNE     2$              ;BR IF EITHER
005070  000456                          BR      ORCA            ;FORTRAN, SEQUENTIAL, ASCII
005072  032767  000000G 000000G 2$:     BIT     #RT$VLN,ITYPE   ;VARIABLE LENGTH INPUT RECORDS?
005100  001016                          BNE     ORVL            ;BR IF YES
                                ;
                                ;  Fixed length record.
                                ;
005102  020405                  ORFL:   CMP     R4,R5           ;IS OUTPUT BUFFER FULL?
005104  103402                          BLO     1$              ;BR IF NOT
005106  004767  000426                 CALL    OUTBUF          ;FLUSH THE BUFFER
005112  112124                  1$:     MOVB    (R1)+,(R4)+     ;MOVE RECORD TO OUTPUT BUFFER
005114  005302                          DEC     R2              ;MOVE ALL OF RECORD
005116  003371                          BGT     ORFL
                                ;  See if we should add CR-LF.
005120  032767  000000G 000000G         BIT     #RT$CR,OTYPE    ;ADD CR-LF?
005126  001574                          BEQ     OREND           ;BR IF NOT
005130  004767  001026                 CALL    ADDCR           ;ADD CR-LF
005134  000571                          BR      OREND
                                ;
                                ;  Variable length record.
                                ;
005136  112103                  ORVL:   MOVB    (R1)+,R3        ;GET NEXT CHAR FROM RECORD
005140  001407                          BEQ     2$              ;BR IF END OF RECORD HIT
005142  020405                          CMP     R4,R5           ;OUTPUT BUFFER FULL?
005144  103402                          BLO     1$              ;BR IF NOT
005146  004767  000366                 CALL    OUTBUF          ;FLUSH BUFFER
005152  110324                  1$:     MOVB    R3,(R4)+        ;MOVE CHAR TO OUTPUT BUFFER
005154  005302                          DEC     R2              ;CONTINUE MOVING RECORD
```

```
     58 005156  003367                              BGT     ORVL
     59                                 ;  If output organization = relative, pad with blanks to full record length.
     60 005160  032767  000000G 000000G 2$:         BIT     #RT$VLN,OTYPE   ;ALLOW SHORT OUTPUT RECORDS?
     61 005166  001010                              BNE     3$              ;BR IF YES
     62 005170  005702                              TST     R2              ;NEED PADDING?
     63 005172  001406                              BEQ     3$              ;BR IF NOT
     64 005174  112703  000040                      MOVB    #' ,R3          ;GET PADDING SPACE
     65 005200  004767  000742           4$:        CALL    OUTCHR          ;STORE BLANKS INTO END OF RECORD
     66 005204  005302                              DEC     R2              ;PAD TO FULL RECORD LENGTH
     67 005206  003374                              BGT     4$
     68                                 ;  Insert carriage-return line-feed.
     69 005210  032767  000000G 000000G 3$:         BIT     #RT$CR,OTYPE    ;NEED TO INSERT CR-LF?
     70 005216  001540                              BEQ     OREND           ;BR IF NOT
     71 005220  004767  000736                      CALL    ADDCR           ;INSERT CR-LF
     72 005224  000535                              BR      OREND
```

```
                                ;
                                ;  COBOL ascii record.  (And FORTRAN sequential, ascii record)
                                ;
005226  112703  000012          ORCA:   MOVB    #LF,R3          ;INSERT LEADING LINE-FEED
005232  004767  000710                  CALL    OUTCHR
005236  112103                  1$:     MOVB    (R1)+,R3        ;GET NEXT CHAR FROM RECORD
005240  001407                          BEQ     3$              ;BR IF END OF RECORD REACHED
005242  020405                          CMP     R4,R5           ;OUTPUT BUFFER FULL?
005244  103402                          BLO     2$              ;BR IF NOT
005246  004767  000266                  CALL    OUTBUF          ;FLUSH BUFFER
005252  110324                  2$:     MOVB    R3,(R4)+        ;MOVE CHAR TO OUTPUT BUFFER
005254  005302                          DEC     R2              ;MORE TO MOVE?
005256  003367                          BGT     1$              ;BR IF YES
                                ;  If file organization is Relative, pad with blanks to full record length.
005260  032767  000000G 000000G 3$:     BIT     #RT$VLN,OTYPE   ;ALLOW SHORT OUTPUT RECORDS?
005266  001010                          BNE     4$              ;BR IF YES
005270  005702                          TST     R2              ;DO WE NEED PADDING?
005272  001406                          BEQ     4$              ;BR IF NOT
005274  112703  000040                  MOVB    #' ,R3          ;GET PADDING BLANK
005300  004767  000642          5$:     CALL    OUTCHR          ;INSERT BLANKS
005304  005302                          DEC     R2              ;FILL RECORD FILLED
005306  003374                          BGT     5$
005310  112703  000015          4$:     MOVB    #CR,R3          ;NOW INSERT CR
005314  004767  000626                  CALL    OUTCHR
005320  000477                          BR      OREND
```

```
                                ;
                                ;  FILES-11 sequential access
                                ;
005322  005204                 ORRS:   INC     R4              ;ROUND BUFFER POINTER TO NEXT
005324  042704  000001                 BIC     #1,R4           ;WORD
                                ;  Strip trailing nulls or blanks.
005330  010103                          MOV     R1,R3           ;GET ADDRESS OF RECORD
005332  060203                          ADD     R2,R3           ;POINT PAST LAST WORD OF RECORD
005334  032767  000000G 000000G        BIT     #RT$RB,OTYPE    ;BINARY MODE?
005342  001405                          BEQ     7$              ;BRANCH IF NOT
005344  105743                 2$:     TSTB    -(R3)           ;SCAN OFF TRAILING NULL WORDS
005346  001010                          BNE     1$              ;BR IF REACHED NON-NULL
005350  005302                          DEC     R2              ;KEEP TRACK OF RECORD LENGTH
005352  003374                          BGT     2$              ;BR IF MORE TO CHECK
005354  000405                          BR      1$
                                ;  ASCII recording mode
005356  122743  000040           7$:     CMPB    #' ,-(R3)
005362  001002                          BNE     1$
005364  005302                          DEC     R2
005366  003373                          BGT     7$
                                ;  Record size word.
005370  020405                 1$:     CMP     R4,R5           ;IS OUTPUT BUFFER FULL?
005372  103402                          BLO     3$              ;BR IF NOT
005374  004767  000140                 CALL    OUTBUF          ;FLUSH BUFFER
005400  010224                 3$:     MOV     R2,(R4)+                ;STORE RECORD LENGTH
005402  001407                          BEQ     6$              ;BR IF NONE
005404  020405                 5$:     CMP     R4,R5           ;OUTPUT BUFFER FULL?
005406  103402                          BLO     4$              ;BR IF NOT
005410  004767  000124                 CALL    OUTBUF          ;FLUSH BUFFER
005414  112124                 4$:     MOVB    (R1)+,(R4)+     ;MOVE RECORD TO OUTPUT BUFFER
005416  005302                          DEC     R2
005420  003371                          BGT     5$
005422  000436                 6$:     BR      ORFND
                                ;
                                ;  COBOL Binary record.
                                ;
005424  042702  000001           ORCB:   BIC     #1,R2           ;MAKE SURE RECORD LENGTH IS EVEN
005430  032767  000000G 000000G        BIT     #RT$OR,OTYPE    ;ORGANIZATION = RELATIVE?
005436  001007                          BNE     1$              ;BR IF YES
                                ;  Strip trailing nulls.
005440  010103                          MOV     R1,R3           ;GET ADDRESS OF RECORD
005442  060203                          ADD     R2,R3           ;POINT PAST LAST WORD OF RECORD
005444  005743                 2$:     TST     -(R3)           ;SCAN OFF TRAILING NULL WORDS
005446  001003                          BNE     1$              ;BR IF REACHED NON-NULL
005450  162702  000002                 SUB     #2,R2           ;KEEP TRACK OF RECORD LENGTH
005454  003373                          BGT     2$              ;BR IF MORE TO CHECK
                                ;  Build record control word.
005456  020405                 1$:     CMP     R4,R5           ;IS OUTPUT BUFFER FULL?
005460  103402                          BLO     3$              ;BR IF NOT
005462  004767  000052                 CALL    OUTBUF          ;FLUSH BUFFER
005466  010214                 3$:     MOV     R2,(R4)         ;STORE RECORD LENGTH
005470  052724  100000                 BIS     #100000,(R4)+   ;SET RECORD-EXISTS FLAG
005474  006202                          ASR     R2              ;GET # WORDS IN RECORD
005476  001407                          BEQ     6$              ;BR IF NONE
005500  020405                 5$:     CMP     R4,R5           ;OUTPUT BUFFER FULL?
005502  103402                          BLO     4$              ;BR IF NOT
005504  004767  000030                 CALL    OUTBUF          ;FLUSH BUFFER
```

```
      58 005510  012124                    4$:     MOV     (R1)+,(R4)+     ;MOVE RECORD TO OUTPUT BUFFER
      59 005512  005302                            DEC     R2
      60 005514  003371                            BGT     5$
      61 005516  000400                    6$:     BR      ORFND
      62                                    ;
      63                                    ;  Finished moving record to output buffer.
      64                                    ;
      65 005520  010467  0000008           ORFND:  MOV     R4,ORPOS        ;SAVE OUTPUT BUFFER POINTER
      66 005524  012605                            MOV     (SP)+,R5
      67 005526  012604                            MOV     (SP)+,R4
      68 005530  012603                            MOV     (SP)+,R3
      69 005532  012602                            MOV     (SP)+,R2
      70 005534  012601                            MOV     (SP)+,R1
      71 005536  000207                            RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  OUTBUF is called to write the current output buffer to the final output file.
                                ;  An abort occurs if the output file overflows.
                                ;  Inputs:
                                ;    R4 = Pointer to end of data in buffer.
                                ;    OBUF1 = Pointer to start of buffer.
                                ;  Outputs:
                                ;    R4 = Address of start of fresh buffer.
                                ;    R5 = Address of end of buffer.
                                ;
005540  004767  000014          OUTBUF: CALL    RLOUT           ;TRY TO WRITE BUFFER TO FILE
005544  103004                          BCC     1$              ;BR IF NO OVERFLOW
005546                                  XXX     16              ;FILE OVERFLOW
005556  000207                  1$:     RETURN
                                ;-------------------------------------------------------------------------------
                                ;  RLOUT is called to write the current buffer full of data up to
                                ;  (but not beyond) the end of the current output file.
                                ;  The c-flag is set on return if the end of file is reached.
                                ;  Inputs:
                                ;    R4 = Address of end of data in current output buffer.
                                ;    OBUF1 = Address of current output buffer.
                                ;  Outputs:
                                ;    R4 = Address of start of fresh buffer.
                                ;    R5 = Address of end of buffer.
                                ;
005560                          RLOUT:  $WAIT   #FD3            ;WAIT FOR LAST WRITE TO FINISH
005674  103523                          BCS     9$              ;BR IF ERROR
005676  032704  000001          1$:     BIT     #1,R4           ;IS R4 ON EVEN BYTE BOUNDARY?
005702  001401                          BEQ     2$              ;BR IF YES
005704  105024                          CLRB    (R4)+           ;PUT IN NULL
005706  166704  000000G         2$:     SUB     OBUF1,R4        ;GET # BYTES IN BUFFER
005712  001477                          BEQ     3$              ;BR IF BUFFER EMPTY
005714  006204                          ASR     R4              ;GET # WORDS IN BUFFER
005716                                  $WRITE  #FD3,OBUF1,R4,NXOBLK
006072  103424                          BCS     9$              ;BR IF HIT END OF FILE
006074  062704  000377                  ADD     #377,R4         ;BOUND # WORDS UP TO BLOCK BOUNDARY
006100  042704  000377                  BIC     #377,R4
006104  000304                          SWAB    R4              ;GET # BLOCKS WRITTEN
006106  060467  173124                  ADD     R4,NXOBLK       ;ADVANCE FILE BLOCK COUNTER
                                ;  Set up new buffer pointers.
006112  016704  000000G         3$:     MOV     OBUF2,R4        ;GET ADDRESS OF NEW BUFFER
006116  016767  000000G 000000G        MOV     OBUF1,OBUF2     ;SWITCH BUFFERS
006124  010467  000000G                 MOV     R4,OBUF1
006130  010405                          MOV     R4,R5
006132  066705  000000G                 ADD     OBUFSZ,R5       ;GET ADDRESS OF END OF BUFFERR
006136  010567  000000G                 MOV     R5,OBFEND
006142  000241                          CLC                     ;SAY NOT AT END OF FILE
006144  000207                  9$:     RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  OUTCHR is called to place the character in R3 in the output buffer.
                                ;  Inputs:
                                ;    R4 = Pointer into output buffer.
                                ;    R5 = Address of end of output buffer.
                                ;
006146  020405                  OUTCHR: CMP     R4,R5           ;IS OUTPUT BUFFER FULL?
006150  103402                          BLO     1$              ;BR IF NOT
006152  004767  177362                  CALL    OUTBUF          ;FLUSH THE BUFFER
006156  110324                  1$:     MOVB    R3,(R4)+        ;MOVE CHAR TO OUTPUT BUFFER
006160  000207                          RETURN

                                ;-------------------------------------------------------------------------------
                                ;  ADDCR is called to insert Carriage-return & Line-feed into the
                                ;  current output buffer.
                                ;  Inputs:
                                ;    R4 = Pointer into output buffer.
                                ;    R5 = Address of end of output buffer.
                                ;
006162  010346                  ADDCR:  MOV     R3,-(SP)
006164  012703  000015                  MOV     #CR,R3          ;INSERT CR
006170  004767  177752                  CALL    OUTCHR
006174  012703  000012                  MOV     #LF,R3          ;INSERT LF
006200  004767  177742                  CALL    OUTCHR
006204  012603                          MOV     (SP)+,R3
006206  000207                          RETURN
```

```
                                .SBTTL  SUBROUTINES
                        ;
                        ;  ROUTINE TO FREE THE DISK BLOCK WHOSE ADDRESS IS IN R4.
                        ;  R4 IS NOT ALTERED BY THE SUBROUTINE.
                        ;
006210  010146          RELGRN: MOV     R1,-(SP)        ;SAVE SOME REGISTERS.
006212  010246                  MOV     R2,-(SP)
006214  010401                  MOV     R4,R1           ;GET DISK GRANULE #.
006216  042701  177760          BIC     #<^O177760>,R1  ;GET # OF BIT IN WORD.
006222  012702  100000          MOV     #<^O100000>,R2
006226  000402                  BR      1$
006230  000241          2$:     CLC
006232  006002                  ROR     R2              ;POSITION BIT TO PROPER POSITION.
006234  005301          1$:     DEC     R1
006236  002374                  BGE     2$
006240  010401                  MOV     R4,R1
006242  006201                  ASR     R1              ;GET INDEX TO WORD.
006244  006201                  ASR     R1              ;(DIVIDE BY 16)
006246  006201                  ASR     R1
006250  006201                  ASR     R1
006252  006301                  ASL     R1              ;CVT TO BYTE INDEX
006254  066701  172720          ADD     GRNTBL,R1       ;GET ADDRESS OF WORD.
006260  020167  172716          CMP     R1,GRNEND       ;DON'T GO PAST END OF TABLE
006264  002052                  BGE     RELOVF
006266  030211                  BIT     R2,(R1)         ;GRANULE SHOULD BE BUSY NOW.
006270  001050                  BNE     RELOVF
006272  050211                  BIS     R2,(R1)         ;MARK GRANULE AS FREE.
006274  012602                  MOV     (SP)+,R2        ;RESTORE REGISTERS.
006276  012601                  MOV     (SP)+,R1
006300  000207                  RETURN
                        ;
                        ;  ROUTINE TO GET THE ADDRESS OF A FREE DISK GRANULE.
                        ;  THE ADDRESS IS RETURNED IN R1.
                        ;
006302  010246          GETGRN: MOV     R2,-(SP)        ;SAVE SOME REGISTERS.
006304  010346                  MOV     R3,-(SP)
006306  010446                  MOV     R4,-(SP)
006310  016701  172664          MOV     GRNTBL,R1       ;GET ADDRESS OF START OF BIT TABLE.
006314  005721          1$:     TST     (R1)+           ;LOOK FOR WORD WITH FREE GRANULE.
006316  001776                  BEQ     1$              ;BRANCH IF NONE IN THIS WORD.
006320  014102                  MOV     -(R1),R2                ;GET FREE GRANULE BITS.
006322  005003                  CLR     R3
006324  006102          3$:     ROL     R2              ;SEARCH FOR FREE GRAN BIT.
006326  103402                  BCS     2$              ;BRANCH WHEN FOUND.
006330  005203                  INC     R3              ;KEEP TRACK OF ADDRESS.
006332  000774                  BR      3$
006334  010304          2$:     MOV     R3,R4           ;SAVE BIT INDEX TO GRANULE.
006336  000241          4$:     CLC                     ;MARK GRANULE AS USED NOW.
006340  006002                  ROR     R2              ;REPOSITION BIT.
006342  005303                  DEC     R3
006344  002374                  BGE     4$
006346  010211                  MOV     R2,(R1)         ;RESTORE WORD WITH GRANULE BUSY.
006350  166701  172624          SUB     GRNTBL,R1       ;GET WORD INDEX.
006354  006301                  ASL     R1              ;CONVERT WORD TO BIT INDEX.
006356  006301                  ASL     R1
006360  006301                  ASL     R1
006362  060401                  ADD     R4,R1           ;ADD IN BIT INDEX WITHIN WORD.
```

SUBROUTINES

```
     58 006364  020167  0000006                  CMP     R1,NXTBLK       ;MAKE SURE WE DON'T OVERFLOW.
     59 006370  003004                           BGT     OVTMP           ;TEMP FILE OVERFLOW
     60 006372  012604                           MOV     (SP)+,R4
     61 006374  012603                           MOV     (SP)+,R3
     62 006376  012602                           MOV     (SP)+,R2
     63 006400  000207                           RETURN
     64 006402                          OVTMP:  XXX     17
     65 006412                          RELOVF: XXX     18
     66                                 ;
     67          006424'                P2TOP   =       .+2
     68          006424                 P2SIZ   =       P2TOP-P2BASE
     69          000001                         .END
```

Symbol table

```
ABORT  = ****** G        FILST  = ****** G        MRGSET  002102R     002 OWCNT  = ****** G        RT$VLN= ****** G
ADDCR  006162R     002   FINCLS  003376R     002 MXFILS= ****** G        PENMRG  001546R     002 RT11   = 000000
BELL   = 000007          FINFLG  001242R     002 MXFIL2= ****** G        PFSPC  = 000002          RWDLEN= ****** G
BIGEST= 177777           FINMOV  002472R     002 MXMRG  = 000050        P2BASE= 000000R     002 SEVERR= 000010
BLKFC  = ****** G        FLBLK  = ****** G        MXMRG2= 000120        P2SIZ  = 006424 G        SFLAGS= ****** G
BUFSIZ= 001000           FLUSH  = ****** G        NFILE2  001204R     002 P2TERM= ****** G        SORTP2  001244RG    002
CHNBIT= 000400           FT$T   = ****** G        NOERR  = 000001        P2TOP  = 006424RG    002 SRFIN  003330R     002
CHNO   = 000000          F$INPL= ****** G         NUMKY2= ****** G        Q       001232R     002 SR2    002270R     002
CISRT2= ****** G         GETGRN  006302R     002 NUMMRG  001206R     002 QLEN   = 000003          SR3    002330R     002
CLSOUT  004646R     002  GRNEND  001202R     002 NXOBLK  001236R     002 RC      001230R     002 SR4    002506R     002
COMPAR= ****** G         GRNTBL  001200R          NXTBLK= ****** G        RECLEN= ****** G         SR4L   003174R     002
CR     = 000015          HB$ARS= 000006           OBFEND= ****** G        RELGRN  006210R     002 SR5    003202R     002
CSPCOF= 000376           HB$DFS= 000002           OBUFSZ= ****** G        RELOVF  006412R     002 SR6    003212R     002
CTRLZ  = 000032          HB$DIP= 000030           OBUF1  = ****** G        RLOUT   005560R     002 SR7    003322R     002
CUROFL= ****** G         HB$DUP= 000001           OBUF2  = ****** G        RMON   = 000054          STRAT  001454R     002
DBLKFC= 000005           HB$RIU= 000032           OFLINK  001240R     002 RN      000620R     002 STRFIN  001652R     002
DOMRG   001656R     002  HB$RSZ= 000004           ORCA    005226R     002 RQ      001234R     002 TMPMOV  002350R     002
DSKCVT= ****** G         HD$DEV= 000000           ORCB    005424R     002 RSTSJB= 000404           TOPMEM= ****** G
EMTBLK  001210R     002  HD$$SZ= 000004           OREND   005520R     002 RT$CBL= ****** G         UEOF   = ****** G
ENDRUN  003144R     002  IRPOS   000740R     002 ORFL    005102R     002 RT$CR  = ****** G         USERRB= 000053
EOFBUF= ****** G         ITYPE  = ****** G        ORPOS  = ****** G        RT$DBL= ****** G         USRLOC= 000046
ERR    = 000006          JSW    = 000044          ORRS    005322R     002 RT$FTN= ****** G         VERSNO= 000276
ERRLOC= 000052           KSTRT  = ****** G        ORVL    005136R     002 RT$F11= ****** G         WRDS   = 000004
FD3    = ****** G        KTYPE  = ****** G        OTYPE  = ****** G        RT$OR  = ****** G         XEMT   = ****** G
FE      000360R     002  LF     = 000012          OUTBUF  005540R     002 RT$OS  = ****** G         XXX16  = 005546R    002
FI      000500R     002  LOC     000120R     002 OUTCHR  006146R     002 RT$OX  = ****** G         XXX17  = 006402R    002
FILFL  = ****** G        LOSER   000240R     002 OUTREC  004766R     002 RT$RA  = ****** G         XXX18  = 006412R    002
FILLST  003504R     002  MAJMRG  001576R     002 OVLBIT= 001000           RT$RB  = ****** G         ...V1  = 000003
FILNB  = ****** G        MRGBA   001060R     002 OVTMP   006402R     002 RT$TXT= ****** G         ...V2  = 000061
FILRN  = ****** G        MRGLST  000000R     002
```

```
. ABS.   000000    000   (RW,I,GBL,ABS,OVR)
         000000    001   (RW,I,LCL,REL,CON)
SORT2    006422    002   (RW,I,GBL,REL,OVR)
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  15783 Words  ( 62 Pages)
Size of core pool:  18432 Words  ( 72 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:26.36
,DB4:CBSRT2,DK:CBSRTH.CRF=DK:CBSRTH,CBSRT2/C
```

Cross reference table (CREF V05.03)

```
. . . V1    6-62    6-82    6-83    6-83    8-20    8-20    8-21    8-21    8-71    8-71    8-72    8-72    8-73    8-73
            8-85    8-85   12-27   12-27   12-35   12-35
. . . V2    6-62    6-82    6-82#   6-82#   6-83    6-83#   8-20    8-20    8-20#   8-20#   8-21    8-21#   8-71    8-71#
            8-72    8-72    8-72#   8-72#   8-73    8-73#   8-85    8-85#   12-27   12-27#  12-35   12-35   12-35#  12-35#
ABORT       2-38   12-13   14-64   14-65
ADDCR       2-46    7-71   13-20#
BELL        1-40#
BIGEST      1-36#
BLKFC       2-46    8-25
BUFSIZ      1-34#
CHNBIT      1-53#
CHNO        1-87#   6-82    6-82    6-83    6-83    8-20    8-20    8-21    8-21    8-71    8-71    8-72    8-72    8-73
            8-73    8-85    8-85   12-27   12-27   12-35   12-35
CISRT2      2-38    3-5
CLSOUT      2-15    8-23    8-45    8-51    8-85#
COMPAR      2-43    6-129
CR          1-38#   2-70#   10-23   13-21
CSPCOF      1-55#
CTRLZ       1-41#   2-72#    7-42
CUROFL      2-45    4-29*   5-21*    5-42    5-105
DBLKFC      1-35#
DOMRG       5-7#
DSKCVT      2-44    6-81
EMTBLK      2-77#
ENDRUN      6-75    6-105#
EOFBUF      2-39
ERR         1-90#   6-82*   6-82*   6-82*   6-83    6-83#   6-83*   6-83*   8-20*   8-20*   8-20*   8-21    8-21*   8-21*
            8-21*   8-71    8-71*   8-71*   8-71*   8-72*   8-72*   8-72*   8-73    8-73*   8-73*   8-73*   8-85    8-85*
            8-85*   8-85*   12-27   12-27*  12-27*  12-27*  12-35*  12-35*  12-35*
ERRLOC      1-49#   6-82    6-83    8-20    8-21    8-71    8-72    8-73    8-85   12-27   12-35
F$INPL      2-51    7-40    8-13    8-61
FD3         2-38    8-20    8-21    8-71    8-72    8-73    8-85   12-27   12-35
FE          2-63#   5-80*   6-116
FI          2-64#   5-81*   6-139
FILFL       2-44    5-109*
FILLST      7-37    7-41    8-4#
FILNB       2-41    5-40    5-52    6-80*
FILRN       2-40    4-11    4-45    5-13    5-16    5-22*   5-35    5-51    6-109*
FILST       2-40    5-92    5-106   5-108*  6-108*
FINCLS      7-5     7-18#
FINFLG      2-84#   3-54*   4-27*    5-7    5-24    5-103   6-18    7-4
FINMOV      6-19    6-54#
FLBLK       2-44    6-39*   7-12*
FLUSH       2-43    6-40    7-13
FT$T        2-45
GETGRN      5-107   6-38   14-35#
GRNEND      2-74#   3-17*  14-23
GRNTBL      2-73#   3-10*  14-22    14-38   14-53
HB$ARS      1-74#
HB$DFS      1-72#
HB$DIP      1-75#
HB$DUP      1-71#
HB$RIU      1-76#
HB$RSZ      1-73#
HD$$SZ      1-81#
HD$DEV      1-80#
```

Cross reference table (CREF V05.03)

```
IRPOS    2-45#   5-91*   6-62    6-100*
ITYPE    2-47    6-47    6-91    9-32
JSW      1-51#
KSTRT    2-35
KTYPE    2-39
LF       1-39#   2-71#   7-30    10-4    13-23
LOC      2-61#   3-39*   6-25    6-55    6-61    6-126   6-128
LOSER    2-62#   5-83*   6-125   6-136   6-137*
MAJMRG   4-22    4-36    4-44#
MRGBA    2-67#   3-41*   5-90    6-63    6-73
MRGLST   2-60#   5-47    5-57*   5-89    6-79    6-106
MRGSET   5-68#
MXFIL2   2-40    4-13    4-47    5-19    5-55
MXFILS   2-40    4-29    4-53
MXMRG    2-57#   2-58    2-60    2-61    2-62    2-63    2-64    2-65    2-66    2-67
MXMRG2   2-58#   3-46
NFILE2   2-75#   3-49*   4-24    4-32    4-39    4-56    4-58
NOERR    1-56#
NUMKY2   2-42
NUMMRG   2-76#   4-28*   4-41*   4-59*   4-60*   5-59    5-68    5-95
NXOBLK   2-81#   3-53*   8-20    8-25*   8-72    12-35   12-40*
NXTBLK   2-43    3-11    3-23    14-58
OBFEND   2-41    3-32*   6-27    6-43    7-19    9-11    12-47*
OBUF1    2-42    3-30*   5-23    6-35    6-41    7-7     8-20    8-67    12-32   12-35   12-43   12-44*
OBUF2    2-42    3-33*   8-72    8-75    12-42   12-43*
OBUFSZ   2-41    2-42    3-29    3-42    12-46
OFLINK   2-82#   5-106*  7-12
ORCA     9-25    9-31    10-4#
ORCB     9-26    11-37#
OREND    9-45    9-47    9-70    9-72    10-25   11-33   11-61   11-65#
ORFL     9-21    9-37#   9-42
ORPOS    2-43    5-28*   6-26    6-49*   7-8     7-18    9-10    11-65*
ORRS     9-20    11-4#
ORVL     9-33    9-51#   9-58
OTYPE    2-51    7-23    7-35    8-4     8-30    9-16    9-44    9-60    9-69    10-15   11-9    11-38
OUTBUF   8-10    8-81    9-39    9-55    10-10   11-24   11-29   11-50   11-57   12-11#  13-9
OUTCHR   7-31    7-43    9-65    10-5    10-20   10-24   13-7#   13-22   13-24
OUTREC   6-56    9-5#
OVLBIT   1-52#
OVTMP    14-59   14-64#
OWCNT    2-46    6-82    6-82    8-20    8-20    8-72    8-72
P2BASE   2-4#    14-68
P2SIZ    2-34    14-68#
P2TERM   2-45    8-86
P2TOP    2-34    3-6     14-67#  14-68
PENMRG   4-23    4-25    4-32#
PFSPC    1-59#
Q        2-79#   3-52*   5-98*   6-24    6-54    6-60    6-72    6-99    6-115   6-127   6-135   6-136*
QLEN     1-48#
RC       2-78#   3-51*   5-97*   6-5     6-13*
RECLEN   2-39    6-29    6-66    8-40    9-12
RELGRN   3-24    6-78    14-6#
RELOVF   14-24   14-26   14-65#
RLOUT    8-43    8-49    12-11   12-27#
RMON     1-57*
RN       2-45#   5-82*   6-122   6-133   6-134*
```

Cross reference table (CREF V05.03)

```
RQ          2-60#   3-50*   5-99*   6-6     6-8     6-13    6-14    6-105*  6-113*  6-120   6-132   6-133*
RSTSJB      1-44#
RT$CBL      2-43    7-24    8-35    9-22
RT$CR       2-47    9-44    9-69
RT$DBL      2-43    7-36    8-31
RT$F11      2-47    9-17
RT$FTN      2-48    7-24    9-27
RT$OR       2-47    9-29    11-38
RT$OS       2-49    7-28    7-38    8-5     9-19
RT$OX       2-47
RT$RA       2-50    7-26    8-33    9-24
RT$RB       2-50    9-29    11-9
RT$TXT      2-48
RT$VLN      2-47    6-47    6-91    9-32    9-60    10-15
RT11        1-47#
RWDLEN      2-38    3-40
SEVERR      1-55#
SFLAGS      2-52    7-40    8-13    8-61
SORTP2      2-34    3-5#
SR2         5-104   6-5#    6-119
SR3         6-7     6-14#
SR4         6-16    6-50    6-60#
SR4L        6-101   6-113#
SR5         6-110   6-115#
SR6         6-117#  6-140
SR7         6-121   6-124   6-130   6-139#
SRFIN       6-12    7-4#
STRAT       4-9#    7-14
STRFIN      4-30    4-42    4-60#
TMPMOV      6-24#
TOPMEM      2-39    3-43
UEOF        2-62
USERRB      1-50#
USRLOC      1-54#
VERSNO      1-53#
WRDS        1-89#   6-82*   6-82*   6-83    8-20*   8-20*   8-21    8-71    8-72*   8-72*   8-73    8-85    12-27   12-35*
            12-35*
XEMT        6-82    6-82    6-82    8-20    8-20    8-20    8-72    8-72    8-72    12-35   12-35   12-35
XXX16       12-13#
XXX17       14-64#
XXX18       14-65#
```

Cross reference table (CREF V05.03)

```
$ERASE    1-653#
$EXIT     1-686#
$FDB      1-145#
$GETBO    1-402#
$GETCM    1-356#
$GETTO    1-424#
$KEEP     1-445#
$LOCK     1-329#
$NEW      1-228#
$OLD      1-260#
$PRINT    1-664#
$PRT      1-696#
$PURGE    1-465#
$READ     1-492#   6-82     8-72
$REOPE    1-309#
$SAVES    1-299#
$SPC      1-159#
$SYDON    1-674#
$SYINI    1-589#
$SYSTE    1-171#
$UNLOC    1-336#
$WAIT     1-558#   6-83     8-21     8-71     8-73     8-85     12-27
$WRITE    1-525#   8-20    12-35
...CM1    6-52     8-20     8-72    12-35
...CM2    6-52     6-82     6-82     6-82     8-20     8-20     8-20     8-20     8-72     8-72     8-72     8-72    12-35    12-35
         12-35    12-35
...CM5    6-82     8-20     8-72    12-35
...CM7    6-82     8-20     8-72    12-35
.CHAIN    1-95#    1-99#
.CLOSE    1-58#
.CSISP    1-100#
.DELET    1-97#
.DSTAT    1-98#
.ENTER    1-96#
.EXIT     1-99#
.FETCH    1-98#
.GTLIN    1-101#
.GVAL     1-100#
.HERR     1-97#
.LOCK     1-96#
.LOOKU    1-96#
.PRINT    1-97#
.PURGE    1-99#
.QSET     1-58#
.READ     1-101#   6-82     8-72
.READW    1-100#
.RELEA    1-99#
.REOPE    1-100#
.SAVES    1-100#
.SERR     1-97#
.SETTO    1-97#
.TTYIN    1-99#
.TTYOU    1-98#
.UNLOC    1-96#
.WAIT     1-101#   6-83     8-21     8-71     8-73     8-85     12-27
.WRITE    1-101#   8-20    12-35
```

Cross reference table (CREF V05.03)

```
.WRITW   1-1018
POP      1-1188#  6-82    6-82    6-83    8-20    8-20    8-21    8-71    8-72    8-72    8-73    8-85    12-27   12-35
         13-35
PUSH     1-1128#  6-82    6-82    6-83    8-20    8-20    8-21    8-71    8-72    8-72    8-73    8-85    12-27   12-35
         13-35
SETNRT   1-2108#
XXX      2-2268#  12-13   14-64   14-65
```