96769260 A

CONTROL DATA CORPORATION

SORT/MERGE VERSION 1.0 REFERENCE MANUAL

CONTROL DATA®

MASS STORAGE OPERATING SYSTEM

REVISION RECORD

REVISION	DESCRIPTION
01	Preliminary version of manual released
(8/76)	
A	Manual released
(9/76)	
Publication No. 96769260	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4455 Eastgate Mall
La Jolla, CA 92037

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision	SFC†
Cover	–	
Title Page	–	
ii	A	
iii	A	
v	A	
vii	A	
1-1	A	
1-2	A	
1-3	A	
2-1	A	
3-1	A	
3-2	A	
3-3	A	
3-4	A	
3-5	A	
3-6	A	
3-7	A	
3-8	A	
3-9	A	
3-10	A	
3-11	A	
3-12	A	
3-13	A	
3-14	A	
3-15	A	
3-16	A	
3-17	A	
3-18	A	
4-1	A	
4-2	A	
4-3	A	
4-4	A	
4-5	A	
4-6	A	
4-7	A	
4-8	A	
4-9	A	
4-10	A	
A-1	A	
A-2	A	
A-3	A	
A-4	A	
B-1	A	
B-2	A	
B-3	A	
B-4	A	
C-1	A	
C-2	A	
C-3	A	
C-4	A	
D-1	A	
D-2	A	
D-3	A	
D-4	A	

Page	Revision	SFC†
E-1	A	
F-1	A	
G-1	A	
G-2	A	
H-1	A	
I-1	A	
I-2	A	
I-3	A	
I-4	A	
I-5	A	
I-6	A	
I-7	A	
J-1	A	
J-2	A	
K-1	A	
K-2	A	
L-1	A	
L-2	A	
L-3	A	
Comment Sheet	–	
Envelope	–	
Back Cover	–	

†Software Feature Change

PREFACE

The Sort/Merge system is available under the Mass Storage Operating System (MSOS), Versions 4.3 and 5.0. Sort/Merge processing is specified through the use of control statements.

The reader is assumed to be familiar with the MSOS system on which Sort/Merge is to be run (including the job processing capabilities of that system) and the file manager systems (to the extent that managed files are used).

Documents of interest to Sort/Merge users are:

Publication	Publication Number
MSOS 4 Reference Manual	60361500
MSOS 5 Reference Manual	96769400
MSOS File Manager Reference Manual	39520600
Software Peripheral Drivers Reference Manual	96769390

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or undefined parameters.

CONTENTS

APPENDIXES

FIGURES

TABLES

GENERAL DESCRIPTION

SORT/MERGE FUNCTIONS

The Sort/Merge utility package rearranges records from one or more files into an operator-specified order. Sort/Merge uses one key or a series of keys selected by the operator as the criteria for sorting or merging presorted files. If several keys are used, the order in which they are entered in the control instruction determines the hierarchy of sorting. The most important key is specified first and is the basis for the primary sorting. The key specified second determines sorting within the primary sorting and the key specified third determines sorting within the second sorting, etc.

Three variations of sort/merge processing are available:

- Sort – Sorts records from one or more input files. The total number of records that can be sorted is a function of the size of core space available to Sort/Merge. Output is a single sorted file.
- Merge-only – Combines two or more presorted files into a single output file
- Copy-only – Copies one or more files to an output device

The copying operation neither sorts nor merges the input files. The merging operation requires files of records presorted to the same keys.

The sorting operation may be performed without merging if the available core is sufficient to hold all the records at one time. Otherwise, records are sorted in groups, and then group pointers are merged in successive passes. Finally, at output time, the sorted and merged pointers are used to retrieve the records. The output of Sort/Merge is the sequence of sorted records in the form of a single output file. When Sort/Merge determines that the merging operation is needed for a sort run, the operator sees little indication of the program choice; he sees only a brief message informing him that intermediate merging is being performed. The operator has no direct control over the intermediate merging. He has only indirect control in that he may increase or decrease the number of records to be sorted on a single run, or may make a greater or lesser amount of core space available to Sort/Merge as a part of the total computing system.

A typical sort request with merging is shown in figure 1-1.

The following features are included in the Sort/Merge package:

- Control language with diagnostics and recovery
- Indefinite number of input files
- Indefinite number of key fields
- Four types of key fields:
  - Logical Binary – Unsigned binary integers assumed to be non-negative and sorted by magnitude. The range is 0 to $FFFF_{16}$.
  - Signed Binary – Same as unsigned binary except both positive and negative integers are possible and the range is 8000_{16} to $7FFF_{16}$, ($0000_{16} = FFFF_{16}$) for sorting purposes.
  - Floating Point – A 32-bit key. CYBER-18/1700 format is used.
  - Character – One or more eight-bit characters as specified by the operator
- Each key field may be separately ascending or descending.
- Run-time error detection – I/O status checking is automatic. The user may select sequence checking for a sort or merge-only run. If he does so, record and sequence counts are listed and checked.
- Input (control or user files) may be entered from cards, a teletypewriter, paper tape, magnetic tape, pseudo tape, or disk (file manager, not for control).
- Output (listing or user files) may be made on cards, teletypewriter, printer, paper tape, magnetic tape, pseudo tape, or disk (file manager, not for listing).
- Sort/Merge provides variable levels of program interaction with the operator; i.e., prompting. The operator selects the desired level at the start of the run. The levels are:
  - Noninteractive – No prompting (level 0)
  - Interactive – Names statement type (level 1)
  - Interactive – Lists statement format (level 2)
- Input files may be controlled by a skip count (to discard a group of leading records) and by a do count (to process or given number of records in the file).
- Run-time prompting for the user input file. At this time the user may supply and ready the file, or he may delete the file.
- User file parameters are as follows:
  - File manager file number
  - ASCII or binary
  - Logical unit
  - Record length (input only)
  - Block size

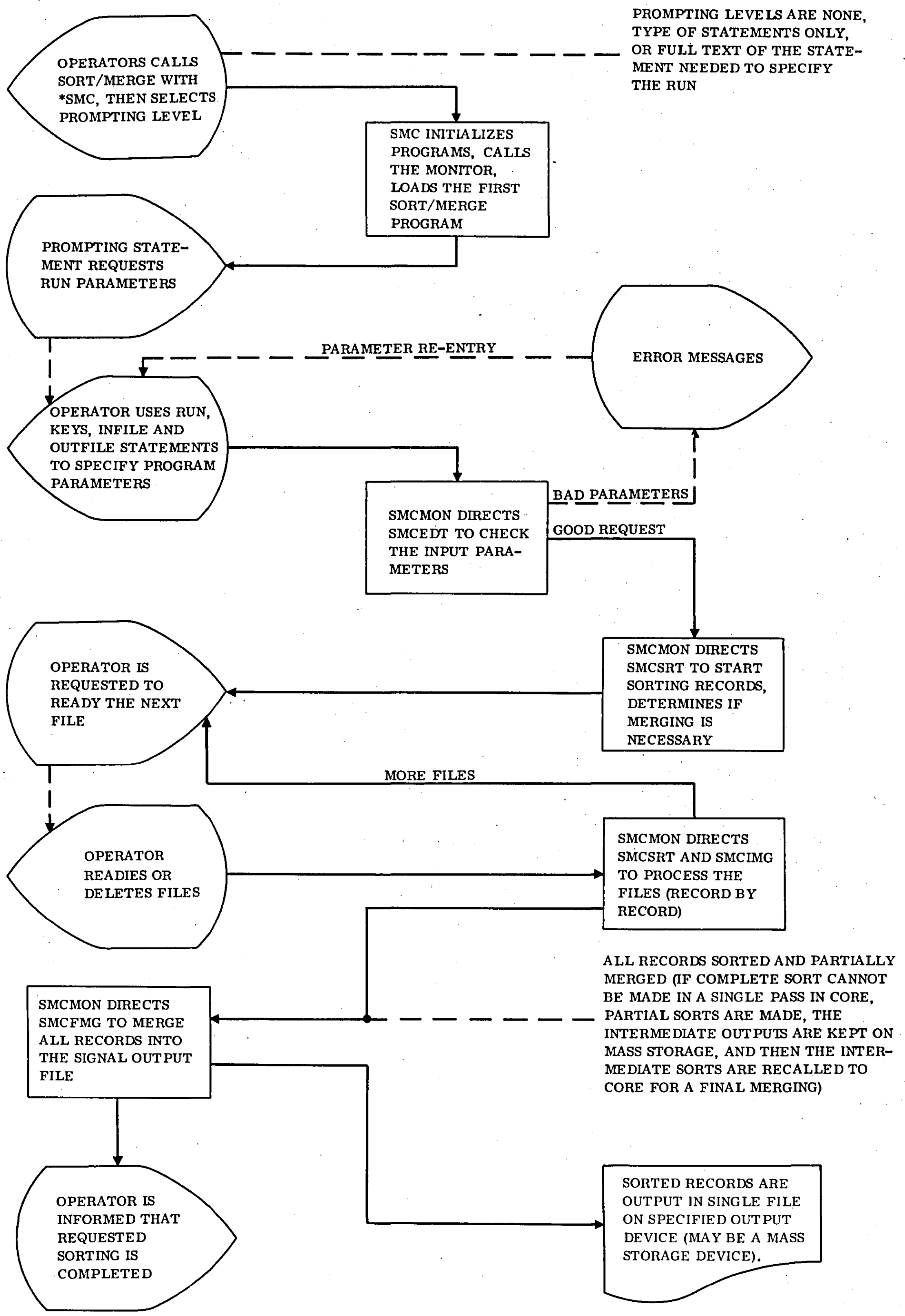

OPERATORS CALLS SORT/MERGE WITH *SMC, THEN SELECTS PROMPTING LEVEL
PROMPTING LEVELS ARE NONE, TYPE OF STATEMENTS ONLY, OR FULL TEXT OF THE STATEMENT NEEDED TO SPECIFY THE RUN
SMC INITIALIZES PROGRAMS, CALLS THE MONITOR, LOADS THE FIRST SORT/MERGE PROGRAM
PROMPTING STATEMENT REQUESTS RUN PARAMETERS
PARAMETER RE-ENTRY
ERROR MESSAGES
OPERATOR USES RUN, KEYS, INFILE AND OUTFILE STATEMENTS TO SPECIFY PROGRAM PARAMETERS
BAD PARAMETERS
GOOD REQUEST
SMCMON DIRECTS SMCEDT TO CHECK THE INPUT PARAMETERS
OPERATOR IS REQUESTED TO READY THE NEXT FILE
SMCMON DIRECTS SMCSRT TO START SORTING RECORDS, DETERMINES IF MERGING IS NECESSARY
MORE FILES
OPERATOR READIES OR DELETES FILES
SMCMON DIRECTS SMCSRT AND SMCIMG TO PROCESS THE FILES (RECORD BY RECORD)
ALL RECORDS SORTED AND PARTIALLY MERGED (IF COMPLETE SORT CANNOT BE MADE IN A SINGLE PASS IN CORE, PARTIAL SORTS ARE MADE, THE INTERMEDIATE OUTPUTS ARE KEPT ON MASS STORAGE, AND THEN THE INTERMEDIATE SORTS ARE RECALLED TO CORE FOR A FINAL MERGING)
SMCMON DIRECTS SMCFMG TO MERGE ALL RECORDS INTO THE SIGNAL OUTPUT FILE
OPERATOR IS INFORMED THAT REQUESTED SORTING IS COMPLETED
SORTED RECORDS ARE OUTPUT IN SINGLE FILE ON SPECIFIED OUTPUT DEVICE (MAY BE A MASS STORAGE DEVICE).
0334

Figure 1-1. Typical Sort Request with Merging

- Skip count (input only)
- Do count (input only)

- Sort/Merge runs in background mode under MSOS.

PROGRAM SUMMARY

Sort/Merge is executed in six phases:

- Phase 1 – Calling Sort/Merge module (SMC)
- Phase 2 – Executing the programs (SMCMON)
- Phase 3 – Defining and checking the run definition (SMCEDT)
- Phase 4 – Sorting (SMCSRT)
- Phase 5 – Intermediate merging (SMCIMG)
- Phase 6 – Final merging (SMCFMG)

The job processor is activated as usual (i.e., *JOB). Then, before calling Sort/Merge, the operator selects the input and list devices using a *K,Ix,Ly statement. This causes the run definition to be accepted from logical unit x and the general output of all messages to be made to logical unit y. For example, if the comment device is the conversational display terminal (CDT) and is used both for run definition and for comments to the operator, and if the CDT is assigned to I/O channel 1, *K,Ix,Ly may be used. Sort/Merge is then activated with an *SMC statement.

After SMC is loaded and activated, the MSOS GTFILE routine is used to load the Sort/Merge monitor, SMCMON (phase 2). When SMCMON is loaded, SMC activates it. SMCMON controls all further operations. SMCMON immediately loads the editor, SMCEDT (phase 3), which receives the operator's definition of the run, checks it for accuracy, and informs the operator of selected run parameters if the operator so requests.

After the run is defined, SMCEDT is ejected as SMCMON loads the sorter, SMCSRT (phase 4). If merging is required (for example, not all files could be sorted within core, and intermediate disk storage of files is necessary), SMCSRT is ejected by the Sort/Merge monitor and the intermediate merge program, SMCIMG, is loaded and executed (phase 5). Finally, in all cases, the Sort/Merge monitor loads and executes the final output processing program, SMCFMG (phase 6). Then control returns to the Sort/Merge monitor, which sends a message to the operator informing him that the sort/merge/copy request has been completed. The operation is summarized in table 1-1.

More details of program operation are described in appendixes C and D.

TABLE 1-1. SORT/MERGE MAJOR PROGRAMS

Program	Definition
SMC	Initializes the Sort/Merge module
SMCMON	Sort/Merge monitor: • Contains Sort/Merge tables • Contains display messages • Contains user input decoder • Monitors I/O and all program execution • Checks key fields
SMCEDT	Interprets user's input parameters: • Selects type of prompting for interaction with the user (see section 4) • Checks user's input parameters
SMCSRT	Prepares files for copying or sorting • Queues files for burst-rate copying • Prepares sequences of files for recursive merging • Counts records that are processed
SMCIMG	Merges input files for certain conditions: • Premerges sequences • Continues to build sequence directory • Optimizes merging sequence • Processes errors
SMCFMG	Merges input and processed (work) files: • Performs final merger of files

NOTE

Sort/Merge may be run in any of three interactive modes: level 0 (no operator interaction), level 1 (limited interaction), or level 2 (complete interaction by use of detailed information and error messages). Level 2 mode requires the most time and operator assistance, but has the advantage that extensive error recovery is available. On the other hand, if the file manipulation techniques have been fully debugged, and the input files are standardized and error-free, level 0 mode provides quick and efficient operation.

SORT/MERGE REQUIREMENTS 2

HARDWARE REQUIREMENTS

Sort/Merge operating under MSOS requires the MSOS hardware minimum requirements. This consists of one CPU (logic, memory, and I/O control), one teletypewriter or conversational display terminal (input/listing/comment device), and one disk drive. In addition, one I/O device is required for each simultaneous user file not on disk; e.g., magnetic or paper tape, teletypewriter, printer, or conversational display terminal.

Disk area must be available for file manager processing and for user and/or sort-only work files, whether those files are pseudo tapes or sequential file manager files.

SOFTWARE REQUIREMENTS

The Sort/Merge utility package requires MSOS with the file manager and with drivers for the input/list/comment device (teletypewriter or conversational display terminal) and for disk. In addition, a driver is needed for each extra I/O device such as a printer, magnetic tape, or paper tape.

RESTRICTIONS

Source language editing flags many user errors. Errors that are not discovered at source language conversion time are generally diagnosed at run time.

Magnetic tape input is limited to a single volume (reel) for each input file or output file. Output to a file manager file is restricted to sequential mode.

Sort/Merge runs should not be terminated by use of the MSOS JOBKIL procedures (*Z or DU) unless the run is successfully completed. These procedures may not release temporary files set up for Sort/Merge. Proper run termination for an unsuccessful run is accomplished by making the input (or output) device not ready. Then when the error/action message is sent to the comment unit, the operator should enter QT (options are GO, QT, or BY), which will cause Sort/Merge to abort the operation in this context.

I/O FORMATS

For Sort/Merge, a block is the information transferred by one of the following MSOS requests: FWRITE, FREAD, STOSEQ, RTVSEQ. STOSEQ and RTVSEQ are used on file manager disk files. Nondisk files are accessed via FWRITE and FREAD.

Since these requests are standard MSOS requests, the MSOS reference manual should be consulted for a complete description of each request. Note particularly that FWRITE and FREAD operations are not the same for each hardware device.

For FREAD operations, Sort/Merge attempts to detect oversize blocks by trying to read one more word than the user-specified maximum block size for the file concerned. If similar oversize blocks are detected, Sort/Merge issues an error message.

For file manager files, Sort/Merge does not need an extra word of buffer specifically for oversize block detection, since the file manager contains logic to detect this type of error. Sort/Merge notifies the operator by outputting the REQIND contents when they are abnormal. Results of the check are passed to Sort/Merge as a bit within the REQIND status buffer.

However, for the STOSEQ and RTVSEQ requests, the buffer is one word larger than the user-specified block size to allow for the length word portion of a file record.

Considering only the data words of blocks (file records have the length word as well) for files containing logical records, Sort/Merge verifies that each such data area is a nonzero multiple of the user-specified logical record size.

Appendix J describes the Sort/Merge-file manager relationship in greater detail.

USING SORT/MERGE

This section describes the operator-entered program control commands necessary to perform a sort, merge-only, or copy run. It is assumed that the job processor is in control.

Before calling Sort/Merge, the operator must define the standard input and list devices for the run. This is done with the following command:

```
*K,Ix,Ly
```

Where: x is the logical unit from which Sort/Merge reads its first control statement before doing any sorting, merging, or copying.

y is the logical unit upon which all listing is done.

Once the desired run has been defined and started, further control statements are requested, as needed, via logical unit x, but are always accepted from the comment device.

Calling of Sort/Merge must also be preceded by defining and opening any pseudo tapes to be used in the run. Pseudo tapes can serve as files for control, listing, user input, or user output.

The first program of the Sort/Merge module, SMC, is then loaded and executed with an *SMC command. SMC automatically calls and executes the second Sort/Merge program, SMCMON. That program starts the interactive phase (messages are sent to the operator announcing the major program phases even if no prompting is selected). From this point forward, operating Sort/Merge is largely a matter of responding to the prompting messages or – in prompting level 0 – entering the run parameters without prompting.

The prompting messages are listed in their normal order of occurrence in figure 4-1, and all messages are described in detail in section 4.

The operator replies to the messages become the source language for the run parameters.

SOURCE LANGUAGE

The parameters of the source language must be submitted in a certain order. Other than this, the source language is free-form in the sense that blanks are ignored, numbers may vary from one to eight digits, and an individual control statement may span an arbitrary number of physical records; e.g., cards, lines on a teletypewriter, tape blocks, etc. Note that two statements may not share the same physical record; i.e., a new statement must start a new physical record.

Each line (physical record) of source language is listed exactly as received so the operator may have some control over the listing of his parameters, and so that he knows exactly what parameters were received by Sort/Merge.

There are five types of source statements, and they must be entered in the order shown:

1. Prompting level statement
2. Run statement
3. Keys statement
4. Input file statement
5. Output file statement

These five control statements are summarized in table 3-1.

Sort runs and merge runs involve all five statement types; the KEYS statement is omitted for copy runs.

Within each statement, each parameter must be followed by a comma unless it is the last parameter and the value is not numeric.

Only the parameter order shown in table 3-1 is permitted. Within each parameter order, each parameter is required and may not be omitted.

A Z may be typed in place of any comma or parameter that allows the operator to recover from his own errors. On entry, this causes the statement involved to be restarted from the beginning, including prompting.

PROMPTING LEVEL STATEMENT

The first source language statement is simply a number, called the prompting level, which tells Sort/Merge how elaborately to prompt the operator as he submits the rest of the source language.

There are three levels of prompting (0, 1, or 2) specified as one of those numbers followed by a comma.

Sort/Merge reads the prompting level statement following the message

```
EDIT BEGINS
```

See the Sample Runs section.

With minimum prompting (prompting level 0), Sort/Merge silently accepts the source language until the first mistake is noticed, whereupon a single diagnostic is issued and the run terminates.

TABLE 3-1. PRINCIPAL CONTROL STATEMENTS FOR SORT/MERGE (SOURCE LANGUAGE)

Statement†	Comments
<u>Prompting level</u> <>,	Level is 0, 1, or 2: 0 = No prompting and no error recovery 1 = Statement named, limited error recovery 2 = Statement and parameters indicated. Full error recovery
<u>Run selection</u> RUN = D, wkbksz, S/N, keycnt, filcnt, cr M, S/N, keycnt, filcnt, cr C, filcnt, cr	Operator selects one of three run types. Run type is D, M, or C D = Sort M = Merge only C = Copy only Parameters are: wkbksz = Size of working area required S/N = Select or ignore file sequence checks keycnt = Number of search keys filcnt = Number of files for run cr = Carriage return
<u>Keys specification</u> KEYS = . . . L/S/F, A/D, keycol, ... C, A/D, keycol, keycols, . . .	Operator specifies keycnt number of keys. One KEYS statement is made for each key used. Statements are ordered by importance of key. L/S/F or C are key types: L = Logical binary S = Signed binary F = Floating point C = Character Parameters are: A/D = Ascending or descending order keycol = Starting column of keyword keycols = Number of characters in character keyword

<u>Input file parameters</u>
INFILE =
D, filnum, reclth, blksiz, skipcnt, docnt, cr
T, lun, reclth, blksiz, skipcnt, docnt, cr
P, A/B, lun, reclth, blksiz, skipcnt, docnt, cr
	Operator selects one of three input file types.
	D, P, or T specifies type of input files:
	D = Disk type
	P = Binary or ASCII (paper tape type)
	T = Binary (magnetic tape type)
	Parameters are:
	filnum = Disk file identification
	lun = Input logical unit
	A/B = ASCII or binary
	reclth = Standard record length
	blksiz = Number of record words Sort/Merge reads/ files

†Statements are shown to their prompting level 2 format.

TABLE 3-1. PRINCIPAL CONTROL STATEMENTS FOR SORT/MERGE (SOURCE LANGUAGE) (Continued)

Statement	Comments
Input file parameters (continued)	skipcnt = Number of leading records to skip/file docnt = Number of records to process/file cr = Carriage return

Output file parameters
OUTFILE =
D, filnum, lun, blksiz, cr
T, lun, blksiz, cr
P, A/B, lun, blksiz, cr
	D, P, or T specifies type of output file:
	D = Disk type
	P = Binary or ASCII (paper tape type)
	T = Binary (magnetic tape type)
	Parameters are:
	filnum = Operator-specified output file identification
	lun = Output device
	A/B = ASCII or binary
	blksiz = Size of output file
	cr = Carriage return

With some prompting (prompting level 1), Sort/Merge issues a brief message naming the desired statement type before reading the user reply. With this level of prompting, a mistake does not terminate the run. Rather, the operator is given the diagnostic:

EXPECTED <parameter name>

FOUND <first character of erroneous reply>

The operator may simply retype the incorrect parameter and its successors.

Maximum prompting (prompting level 2), is the same as some prompting, except is announces the format of each statement rather than the name of the desired statement.

Prompting levels 1 and 2 are the interactive mode of Sort/Merge, while prompting level 0 is noninteractive.

Parameters for a frequently used run could be debugged interactively and then repeatedly used noninteractively.

RUN STATEMENT

The RUN statement summarizes the nature of the run using two to five parameters. There are three formats for the RUN statement, corresponding to the three types of runs: D = sort, M = merge-only, and C = copy.

RUN =

D, wkbksz, S/N, keycnt, filcnt, cr M, S/N, keycnt, filcnt, cr C, filcnt, cr	Operator chooses one of these three options

Sort Format

The sort format of the RUN statement is as follows:

D, wkbksz, S/N, keycnt, filcnt, cr

D is disk sort; i.e., the work files are held on disk.

wkbksz is the block size of those work files that are sequences, destined to be merged into the user output file. wkbksz should be a multiple of the logical record size; otherwise a diagnostic is issued when the logical record size is submitted in an INFILE statement. wkbksz strongly affects sort performance. For an analysis of selecting wkbksz, see appendix G.

As released, Sort/Merge requires a wkbksz of 92 words or less, but this is an assembly option. Depending on other Sort/Merge parameters, the maximum wkbksz for a 32K machine is on the order of 3000 words. However, such a maximum is not imposed until all Sort/Merge parameters have been submitted, so that 92, . . ., 32767 is the allowable range at the time wkbksz is submitted. For wkbksz, 500 words are usually a reasonable order of magnitude. For S/N, S selects sequence-checking of all merge output, whether intermediate or final, and N specifies no sequence checking. S should normally be specified to guard against sequence errors due to undetected hardware (especially I/O) problems. When sequence checking is enabled and a sequence error is detected, a run-time message is listed. This message includes a hexadecimal dump of the new (out-of-sequence) record and of the last record output. The user is then given the option of including or deleting the new record.

N is used on those occasions when there are hardware problems, but perfection is not worth the expenditure of listing errors or requiring user intervention.

keycnt is the number of user key fields to be defined for the run. Sort/Merge does not finish processing the KEYS statement (which is entered next) until the user has input keycnt key fields.

Depending on other Sort/Merge parameters, the maximum keycnt for a 32K machine is on the order of 5000. However, such a maximum is not imposed until all Sort/Merge parameters have been submitted, so that 1, . . ., 32767 is the allowable range at the time keycnt is submitted.

Most Sort/Merge runs use a keycnt of 10 or less.

filcnt is the number of user input files to be defined for the run. Sort/Merge expects that filcnt INFILE statements follow the KEYS statement and precede the OUTFILE statement.

At run time, Sort/Merge requests and reads these files serially in the order of definition.

Depending on other Sort/Merge parameters, the maximum filcnt for a 32K machine is on the order of 1000 for a sort-only or copy run, and 200 for a merge-only run. However, such a maximum is not imposed until all Sort/Merge parameters have been submitted, so that 1, . . ., 32767 is the allowable range at the time filcnt is submitted.

cr is a carriage return.

Merge-Only Format

The merge-only format of the run statement is as follows:

M, S/N, keycnt, filcnt, cr

M specifies a merge-only run.

S/N, keycnt, and filcnt are the same as for a sort run, except that at run-time all filcnt files are merged in parallel.

M,S,1,1 cr is a legal command and an alternate method of copying a file. It could be used to sequence check a file while copying it to another file, real or dummy.

M,N,1,1 cr could also be used for a copying operation instead of C,1 cr (see below), and would take slightly less core since no sequence checking would occur.

Copy Format

C, filcnt, cr

C selects a copy run.

filcnt is used as in a sort run. (A copy run uses a subset of the logic of the internal sort of a sort-only run.)

KEYS STATEMENT

The format of the KEYS statement is as follows.

KEYS =

definition of first (most major) key, definition of second key, . . ., definition of keycnt (most minor) key, cr	Operator specifies a total of keycnt separate keys.

The user defines all keycnt keys, in order of importance, starting on the left with the most important key, and ending on the right with the least important (the keycnth) key.

All the keys defined must be wholly containable within the logical record length, or else that logical record length gets a diagnostic when it is specified in an INFILE statement. Key fields are allowed to overlap. This is rarely justified, however, and if not justified, it will waste CPU time and core space.

For example, S,A,7,L,D,7, cr† would make -0 precede +0, whereas otherwise they might intermingle. However, L,D,7,S,A,7, cr makes no sense since S,A,7, is not used unless L,D,7, ties. But if L,D,7, ties, then S,A,7, also ties.

There are four types of keys and two formats for defining them.

L/S/F Format

The L/S/F format of key definition is as follows:

L/S/F, A/D, keycol,

Three of the four key types use this format: logical binary, signed binary, and floating point.

L signifies logical binary. This is 16 bits of unsigned binary on a 16-bit boundary; i.e., $FFFF_{16}$ is the maximum value for such a key, while 0000_{16} is the minimum value.

S signifies signed binary. This is 16 bits of ones complement signed binary on a 16-bit boundary; i.e., $7FFF_{16}$ is the maximum value, $0000_{16} = FFFF_{16}$; 8000_{16} is the minimum value.

F signifies floating point. This 32-bit key must start on a 16-bit boundary. The key format is CYBER 18/1700 floating point.

In the A/D statement, A signifies ascending order and D signifies descending order, relative to the key viewed as a number. The proper order for an L,A, . . . key could be 1,2,3 . . .; or the proper order for an S,D, . . . key could be $7FFF_{16}$, $7FFE_{16}$, . . ., 0000_{16}, . . ., 8000_{16}.

keycol is where the key starts, relative to the beginning of a logical record, where 1 means the first column of the record. For example, L,A,9 means logical binary, ascending order, starting in the ninth column of a logical record.

For these three key types, the 16-bit boundary restriction implies that keycol must be an odd number.

†S,A,7 = Signed binary, ascending order, column 7 start
L,D,7 = Logical binary, descending order, column 7 start

Depending on other Sort/Merge parameters, the maximum keycol for a 32K machine is on the order of 11,000. However,the first maximum is imposed when reclth is known. The ultimate maximum is imposed when all Sort/Merge parameters are known. At the time keycol is submitted, the allowable range is 1, . . ., 32767.

Key length is not a parameter for these three key types, since their lengths are fixed, not variable.

C Format

The C format of key definition is as follows:

C, A/D, keycol, keycols,

C signifies the character key, i.e., a field of eight-bit bytes starting on an eight-bit boundary, viewed as a single logical binary number whose length is some multiple of eight bits.

A/D and keycol are the same as for the L/S/F format, except that keycol may be an even number, as well as an odd number, since an eight-bit boundary applies.

keycols is the key length as a number of eight-bit bytes, i.e., a C key is variable in length.

keycols is subject to the same legal value range as keycol.

INFILE STATEMENT

Each of the filcnt user input files must be defined by a separate INFILE statement in the order in which the user wants Sort/Merge to request those files at run time. At run time, Sort/Merge asks the user to ready each of those files, but gives the user the alternative of deleting any or all of those files.

There are three different formats of the INFILE statement. D indicates disk, T indicates magnetic tape, and P indicates paper tape or printout. These are usable for other media as well. The INFILE statement formats are as follows:

INFILE =

D, filnum, reclth, blksiz, skipcnt, docnt, cr	Operator chooses one of these three statements
T, lun, blksiz, skipcnt, docnt, cr	
P, A/B, lun, reclth, blksiz, skipcnt docnt, cr	

D Format

The D format of the INFILE statement is as follows:

D, filnum, reclth, blksiz, skipcnt, docnt, cr

D signifies the disk file-manager file. See appendix J for a summary of the outputs of the file manager used by Sort/Merge.

filnum identifies the file and must be one of these numbers: 1,2,3, . . ., 32767.

reclth is the length in words of a Sort/Merge logical record.

reclth must be an even divisor of all user-specified block sizes for the run; i.e., wkbksz and all blksizs.

reclth must be the same for each INFILE statement and must be large enough to entirely contain each user-specified key field for the run. Depending on the other Sort/Merge parameters, the maximum reclth for a 32K machine is on the order of 2000 for a sort-only run, and 5000 for a merge-only or copy run. When wkbksz has not been specified, the legal reclth range is 1, . . ., 32767 when first submitted. Very soon, however, other parameters such as blksiz act as constraints.

blksiz is the number of words of logical records that Sort/Merge reads during a read request on the file. For a file manager file, blksiz is the number of data words in a file record. blksiz may differ for each INFILE statement, but each blksiz must be a nonzero multiple of reclth.

At run time, Sort/Merge checks the length of each block (of Sort/Merge logical records) read, to verify that the actual blksiz is a nonzero multiple of the specified reclth, and to verify that the actual blksiz does not exceed the specified blksiz. Whenever violations of these constraints are detected, corresponding diagnostics are issued.

Depending on other Sort/Merge parameters, the maximum blksiz for a 32K machine is on the order 11,000. However, a range of 1, . . ., 32767 is legal until other parameters impose constraints.

skipcnt is the number of leading Sort/Merge logical records to discard for the current file, before processing any logical records. It must be one of these numbers: 1,2,3, . . ., 99 999 999.

docnt is the number of Sort/Merge logical records to process for the current file, and must be one of these numbers: 1,2,3, . . ., 99 999 999, or E, which means "do every record up to the end of the file".

T Format

The T format of the INFILE statement is as follows:

T, lun, blksiz, skipcnt, docnt, cr

T is a mnemonic for magnetic tape, but this format may be used for all files for which a binary FREAD is appropriate, including pseudo tapes.

lun is the logical unit number. Sort/Merge checks that there is such a logical unit number, but does not check whether the corresponding physical device is appropriate. An improper device would become evident, because a run-time diagnostic would be sent to the operator as a result of Sort/Merge checking status for its MSOS requests.

blksiz is generally equivalent to maximum physical record size; other than that, the comments made for blksiz for the D format apply.

reclth, skipcnt, and docnt are the same as for the D format.

P Format

The P format of the INFILE statement is as follows:

P, A/B, lun, blksiz, reclth, skipcnt, docnt, cr

P is the mnemonic for paper tape or printout, but this statement is equivalent to the T format with the added ability to specify ASCII recording mode, as well as binary.

In A/B, A means ASCII and B means binary.

OUTFILE STATEMENT

This is the last statement and is used to define the output file.

There are three formats for the OUTFILE statement, corresponding to the three INFILE formats, and using a subset of the same parameters:

OUTFILE =

D, filnum, blksiz, cr

T, lun, blksiz, cr

P, A/B, lun, blksiz, cr.

Except for adding lun to the D format, each OUTFILE format is derived from its corresponding INFILE format by deleting reclth, skipcnt, and docnt.

lun was added to the D format used in defining the output file, which is first released and is then redefined so as to begin at record 1 if the file is already defined. A runtime diagnostic results if the actual blksiz of such a predefined file is less than the specified blksiz.

reclth is inferred from the INFILE statement(s), and must evenly divide the specified blksiz.

skipcnt and docnt represent options not available for the output file.

NOTE

If an MSOS file is specified in the OUTFILE statement, the file is released and redefined as an MSOS sequential file to ensure that the output is stored starting at record 1.

CAUTION

Only MSOS sequential files may be used for Sort/Merge output. Sort output may be specified as the same MSOS FILNUM as was used for the input file. The file is reset to record 1 before the output operation begins. If the output is made using a physical device, the device is not rewound.

D Format

The D format of the OUTPUT statement is as follows:

D, filnum, lun, blksiz, cr

T Format

The T format of the OUTFILE statement is as follows:

T, lun, blksiz, cr

P Format

The P format of the OUTFILE statement is as follows:

P, A/B, lun, blksiz, cr

SAMPLE RUNS

In the four runs shown in tables 3-2 through 3-5, the data being sorted is assembly language for part of Sort/Merge. The source language output (the sorted records, each one a program statement) has been deleted from the listing. This would be normal if the output of the run were assigned by the OUTFILE statement to some device other than the listing device.

TABLE 3-2. SAMPLE SORT RUN WITHOUT MERGING WITH LEVEL 2 PROMPTING (*K,I4,L6)

Sample Sort Run	Comments
*JOB	Job processor requested and placed in control
J *K,I4,L6	Standard input from device on logical unit 4; listing on device on logical unit 6
J *SMC	Calls Sort/Merge
SMC BEGINS EDIT BEGINS 2,	Level 2 (full prompting) requested
RUN= D,<WKBKSZ>,<S/N>,<KEYCNT>,<FILCNT>,<CR> M,<S/N>,<KEYCNT>,<FILCNT>,<CR> C,<FILCNT>,<CR> AWAITING REPLY	Possible run parameters and request for response
D,120,S,1,1,	Selected run parameters: disk type, 120 word block, sequence checked desired; one key, one file
KEYS= . . .L/S/F,A/D,<KEYCOL>, ...C,A/D,<KEYCOL>,<KEYCOLS>, . . . AWAITING REPLY	Possible keys parameters and request for response
C,D,76,5,	Selected keys parameters: character mode, descending order, starts in column 76, five key columns
INFILE 0001= D,<FILNUM>,<RECLTH>,<BLKSIZ>,<SKIPCNT>, <DOCNT>,<CR> T,<LUN>,<RECLTH>,<BLKSIZ>,<SKIPCNT>, <DOCNT>,<CR> P,A/B,<LUN>,<RECLTH>,<BLKSIZ>,<SKIPCNT>, <DOCNT>,<CR> AWAITING REPLY	Possible input file parameters and request for response
P,A,17,40,40,0,200,	Selected input file parameters: P type, ASCII code from logical unit 17, 40 word records and blocks, no SKIPCNT, DOCNT = 200
OUTFILE= D,<FILNUM>,<LUN>,<BLKSIZ>,<CR> T,<LUN>,<BLKSIZ>,<CR> P,A/B,<LUN>,<BLKSIZ>,<CR> AWAITING REPLY	Possible output file parameters and request for response
TYPE-IN ERROR TYPE-IN ERROR TYPE-IN ERROR	Operator was slow in replying; program repeated request for reply until operator answered.
P,A,6,40,	Selected output file parameters: P type, ASCII code, to logical unit 6, 40 word blocks
G = 0271	Tournament is run with less than 200 (all) records in one pass

IWAY = 0074
FWAY = 0074
	Seventy-three 120-word buffers wkbksz are used for sorting, plus one buffer for output

INTERNAL SORT BEGINS	

TABLE 3-2. SAMPLE SORT RUN WITHOUT MERGING WITH LEVEL 2 PROMPTING (*K,I4,L6) (Continued)

Sample Sort Run	Comments
READY FILE = 0001 TYPE GO/QT/BY	Operator must ready or delete file 1
GO	File selected for use

LUN = 0017
FILNUM = 0000
	Following message concerns this input file, which is not a file manager file.

PASSED = 00000200	All 200 records are input to the tournament.

NOTE

Before executing with *SMC, the operator selected the standard devices (*K,Ix,Ly) and output device (OUTFILE) so that the listing and output devices were the same unit. The following is the sorted output of the run.

1. RUN = D,120,S,1,1, where D is a disk run, wkbksz = 120 = n*40. (40 is the record size.) One key and one record are used.
2. KEYS = C,D,76,5 where the character key starting in column 76 and extending to column 80 is the programmer's statement identification. Descending order is selected.
3. INFILE = P,A,17,40,40,0,200 where an ASCII input file is read from logical unit 17 (disk) with reclth = blksiz = 40, and no records are skipped. 200 records are inspected.
4. OUTFILE = P,A,6,40 with output 40 word records (same as input) in ASCII format on logical unit 6 (also the listing device). Following the truncated listed output sorting (a part of Sort/Merge), the final program comments are made.

(Q)EXIT = FWA OF LOSER RECORD (A)EXIT = FWA OF WINNER RECORD IF NO TIE, THEN (KEYTBL)EXIT = (KEYTBL) ENTRY (I)EXIT = (I)ENTRY	00200 00199 00198 00197 00197
. . .	. . .
AMONI EQU AMONI($F4) PRLVL EQU PRLV(0) ENT SMC NAM SMC OPT	00005 00004 00003 00002 00001

LUN = 0006
FILNUM = 0000
	Following message concerns the output file on logical unit 6.

PASSED = 00000200	All 200 records appear in the output file.
SEQUENCES = 0001	Number of sequences = 1 (single sorted file)
RECORDS IN = 00000200 RECORDS OUT = 00000200	Number of files output = number of files input; i.e., no files are deleted
SMC ENDS	

TABLE 3-3. SAMPLE SORT RUN WITH MERGING WITH LEVEL 0 PROMPTING (*K,I4,L4)

Sample Sort Run	Comments
*JOB	Job processor requested and placed in control.
J *K,I4,L6	Standard input from device on logical unit 4; listing on device on logical unit 6
J *SMC	Calls Sort/Merge
SMC BEGINS EDIT BEGINS 1,	Level 1 (some prompting) requested
RUN= D,3000,S,1,1,	Run parameters: disk type, 3000 word work blocks, sequence number checking desired, one key, one file
KEYS= C,A,76,5,	Key parameters: character mode, ascending, key starts in column 76, five key columns
INFILE 0001= P,A,17,40,8000,0,300,	Input file parameters: P type, ASCII code, from logical unit 17, 40 word records, 8000 word blocks, no SKIPCNT, 300 DOCNT
OUTFILE= P,A,6,40,	Output file parameters: P type, ASCII code, to logical unit 7, 40 word blocks
G = 0013	Tournaments are run with 13 records per tournament. 40 x 13 = 520 words per RSA
IWAY = 0002	Two 3000 word buffers (WKBKSZ) are used during SMCIMG
FWAY = 0003	Three 3000 word buffers are used during final merging (SMCFMG)
INTERNAL SORT BEGINS	
READY FILE = 0001 TYPE GO/QT/BY	Operator must ready or delete file 1
GO	File selected for use

LUN = 0008
FILNUM = 0179
PASSED = 00000013
	Following messages concern this file which has a file manager identification of 0179. Intermediate storage is on logical unit 8. 13 records are input for tournament 1

LUN = 0008
FILNUM = 0939
PASSED = 00000013
	Sort/Merge strategy has minimized sorting. During the first sort, $\frac{300}{13}$ = [23+] = 24 input tournaments must be performed. First internal sort uses record bins (RSA).

LUN = 0008
FILNUM = 4241
PASSED = 00000013	

LUN = 0008
FILNUM = 5475
PASSED = 00000013	

LUN = 0008
FILNUM = 00029753
PASSED = 00000013	

LUN = 0008
FILNUM = 00017371
PASSED = 00000013	

TABLE 3-3. SAMPLE SORT RUN WITH MERGING WITH LEVEL 0 PROMPTING (*K,I4,L4) (Continued)

Sample Sort Run	Comments

LUN = 0008
FILNUM = 00029217
PASSED = 00000013
	Sort/Merge strategy has minimized sorting. During the first sort, $\frac{300}{13}$ = [23+] = 24 input tournaments must be performed. First internal sort uses record bins (RSA).

LUN = 0008
FILNUM = 00019731
PASSED = 00000013	

LUN = 0008
FILNUM = 00025673
PASSED = 00000013	

LUN = 0008
FILNUM = 7945
PASSED = 00000013	

LUN = 0008
FILNUM = 00023617
PASSED = 00000013	

LUN = 0008
FILNUM = 00022467
PASSED = 00000013	

LUN = 0008
FILNUM = 00023897
PASSED = 00000013	

LUN = 0008
FILNUM = 00017723
PASSED = 00000013	

LUN = 0008
FILNUM = 00026689
PASSED = 00000013	

LUN 0008
FILNUM = 00025971
PASSED = 00000013	

LUN 0008
FILNUM = 00028521
PASSED = 00000013	

LUN = 0008
FILNUM = 00026219
PASSED = 00000013	

LUN = 0008
FILNUM = 7377
PASSED = 00000013	

LUN = 0008
FILNUM = 9763
PASSED = 00000013	

LUN = 0008
FILNUM = 00010873
PASSED = 00000013	

LUN = 0008
FILNUM = 00012955
PASSED = 00000013	

TABLE 3-3. SAMPLE SORT RUN WITH MERGING WITH LEVEL 0 PROMPTING (*K,I4,L4) (Continued)

Sample Sort Run	Comments

LUN = 0017
FILNUM = 0000†
PASSED = 00000300†
	Sort/Merge strategy has minimized sorting. During the first sort, $\frac{300}{13}$ = [23+] = 24 input tournaments must be performed. First internal sort uses record bins (RSA).

LUN = 0008
FILNUM = 00025185
PASSED = 00000013	

LUN = 0008
FILNUM = 00018899††
PASSED = 00000001††	

SEQUENCES = 0024	First and last sequences are not counted
RECORDS IN = 00000300 RECORDS OUT = 00000300	Same number of records are output to SMCIMG as were read in by SMCSRT
INTERMEDIATE MERGE BEGINS 1STWAY = 0002 U = 0072	Number of strings to be merged are unit strings rating

LUN = 0008
FILNUM = 0939
PASSED = 00000013
	First intermediate merge uses arrays of two word lines. Word 0 points to record fixed word address; word 1 points to file table.

LUN = 0008
FILNUM = 0179
PASSED = 00000013
	Intermediate merging rereads the 1STWAY sorted strings and merges them progressively into longer strings: two 13-pointer strings into one 26-pointer string (12 repetitions).

LUN = 0008
FILNUM = 0939
PASSED = 00000026
	New values are stored in the previous FILNUM space.

LUN = 0008
FILNUM = 5475
PASSED = 00000013	

LUN = 0008
FILNUM = 4241
PASSED = 00000013	

LUN = 0008
FILNUM = 0179
PASSED = 00000026	

LUN = 0008
FILNUM = 00017371
PASSED = 00000013	

LUN = 0008
FILNUM = 00029753
PASSED = 00000013	

LUN = 0008
FILNUM = 0939
PASSED = 00000026	

†This is a preliminary sorting of all the records.

†† $\frac{300}{13}$ = 26 + 1/13. Last record must be read separately.

TABLE 3-3. SAMPLE SORT RUN WITH MERGING WITH LEVEL 0 PROMPTING (*K,I4,L4) (Continued)

Sample Sort Run	Comments

LUN = 0008
FILNUM = 00019731
PASSED = 00000013
	Intermediate merging rereads the 1STWAY sorted strings and merges them progressively into longer strings: two 13-pointer strings into one 26-pointer string (12 repetitions).
	
	New values are stored in the previous FILNUM space.

LUN = 0008
FILNUM = 00029217
PASSED = 00000013	

LUN = 0008
FILNUM = 4241
PASSED = 00000026	

LUN = 0008
FILNUM = 7947
PASSED = 00000013	

LUN = 0008
FILNUM = 00025673
PASSED = 00000013	

LUN = 0008
FILNUM = 5475
PASSED = 00000026	

LUN = 0008
FILNUM = 00022467
PASSED = 00000013	

LUN = 0008
FILNUM = 00013489
PASSED = 00000013	

LUN = 0008
FILNUM = 00029753
PASSED = 00000026	

LUN = 0008
FILNUM = 00017723
PASSED = 00000013	

LUN = 0008
FILNUM = 00023897
PASSED = 00000013	

LUN = 0008
FILNUM = 00017 71
PASSED = 00000013	

LUN = 0008
FILNUM = 00026371
PASSED = 00000013	

LUN = 0008
FILNUM = 00026689
PASSED = 00000013	

LUN = 0008
FILNUM = 00029217
PASSED = 00000026	

LUN = 0008
FILNUM = 00026219
PASSED = 00000013	

TABLE 3-3. SAMPLE SORT RUN WITH MERGING WITH LEVEL 0 PROMPTING (*K,I4,L4) (Continued)

Sample Sort Run	Comments

LUN = 0008
FILNUM = 00028521
PASSED = 00000013
	Intermediate merging rereads the 1STWAY sorted strings and merges them progressively into longer strings: two 13-pointer strings into one 26-pointer string (12 repetitions).

LUN = 0008
FILNUM = 00019731
PASSED = 00000026	

LUN = 0008
FILNUM = 9763
PASSED = 00000013	

LUN = 0008
FILNUM = 7377
PASSED = 00000013	

LUN = 0008
FILNUM = 00025673
PASSED = 00000026	

LUN = 0008
FILNUM = 00012955
PASSED = 00000013	

LUN = 0008
FILNUM = 00010873
PASSED = 00000013	

LUN = 0008
FILNUM = 7947
PASSED = 00000026	

LUN = 0008
FILNUM = 00018899
PASSED = 00000001
 	
LUN = 0008
FILNUM = 00025185
PASSED = 00000013
 	
LUN = 0008
FILNUM = 00013489
PASSED = 00000014 †	

LUN = 0008
FILNUM = 0179
PASSED = 00000026
	Second intermediate stage merges two 26-pointer strings into one 52-pointer string (six repetitions).

LUN = 0008
FILNUM = 7817
PASSED = 00000026	

LUN = 0008
FILNUM = 00022467
PASSED = 00000052	

LUN = 0008
FILNUM = 4241
PASSED = 00000052	

LUN = 0008
FILNUM = 00025181
PASSED = 00000026	

†Shorter strings are treated as if they were standard length.

TABLE 3-3. SAMPLE SORT RUN WITH MERGING WITH LEVEL 0 PROMPTING (*K,I4,L4) (Continued)

Sample Sort Run	Comments

LUN = 0008
FILNUM = 0179
PASSED = 00000052
	Second intermediate stage merges two 26-pointer strings into one 52-pointer string (six repetitions).

LUN = 0008
FILNUM = 00029753
PASSED = 00000026	

LUN = 0008
FILNUM = 5475
PASSED = 00000026	

LUN = 0008
FILNUM = 0939
PASSED = 00000052	

LUN = 0008
FILNUM = 00029217
PASSED = 00000026	

LUN = 0008
FILNUM = 00017371
PASSED = 00000026	

LUN = 0008
FILNUM = 4241
PASSED = 00000052	

LUN = 0008
FILNUM = 00025673
PASSED = 00000026	

LUN = 0008
FILNUM = 00019731
PASSED = 00000026	

LUN = 0008
FILNUM = 5475
PASSED = 00000052	

LUN = 0008
FILNUM = 00013489
PASSED = 00000014	

LUN = 0008
FILNUM = 7947
PASSED = 00000026	

LUN = 0008
FILNUM = 00029753
PASSED = 00000040	

LUN = 0008
FILNUM = 0179
PASSED = 00000052
	Third (and final) intermediate stage merges two 52-pointer strings into one 104-pointer string.
	After this stage, SMCFMG can merge all the remaining strings in a single pass.

LUN = 0008
FILNUM = 00022467
PASSED = 00000052	

LUN = 0008
FILNUM = 00017371
PASSED = 00000104	

TABLE 3-3. SAMPLE SORT RUN WITH MERGING WITH LEVEL 0 PROMPTING (*K,I4,L4) (Continued)

Sample Sort Run	Comments

LUN = 0008
FILNUM = 4241
PASSED = 00000052
 	
LUN = 0008
FILNUM = 0939
PASSED = 00000052
 	
LUN = 0008
FILNUM = 0179
PASSED = 00000104
 	
LUN = 0008
FILNUM = 00029753
PASSED = 00000040
 	
LUN = 0008
FILNUM = 5475
PASSED = 00000052
 	
LUN = 0008
FILNUM = 0939
PASSED = 00000092
	Third (and final) intermediate stage merges two 52-pointer strings into one 104-pointer string.
	After this stage, SMCFMG can merge all the remaining strings in a single pass.

DELETES = 0000	No records were deleted.
RECORDS IN = 00000900 RECORDS OUT = 00000900	Three intermediate stages; each processed 300 records; none was lost.
FINAL-MERGE BEGINS	

LUN = 0008
FILNUM = 00017371
PASSED = 00000104
	Final merging performed by SMCFMG. 104 pointer strings are merged to the final 300 pointer string, which is used for the final output.

NOTE

Prior to executing with *SMC, the operator selects his standard devices (*K,Ix,Ly) and later his output device (OUTFILE) so that the listing and output are made on the same device. Both input and output therefore occur in one listing. The following is the sorted output record. The input records are part of the Sort/Merge program itself. Record length is 80 words (enough to take up program statements requiring several cards). The parameters are:

1. RUN = D,120,S,1,1 where 120 = wkbksz = n*40 and 40 is the record size.

2. KEYS = C,D,76,5 which specified a character key, five characters long, starting in column 76, and running in descending order. This key specifies the programmer's sequencing of statement (columns 76 through 80).

3. INFILE = 0001 = P,A,17,40,8000,0,300 (ASCII input from logical unit 17, record length of 40, no records are to be skipped in this record, and there are 300 records to be input from this file).

4. OUTFILE = P,A,6,40 (ASCII output on logical unit 6, with 40 word records; i.e., the same length records as used in the input records).

Only a truncated portion of the output is shown below. This is followed by the end of the program.

TABLE 3-3. SAMPLE SORT RUN WITH MERGING WITH LEVEL 0 PROMPTING (*K,I4,L4) (Continued)

Sample Sort Run	Comments
SAM CMPSA1 SKIP IF AREC -. 00301 *AREC +. 00302 AAQ A AREC +, (A) = AREC - QREC. 00303 . . . . . . *2-WORD-MINUEND - 2-WORD-SUBTRAHEND = 2-WORD-DIFFERENCE. 00597 *. . . ,$270F270F, . . . ,$00010000,$0000270F, . . . , $00000000,$FFFE270F, . . . 00598 *=. . .99999999, . . . ,10000,9999, . . . ,0,-1, . . . RESPECTIVELY. 00599 *THE CALLING SEQUENCE IS (Q) ENTRY = -1 + FWA OF 2-WORD-MINUEND. 00600	

LUN = 0006
FILNUM = 0000
	Following message concerns the output file. It is not a file manager file.

PASSED = 00000300	300 records processed
DELETES = 0000	No records lost
RECORDS IN = 00000300 RECORDS OUT = 00000300	Final merge output same number of records it received as input; i.e., no records were lost at any stage of the program.
SMC ENDS	

TABLE 3-4. SAMPLE MERGE-ONLY RUN WITH LEVEL 0 PROMPTING (*K,I4,L4)

Sample Sort Run	Comments
*JOB	Job processor requested and placed in control
J *K,I4,L4	Select standard input and output listing devices
J *SMC SMC BEGINS EDIT BEGINS	Call Sort/Merge
0, 0,	0 level (no prompting) requested
M,S,1,2, M,S,1,2,	Run parameters: merge only, sequence checking requested, one key, two files
C,A,76,5, C,A,76,5,	Keys parameters: character type key, ascending order, start in column 76, five key columns used
P,A,6,40,40,50,100, P,A,6,40,40,50,100,	Input file 1 parameters: P type, ASCII code, file on logical unit 6 device, 40 word records, 40 word BLKSIZE, skip first 50 records, do 100 records
P,A,7,40,160,0,100, P,A,7,40,160,0,100,	Input file 2 parameters: P type, ASCII code, logical unit 7, 40 word records, 160 word blocks, SKIPCNT = 0, DOCNT = 160

TABLE 3-4. SAMPLE MERGE-ONLY RUN WITH LEVEL 0 PROMPTING (*K,I4,L4) (Continued)

Sample Sort Run	Comments
P,A,9,40, P,A,9,40,	Output file parameters: P type, ASCII code, to logical unit 9, 40 word blocks
MERGE-ONLY BEGINS	
READY FILE = 0001 TYPE GO/QT/BY	Operator must choose to ready or delete file 1.
GO GO	File selected for use
READY FILE = 0002 TYPE GO/QT/BY	Operator must ready or delete file 2.
GO GO	File selected for use

LUN = 0007
FILNUM = 0000
	Following messages concern input file from logical unit 7; not a file management file.

PASSED = 00000100	100 records were deblocked and processed.

LUN = 0006
FILNUM = 0000
	Following messages concern input file from logical unit 6.

PASSED = 00000150	150 records were deblocked and used or skipped.
DONE = 00000100	100 of the records were deblocked and used.

LUN = 0009
FILNUM = 0000
	Following messages concern output file on logical unit 9.

PASSED = 00000200
DELETES = 0000
	200 records were used, none were deleted.

RECORDS IN = 00000200 RECORDS OUT = 00000200	Input and output record count
SMC ENDS	
J *Z	Return to job processor and release control.

TABLE 3-5. SAMPLE COPY RUN WITH LEVEL 1 PROMPTING (*K,I4,L6)

Sample Sort Run	Comments
*JOB	Job processor requested and placed in control
J *K,I4,L6	Standard input from device on logical unit 4; listing on device on logical unit 6.
J *SMC	Calls Sort/Merge
SMC BEGINS EDIT BEGINS 1,	Level 1 (some prompting) requested.

TABLE 3-5. SAMPLE COPY RUN WITH LEVEL 1 PROMPTING (*K,I4,L6) (Continued)

Sample Sort Run	Comments
RUN= C,1,	Run parameters: copy run, one file.
INFILE 0001= P,A,18,40,40,0,600,	Input file parameters: P type, ASCII code, logical unit 18, RECLTH = 40 words, BLKSIZ = 40 words, no SKIPCNT, DOCNT = 600
OUTFILE= P,A,19,4000,	Output file parameters: P type, ASCII code, output on logical unit 19, 4000 word blocks
COPY BEGINS	
READY FILE = 0001 TYPE GO/QT/BY	Operator must choose to use or delete file.
GO	File selected for use.

LUN = 0018
FILNUM = 0000
	Following messages concern this file which is not a file manager file.

PASSED = 00000600	600 records were deblocked and processed.

LUN = 0019
FILNUM = 0000
	Following messages concern this file.

PASSED = 00000600	600 records deblocked and processed.
RECORDS IN = 00000600 RECORDS OUT = 00000600	Input and output record count
SMC ENDS	

INPUT, MESSAGES, AND OUTPUT 4

This section describes the operator-entered parameters; the interactive messages prompting the operator to enter parameters, file data, or corrections; and the Sort/Merge output. Depending upon the chosen level of prompting, parameter request messages are nonexistent (no prompting), brief (only an indication that this type of parameter must be entered now), or complete (the format of the parameter is displayed to the operator who then fills in the values).

INPUT

Two types of input are required: the control statements (RUN, KEYS, INFILE, OUTFILE, and prompting level) and the input files. The control statements (and the range of values allowed for each control parameter) are described in section 3.

The input files must be composed of uniform records collected into one or more files or records formatted for the file manager if entered from a file manager controlled medium or output from the file manager. The INFILE command specifies both the data source and the record parameters. The INFILE statement's skip count and do count parameters allow the operator to use any one block of records within a file so long as records in the block are logically sequential (i.e., leading records may be skipped and/or trailing records may be ignored).

No file can be started without the operator's express command. However, at the time the first file is to be started, the operator may specify that this and all succeeding files are to be processed without his further approval. This option, once chosen, is irrevocable for the rest of the run.

If the file is on a medium (e.g., magnetic tape) that requires the data medium to be mounted and readied, the operator must do this before he indicates to Sort/Merge that the file is ready to be processed.

MESSAGES

This section lists each of the program messages, in the order in which it would normally appear. Naturally, if the run is error free and the parameters are supplied correctly upon initial entry, none of the diagnostic messages are displayed. Table 4-1 summarizes the messages in the usual order of appearance.

TABLE 4-1. MESSAGES USED IN SORT/MERGE

Message	Meaning	Phase
SMC BEGINS	Program is beginning.	SMCMON
TYPE-IN ERROR	Error in trying to interpret operator's command	SMCMON, SMCEDT, SMCSRT, SMCIMG, and SMCFMG
ABNORMAL ERROR = <n>	Miscellaneous errors	SMCMON, SMCEDT, SMCSRT, SMCIMG, and SMCFMG
TOO LITTLE CORE	Requested inputs cannot be processed in available core.	SMCMON, SMCEDT
SMC ENDS	Program has been completed.	SMCMON
EDIT BEGINS	Edit is ready for operator inputs using prompting messages.	SMCEDT
RUN = <parameters>	Operator should supply the run parameters.	SMCEDT
KEYS = <parameters>	Operators should supply the file keys.	SMCEDT
INFILE <n> = <parameters>	Operator should supply the input file parameters.	SMCEDT
OUTFILE <n> = <parameters>	Operator should supply the output file parameters.	SMCEDT
EXPECTED <parameter> FOUND<character>	Editor did not find the type of parameter that should have been entered. Operator may be able to correct error.	SMCEDT

TABLE 4-1. MESSAGES USED IN SORT/MERGE (Continued)

Message	Meaning	Phase
G = <n>	Sort-only run. Input has been checked and accepted; n-1 indicates largest number of records that can be sorted in core (if more than n records, Sort/Merge and mass storage are required).	SMCEDT
IWAY = <n>	Sort-only run. Maximum number of wkbksz input string buffers that can be used during intermediate merging.	SMCEDT
FWAY = <n>	Sort-only run. Maximum number of wkbksz string input buffers that can be used during final merging.	SMCEDT
COPY BEGINS	SMCSRT is loaded and is starting a copy run.	SMCSRT
INTERNAL SORT BEGINS	SMCSRT is loaded and is starting a sort-only run.	SMCSRT
READY FILE = <n>	User should ready the file, or direct Sort/Merge to delete or bypass the file.	SMCSRT
FREAD STATUS = <parameters>	Operator may direct the program to reread the file, to delete it, or to continue without operator interaction for format read errors.	SMCSRT, SMCFMG
BLKSIZ/RECLTH≠ <parameters>	Operator may direct the program to reread the file, to delete it, or to continue without operator interaction for the record size type of error.	SMCSRT, SMCIMG, SMCFMG
OVERSIZE BLOCK <parameters>	Operator may direct the program to reread the file, to delete it, or to continue without operator interaction for this block size type error.	SMCSRT, SMCFMG
FWRITE STATUS = <parameters>	Operator may direct program to rewrite the file, to abort the run, or to continue without operator interaction for format write errors.	SMCSRT, SMCFMG
RTVSEQ REQIND = <parameters>	Operator may direct the program to again retrieve the file, to delete it, or to continue without operator interaction for this type of retrieval error.	SMCSRT, SMCIMG, SMCFMG
STOSEQ REQIND = <parameters>	Operator may direct the program to again store the file, to abort the run, or to continue without operator interaction for this type of store error.	SMCSRT, SMCIMG. SMCFMG
LUN = <k> FILNUM = <f>	Following messages concern file f from logical unit k	SMCSRT, SMCIMG, SMCFMG
PASSED = <n>	Specified file composed of n records was either down or skipped.	SMCSRT, SMCIMG, SMCFMG
SEQUENCES = <n>	All completed logical records have been grouped in n sequences following a completed internal sort procedure.	SMCSRT
RECORDS IN = <m> RECORDS OUT = <n>	RECORDS IN specifies the number of records sent from unblocking to processing; RECORDS OUT specifies the number of records for the converse. Unless a record is deleted or lost (hardware error), m = n.	SMCSRT, SMCIMG, SMCFMG
DONE = <n>	Number of records deblocked and processed	SMCSRT, SMCFMG
DEFFIL REQIND = <parameters>	Bad user-defined output file status; run aborted	SMCSRT, SMCIMG SMCFMG

TABLE 4-1. MESSAGES USED IN SORT/MERGE (Continued)

Message	Meaning	Phase
SEGMENT LIST ERROR	Sort-only run. Work file accountability lost; run aborted.	SMCSRT, SMCIMG SMCFMG
SEQ DIR ERROR	Sort-only run. Sequence directory read/write error; run aborted	SMCSRT, SMCIMG SMCFMG
TOO LITTLE DISK	Sort-only run. Inadequate disk space; run aborted	SMCSRT, SMCIMG
INTERMEDIATE MERGE BEGINS	Sort-only run. SMCIMG is ready to start its merging process.	SMCIMG
1STWAY = <n>	Sort-only run. SMCIMG has optimized merge strategy; n is the number of strings used for first merge.	SMCIMG
U = <n>	Sort-only run. SMCIMG has optimized merge strategy; n is the unit strings rating.	SMCIMG
SEQUENCE ERROR	Latest record should have preceded previous record in key merging. Operator may direct program to delete the record or to continue with or without operator interaction for this type of error.	SMCIMG, SMCFMG
RELFIL REQIND = <parameters>	The release file operation failed. Operator may direct program to retry the release, or to continue with or without operation interaction for this type of error.	SMCIMG, SMCFMG
DELETES = <n>	The phase is ended and n records have been deleted.	SMCIMG, SMCFMG
FINAL-MERGE BEGINS	Sort-only run. SMCFMG is ready to start the final merging process.	SMCFMG
MERGE-ONLY BEGINS	Merge-only run. SMCFMG is ready to start the final merging process.	SMCFMG
<n> = 12	Erroneous SMCFMG fixed table size; run aborted	SMCFMG
INTERPHASE RECORD COUNTS DISAGREE	Number of output records does not equal the number of input sort records.	SMCFMG

MONITOR (PHASE 2)

SMC BEGINS

Phase	SMCMON
Type	Informative
Meaning	The first phase (SMC) is in core and initialized. Phase SMC has loaded phase SMCMON, which is now initializing itself for use by SMCEDT, SMCSRT, SMCIMG, and SMCFMG. Phase SMCEDT is loaded and executed immediately following SMCMON.

TYPE-IN ERROR

Phase	SMCMON, SMCEDT, SMCSRT, SMCIMG, SMCFMG
Type	Fatal or action
Meaning	An attempt to FREAD a user reply incurred an error status; e.g., the reply was not typed soon enough.
Action	If SMCEDT is running with prompting level 0, then the error is fatal. Otherwise, the user resupplies the reply.

ABNORMAL ERROR = <n>

Phase	SMCMON, SMCEDT, SMCSRT, SMCIMG, SMCFMG
Type	Fatal or informative
Meaning	Every error should be abnormal, but this message is used to announce several errors

that are abnormal; i.e., the error reflects undetected hardware errors, incorrect installation of Sort/Merge, or a logic error in Sort/Merge.

The following is a list of values of n and the significance of each value:

n = 1

Phase	SMCMON, SMCIMG, SMCFMG
Type	Informative
Meaning	Unexpected RELFIL status return

n = 2

Phase	SMCSRT, SMCIMG, SMCFMG
Type	Informative
Meaning	Unexpected RTVSEQ status return

n = 3

Phase	SMCSRT, SMCIMG, SMCFMG
Type	Informative
Meaning	Unexpected STOSEQ status return

n = 4

Phase	SMCMON, SMCSRT, SMCIMG, SMCFMG
Type	Fatal
Meaning	Illegal work-file logical unit is in use

n = 5

Phase	SMCSRT, SMCIMG, SMCFMG
Type	Fatal
Meaning	Unexpected call to or status from DEFFIL

n = 6

Phase	SMCSRT, SMCIMG, SMCFMG
Type	Fatal
Meaning	BINDEC was called with an argument greater than 9999

n = 7

Phase	SMCEDT
Type	Fatal
Meaning	Fixed tables contain incorrect SMCEDT size

n = 8

Phase	SMCSRT
Type	Fatal
Meaning	Fixed tables contain incorrect SMCSRT size

n = 9

Phase	SMCIMG
Type	Fatal
Meaning	SMCIMG call was unjustified because fway strings or less than fway strings are to be merged yet

n = 10

Phase	SMCIMG
Type	Fatal
Meaning	Fixed tables contain incorrect SMCIMG size

n = 11

Phase	SMCFMG
Type	Fatal
Meaning	SMCFMG call was unjustified because greater than fway strings are to be merged yet

TOO LITTLE CORE

Phase	SMCMON, SMCEDT
Type	Fatal or informative
Meaning	There is not enough core to continue the run, so the run must terminate. If this message appears after the message EDIT BEGINS, the user may be able to complete the run with changed parameters; e.g., lower buffer sizes or fewer files. If this message appears before the message EDIT BEGINS, there is too little core available to run any option of Sort/Merge.

SMC ENDS

Phase	SMCMON
Type	Informative
Meaning	The current Sort/Merge run has terminated normally.

EDITING INPUT RECORDS (PHASE 3)

EDIT BEGINS

Phase	SMCEDT
Type	Action
Meaning	SMCEDT is initialized and is attempting to read the prompting level statement.
Action	Supply the prompting level statement: no prompting = 0, limited prompting = 1, full prompting = 2.

The following four messages: RUN, KEYS, INFILE, and OUTFILE define the run parameters. The parameter values and their calculation are described in detail in section 3.

```
RUN = <— prompting level 1 version
D, <WKBKSZ>, <S/N>, <KEYCNT>, <FILCNT>,
 <CR>                                       prompting
M, <S/N>, <KEYCNT>, <FILCNT, <CR>           level 2
C, <FILCNT>, <CR>                           version
AWAITING REPLY
```

Phase	SMCEDT
Type	Action
Meaning	SMCEDT is attempting to FREAD the RUN statement.
Action	Supply the RUN statement. Operator replies using one of the three one-line options: D for sort (sort or sort and merge), M for merge-only, C for copy.

```
KEYS = <— prompting level 1 version
. . .<L/S/F>, <A/D>, <KEYCOL>, . . .        prompting
. . .<C>, <A/D>, <KEYCOL>, <KEYCOLS>, . . . level 2
AWAITING REPLY                              version
```

Phase	SMCEDT
Type	Action
Meaning	SMCEDT is attempting to FREAD the keys statement.
Action	Supply the KEYS statement: L is used for logical binary, S is used for signed binary, and F is used for floating point key. C is used for a character key.

```
INFILE <n> = <— prompting level 1 version
D, <FILNUM>, <RECLTH>, <BLKSIZ>,
 <SKIPCNT>, <DOCNT>, <CR>
T, <LUN>, <RECLTH>, <BLKSIZ>, <SKIPCNT>,    prompting
 <DOCNT>, <CR>                              level 2
P, <A/B>, <LUN>, <RECLTH>, <BLKSIZ>,        version
 <SKIPCNT>, <DOCNT>, <CR>
AWAITING REPLY
```

Phase	SMCEDT
Type	Action
Meaning	SMCEDT is attempting to FREAD the INFILE statement for the nth input file.
Action	Supply the INFILE statement: D is used for disk input, T is used for magnetic tape binary (read or pseudo), P is used for paper tape (ASCII type input).

```
OUTFILE = <— prompting level 1 version
D, <FILNUM>, <LUN>, <BLKSIZ>, <CR>          prompting
T, <LUN>, <BLKSIZ>, <CR>                    level 2
P, <A/B>, <LUN>, <BLKSIZ>, <CR>             version
AWAITING REPLY
```

Phase	SMCEDT
Type	Action
Meaning	SMCEDT is attempting to FREAD the OUTFILE statement.
Action	Supply the OUTFILE statement: D is used for disk, T is used for magnetic tape (read or pseudo binary input) and P is used for ASCII (paper tape) input.

EXPECTED <parameter name>
FOUND <first character of erroneous reply>

Phase SMCEDT

Type Fatal or action

Meaning This is a class of messages used to pinpoint source language errors.

SMCEDT names the expected parameter, and displays the first character of the character string having the location of the expected parameter but violating the rules for that parameter.

This is a fatal error when the prompting level is zero. Otherwise, SMCEDT is attempting to FREAD a new value of the expected parameter, followed by the rest of the statement concerned.

Following is a list of expectations (parameter names) as used in the above class of messages. The statement types concerned are listed next to each parameter name.

The user should determine which statement type is concerned, and should compare his version of that statement with the requirements of that statement, especially concentrating on the expected parameter.

Expectation	Statements
,	All
A/B	INFILE, OUTFILE
A/D	KEYS
D/M/C	RUN

Expectation	Statements
D/T/P	INFILE, OUTFILE
L/S/F/C	KEYS
S/N	RUN
blksiz	INFILE, OUTFILE
docnt	INFILE
filcnt	RUN
filnum	INFILE, OUTFILE
keycnt	RUN
keycol	KEYS
keycols	KEYS
lun	INFILE, OUTFILE
	Prompting level
reclth	INFILE
skipcnt	INFILE
wkbksz	RUN

Action Supply the unaccepted part of the statement concerned, including the expected parameter.

G = <n>

Phase SMCEDT

Type Informative

Meaning For a sort-only run, SMCEDT has format read (FREAD) and digested all source statements, has done memory calculations, and has indicated that the tournament can hold n logical records. Therefore, n or more logical records require use of disk work space and merging, while fewer than n logical records may be sorted entirely in memory.

IWAY = <n>

Phase SMCEDT

Type Informative

Meaning For a sort-only run, SMCEDT has format read (FREAD) and digested all source statements, has done memory calculations, and has indicated that SMCIMG can afford n+1 buffers of size WKBKSZ. Therefore, SMCIMG has a maximum way-of-merge of n, using n buffers for input and one for output.

FWAY = <n>

Phase SMCEDT

Type Informative

Meaning For a sort-only run, SMCEDT has format read (FREAD) and digested all source statements, has done memory calculations, and has indicated that SMCFMG can afford n input buffers of WKBKSZ size. Therefore, SMCIMG has a maximum way-of-merge of n.

INITIAL SORTING (PHASE 4)

COPY BEGINS

Phase SMCSRT

Type Informative

Meaning SMCSRT is now initializing itself to perform a copy run.

INTERNAL SORT BEGINS

Phase SMCSRT

Type Informative

Meaning SMCSRT is loaded and is now initializing itself to perform the internal sort of a sort run.

READY FILE = <n>
TYPE GO/QT/BY

Phase SMCSRT

Type Action

Meaning The user is asked to ready the input file specified by the nth INFILE statement. The user has the options of supplying or deleting that file.

Action Type GO (go ahead and process the file).

Type QT (quit considering; i.e., delete, this file).

Type BY (bypass the operator for this situation from now on, and assume GO).

FREAD STATUS = <hhhh>
LUN = <k>
FILNUM = <f>
<hexadecimal dump of buffer>
TYPE GO/QT/BY

Phase SMCSRT, SMCFMG

Type Action

Meaning For the designated user input file, the last FREAD returned bad hardware status = hhhh.

The buffer used for that FREAD is dumped in hexadecimal.

The user has the option of continuing with or deleting the file.

If the current file is continued, the current block is deleted and the FREAD is repeated.

Action	Type GO (go on using this file, but delete the current block and repeat the FREAD).
	Type QT (quit using; i.e., delete, this file).
	Type BY (bypass the operator for this situation from now on, and assume GO).

```
BLKSIZ/RECLTH ≠ 1,2,3. . .
LUN = <k>
FILNUM = <f>
<hexadecimal dump of buffer>
TYPE GO/QT/BY
```

Phase	SMCSRT, SMCIMG, SMCFMG
Type	Action
Meaning	For the designated file, the size of the last format read (FREAD) block is not a nonzero multiple of the specified <RECLTH>.
	The buffer used for that FREAD is dumped in hexadecimal.
	The user has the options of continuing with or deleting the file. If the current file is continued, the current block is deleted and the FREAD is repeated.
Action	Type GO (go on using this file, but delete the current block and repeat the (FREAD).
	Type QT (quit using; i.e., delete, this file).
	Type BY (bypass the operator for this situation from now on, and assume GO).

```
OVERSIZE BLOCK
LUN = <k>
FILNUM = <f>
<hexadecimal dump of buffer>
TYPE GO/QT/BY
```

Phase	SMCSRT, SMCFMG
Type	Action
Meaning	For the designated file, the size of the last format read (FREAD) block exceeds the blksiz specified for that file.
	The buffer used for that read is dumped in hexadecimal.
	The user has the options of continuing with or deleting the file.
	If the current file is continued, the current block is deleted and the FREAD is repeated.
Action	Type GO (go on using this file, but delete the current block and repeat the FREAD).
	Type QT (quit using; i.e., delete, this file).
	Type BY (bypass the operator for this situation from now on, and assume GO).

```
FWRITE STATUS = <hhhh>
LUN = <k>
FILNUM = <f>
<hexadecimal dump of buffer>
TYPE GO/QT/BY
```

Phase	SMCSRT, SMCFMG
Type	Action
Meaning	For the designated user input file, the last FWRITE returned bad status = hhhh.
	The buffer used for that FWRITE is dumped in hexadecimal.
	The user has the options of retrying the FWRITE or terminating the run.
Action	Type GO (go on with the run; retry the STOSEQ).
	Type QT (quit the run).
	Type BY (bypass the operator for this situation from now on, and assume GO).

```
RTVSEQ REQIND = <hhhh>
LUN = <k>
FILNUM = <f>
<hexadecimal dump of buffer>
TYPE GO/QT/BY
```

Phase	SMCSRT, SMCIMG, SMCFMG
Type	Action
Meaning	For the designated file, the last RTVSEQ incurred bad status = hhhh.
	The buffer used for that RTVSEQ is dumped in hexadecimal.
	The user has the option of continuing with or deleting the file. If the current file is continued, the current block is deleted and the RTVSEQ is repeated.
Action	Type GO (go on using this file, but delete the current block and repeat the RTVSEQ).
	Type QT (quit using; i.e., delete, this file).
	Type BY (bypass the operator for this situation from now on, and assume GO).

```
STOSEQ REQIND = <hhhh>
LUN = <k>
FILNUM = <f>
<hexadecimal dump of buffer>
TYPE GO/QT/BY
```

Phase	SMCSRT, SMCIMG, SMCFMG
Type	Action
Meaning	For the designated file, the last STOSEQ incurred bad status = hhhh.
	The buffer used for that STOSEQ is dumped in hexadecimal.
	The user has the option of retrying the STOSEQ or terminating the run.
Action	Type GO (go on with the run; retry the STOSEQ).
	Type QT (quit the run).
	Type BY (bypass the operator for this situation from now on, and assume GO)!

LUN = <k>
FILNUM = <f>

Phase SMCSRT, SMCIMG, SMCFMG

Type Informative

Meaning The file on logical unit k with filnum f is the subject of the succeeding message(s).

If f does not equal 0, the file is a file manager file; otherwise it is not.

PASSED = <n>

Phase SMCSRT, SMCIMG, SMCFMG

Type Informative

Meaning This message can appear for both output files and input files.

For the file designated, n is the number of logical records blocked or deblocked. This includes not only the number of logical records docnt that were deblocked and then processed (done), but also the number of logical records skipcnt that were deblocked and then discarded (skipped).

If an input block is discarded due to an I/O error, its logical records are never deblocked and thus do not relate to this message.

When there are no I/O errors, and the input file is long enough, then n = skipcnt + docnt.

If the input file is short enough, then n can be less than skipcnt and less than docnt.

SEQUENCES = <n>

Phase SMCSRT

Type Informative

Meaning For a sort run, SMCSRT has performed the internal sort and the logical records done are grouped into n sequences.

The average string length may be computed from n and the logical record count for the phase. This average is 2*G for a random file, and lower or higher, respectively, as the inherent order of user input decreases or increases.

For a given file, the multiplier of G (e.g., 2 above) tends to be a constant associated with the degree of inherent order and independent of G, logical record count, or sequence count.

This constant is referred to as ORDER in the timing equations in appendix G. For example, ORDER equals 1, 2, infinity for reverse, random, and perfect input order, respectively.

RECORDS IN = <m>
RECORDS OUT = <n>

Phase SMCSRT, SMCIMG, SMCFMG

Type Informative

Meaning For the phase concerned, m is the number of logical records sent to the processing logic from the deblocking logic, while n is the number of logical records sent to the blocking logic by the processing logic.

Aside from hardware failure, m is equal to n except when the user elects to delete logical records due to sequence errors (a special count is published for such deletions).

DONE = <n>

Phase SMCSRT, SMCFMG

Type Informative

Meaning This message appears only for input files with a skipcnt not equal to 0.

For the designated file, n is the number of logical records docnt that were done (deblocked and then processed), rather than the number skipcnt skipped (deblocked but then discarded), and rather than lost due to discarding blocks that experienced I/O errors.

DEFFIL REQIND = <hhhh>
LUN = <k>
FILNUM = <f>

Phase SMCSRT, SMCIMG, SMCFMG

Type Fatal

Meaning For the designated user-specified output file, DEFFIL returned bad status = hhhh.

The run terminates for lack of a final output file.

The DEFFIL was tried because the file was not defined before the Sort/Merge run.

SEGMENT-LIST ERROR

Phase SMCSRT, SMCIMG, SMCFMG

Type Fatal

Meaning A sort run terminates due to an error on the segment list, which is used to keep track of workfile extensions.

SEQ. DIR. ERROR

Phase SMCSRT, SMCIMG, SMCFMG

Type Fatal

Meaning A sort run terminates because an error occurred while reading or writing the sequence directory.

If no mass storage error message appeared just before this message, then this was an undetected disk hardware error (part of Sort/Merge's error detection is context-dependent).

TOO LITTLE DISK

Phase SMCSRT, SMCIMG

Type Fatal

Meaning A sort run must terminate because a needed work file cannot be defined due to inadequate available disk space.

Perhaps the run could be retried and succeed after some file manager files were released.

Otherwise, if the amount of data to be sorted is still excessive, the run has to be segmented into several smaller sort and merge-only runs.

INTERMEDIATE MERGING (PHASE 5)

INTERMEDIATE MERGE BEGINS

Phase SMCIMG

Type Informative

Meaning SMCIMG is loaded and is now initializing itself to perform the intermediate merging of a sort run.

1STWAY = <n>

Phase SMCIMG

Type Informative

Meaning SMCIMG is initializing itself to perform the intermediate merging of a sort run.

SMCIMG has determined the optimum merge strategy for IWAY, FWAY, and the number of strings produced by the internal sort.

The first way-of-merge to be used by SMCIMG equals n. Any subsequent way-of-merge equals IWAY for SMCIMG, and equals FWAY for SMCFMG.

U = <n>

Phase SMCIMG

Type Informative

Meaning SMCIMG is initializing itself to perform the intermediate merging of a sort run. SMCIMG has determined that n is the unit strings rating of the optimum merge strategy for IWAY, FWAY, and the number of strings produced by the internal sort.

SEQUENCE ERROR
<hexadecimal dump of new logical record to be output>
PRECEDES
<hexadecimal dump of last logical record output>
TYPE GO/QT/BY

Phase SMCIMG, SMCFMG

Type Action

Meaning Sequence checking of merge output was selected by the user and has just detected a sequence error; the new logical record to be output precedes, with respect to key values, the last logical record output.

The user has the options of retaining or deleting the new logical record. However, the last logical record output may actually be the defective logical record, or both logical records might be defective.

Both logical records are dumped in hexadecimal format for user inspection.

Deleting the new logical record deletes one sequence error for each future input of the current output. However, the wrong logical record may have been deleted.

A merge-only run could be used to update the final output with replacements for deletions, whether those deletions were of good or bad logical records.

The logical record deletion count is output for each merge phase (SMCIMG and SMCFMG).

Action Type GO (go on using; i.e., retain, the new logical record).

Type QT (quit using, i.e., delete, the new logical record).

Type BY (bypass the operator for this situation from now on, and assume GO).

RELFIL REQIND = <hhhh>
LUN = <k>
FILNUM = <f>
TYPE GO/QT/BY

Phase SMCIMG, SMCFMG

Type Action

Meaning For the designated file, the RELFIL incurred bad status = hhhh.

The user has the options of retrying or skipping that RELFIL.

Action Type GO (go on without this particular RELFIL).

Type QT (retry this particular RELFIL).

Type BY (bypass the operator for this situation from now on, and assume GO).

DELETES = <n>

Phase SMCIMG, SMCFMG

Type Informative

Meaning The current phase is ending, and n is the number of logical records the user elected to delete because of sequence errors.

FINAL MERGING (PHASE 6)

FINAL-MERGE BEGINS

Phase	SMCFMG
Type	Informative
Meaning	SMCFMG is now initializing itself to perform the final merging of a sort run.

MERGE-ONLY BEGINS

Phase	SMCFMG
Type	Informative
Meaning	SMCFMG is loaded and is now initializing itself to perform a merge-only run.

<n> = 12

Phase	SMCFMG
Type	Fatal
Meaning	Fixed tables contain incorrect SMCFMG size.

INTERPHASE RECORD COUNTS DISAGREE

Phase	SMCFMG
Type	Informative
Meaning	The number of logical records output by the final merge disagrees with the number input to the internal sort. Perhaps messages have already appeared indicating that blocks were discarded due to I/O errors, or that logical records were discarded due to sequence errors. By means of messages such as that cited above, the user should be able to explain record count messages for each phase, and the interphase disagreement in particular. Otherwise, it should be assumed that undetected hardware errors are causing the problem.

OUTPUT

The output is always a single file, even for a copy run. Restrictions on file mode for files to be used by the file manager were mentioned in section 2. More specific restrictions for those files are given in appendix J.

The OUTFILE statement specifies the medium (and, where applicable, the logical unit). Whether the operator lists the output together with the RUN statements depends on whether or not he specifies the comment and output devices to be the same unit.

If the operator chooses an output to mass storage or to magnetic tape, he can always retrieve the file for viewing later since he specified the output file identification filnum and can use that to dump the file on a readable medium.

The content of each record on the output file should be identical to that of the comparable records on the input file with two exceptions:

- If the operator skipped or deleted records, these will not appear in the output file.
- If the system notified the operator of an input/output error, but the operator chose to include the record(s) anyway, there may be copying errors in the records. These errors can be removed manually by the operator at some later time, although this is a time-consuming task. If computing errors are suspected, it is more efficient to rerun the entire Sort/Merge procedure.

GLOSSARY

<>	Brackets enclose user-specified parameters; e.g., <item>
	Number rounded down to nearest integer; e.g., 5.3 = 5
	Number rounded up to the next integer; e.g., 5.3 = 6
1STWAY	SMCIMG's method of merging the first group of strings
A	1. CPU register A 2. Ascending order (key sorting) 3. ASCII mode
A/B	ASCII/binary parameter of a file statement
A/D	Ascending/descending parameter in KEYS statement
((. . .(address). . .))	Contents of contents of. . . contents of the address
(address)	Contents of the address. The address may be omitted when the context clearly indicates which address is referenced (usually when sixteen feet of contents are specified).
(address) entry	(address) is in entry point to the logic being discussed.
(address) exit	(address) when exiting from the logic being discussed
(address) m	Bit m of (address)
(address) m-n	Bits m through n of (address)
(address+0, +1)	(address), (address+1)
ASCII	American Standard Code for Information Interchange; i.e., a standard correspondence between graphic symbols and bit patterns
*	Prefix of batch control statements in MSOS
**	Exponential
*JOB	Control statement that initiates a job
*K,Ia,Pb,Lc	Control statement that selects logical units to be used for standard input (I), standard binary output (P), and standard listing (L)
Bin	A storage area used for the tournament comparisons. Initially holds records, later holds record pointers
blksiz	Parameter for INFILE and OUTFILE commands
BY	Operator reply to certain ready file or error messages; causes a GO condition for this and similar future errors (bypass operator interaction)
CDT	Conversational display terminal (CRT and keyboard)
CFO	Comments from operator logical unit of MSOS
Copy run	Logical records from one or more user input files are copied onto a single output file.
CPC	Computer program component
cr	Carriage return
CTO	Comments to operator logical unit of MSOS
D	1. RUN statement: disk sorting selection 2. KEYS statement: use descending order 3. INFILE or OUTFILE statement: use disk medium
Deblock	Use the input record; rewrite it in new files.
Dispatcher	The MSOS routine that selects and activates the next program to run
docnt	INFILE command parameter - number of records to process during the run
$	Hexadecimal; e.g., $1A = 26
E	Value of DOCNT: Do every record until end-of-file.
EOF	End-of-file
F	KEYS command parameter: key is in floating point format
filcnt	RUN command parameter: number of files to be processed
File	Data storage, composed of one or more records
File record	A record in a file managed by Sort/Merge or the file manager. Records are stored in file record blocks.
File table	A table accounting for files

File table labels	Labels in the file table
filnum	The unique identifier for a file manager file. Used in the INFILE and OUTFILE commands
FIS	File information segment - a file manager table that contains file indexing, identification, type addressing, and keying information for each file defined by a filnum
FIS block	The set of FIS clocks that contain all the non-key indexing information for the file management system
FRB	File record block - an area of mass storage controlled by the file manager that contains records managed by the file management system. The set of all FRBs contains all the records controlled by the file management system.
FREAD	Formatted read statement of MSOS
FSLIST	The fixed tables of Sort/Merge
FWA	First word address
fway	Maximum number of strings being combined by the current merging operation
FWRITE	Formatted write statement of MSOS
G	Number of real bins used by tournament (group size)
GO	A legal operator reply to an action message: the program will process the record
GTFILE	An MSOS request to read into core a file on the program library
I	A CPU register used for indexing
INFILE	Input command containing input file parameters
INIT	Several separate routines that respectively initiate SMCMON, SMCSRT, SMCIMG, and SMCFMG
iway	Maximum number of strings being merged
Job processor	The background monitor for MSOS
Key	Label or index in the record by which records can be sorted
Key table	File manager table containing key information
keycnt	RUN command parameter: number of keys to be used for sorting by this run
keycol keycols	KEYS command parameters: keycol specifies the column in which the first character of the key is found; keycols specifies the number of columns used for the key if it is a character type key
KEYS	Input command that defines the key or keys to be used during sorting/merging runs
L	KEYS command parameter: logical binary type key
LIBEDT	Library editor program
L/S/F	KEYS command parameter in prompting statement. Specifies by type: L = logical binary, S = signed binary, F = floating point
lun	Logical unit
LWA	Last word address
M	RUN command parameter: merge-only run
Merge-only	Sort/Merge run where sorting is done prior to start of run and sorted files are merged
Merge order	Number of buffers used during current merging phase (intermediate or final)
Messages	Statements sent to operator: may be requests for action or records by program performance
MONI	Monitor call
MSOS	Mass Storage Operating System
OUTFILE	Input command containing output file parameters
P	1. Program index register of CPU 2. INFILE or OUTFILE command parameter: use device with paper type tape input; i.e., ASCII or binary 3. Prompting level parameter: 0 = none, 1 = some, 2 = full
Passed	Record has been processed
P+n exit	Exit to the nth word following the calling statement
Phase	Execution portion of Sort/Merge. There are six phases, executed in the given order: SMC, SMCMON, SMCEDT, SMCSRT, SMCIMG, and SMCFMG.
Prefix	Many Sort/Merge messages contain two parts - an initial alphabetic message describing the contents and a second part (suffix) that is a decimal or hexadecimal representation of data

Prompting	Operator interaction level for Sort/Merge: 0 = no prompting, 1 = some prompting, 2 = maximum prompting
Pseudo tape	Storage area formatted as if it were a magnetic tape storage medium
Q	A CPU register
QT	Legal operator reply to an action message: delete (quit using) this file
reclth	INFILE command parameter: record length
RECPTR	Two word disk address of file record
RELOC4	Relocate table containing the absolutizing value added to relative statements in a relocatable program to absolutize the program.
REQBUF	Work area in core used by the file manager when processing requests
REQIND	One word status buffer used by file manager
RSA	Record storage area: an array of bins used by tournament. Initially holds records: later holds pointers to records.
RTJ	Return jump command
RUN	Input command containing parameters that define type of run desired
Runtime	Time when sorting, merging, or copying is being performed
S	1. Number of strings resulting from an intermediate sorting operation 2. KEYS command parameter: signed binary mode
Scatter code	A hashing code used to generate indexing for files designed by operator selected filnum.
Segment	A disk file of entries; segments of one filnum file may reside (in segments) on two or more disks
Segment file	A file that exists in segments on two or more disks
Segment list	A list that links file segments
Sequence	A group of logic records ordered by some key or keys value(s)
Sequence directory	A disk work file: except for the final output file, it lists logical unit number, filnum, and some flags for each output string generated by a sort run
skipcnt	INFILE command parameter: the number of records to skip in the current file before processing a record of interest
SMC	The loader/initializer phase of Sort/Merge
SMCEDT	The editing and run definition phase of Sort/Merge: it checks the operator input parameters.
SMCFMG	The final merging and output phase of Sort/Merge
SMCIMG	The intermediate merging phase of Sort/Merge. It is used when not all records can be sorted in core at one time.
SMCMON	The monitor phase of Sort/Merge
SMCSRT	The sorting phase of Sort/Merge
S/N	RUN command parameter: selects or rejects sequencing check
Sort	Sorting run: merging may be requested also if not all records can be kept in core at one time to perform a single tournament sorting operation
Source language	1. Language from which program assembly language is generated 2. Sort/Merge: the five commands (P, RUN, KEYS, INFILE, OUTFILE) that define the run parameters
String	Sequence of records
Suffix	Second part of certain Sort/Merge messages: see Prefix
Tournament	A comparison routine that selects a winner from a group of contestants; e.g., sorts two records by key words, one letter at a time. See appendices D, E, F, and I.
TSA	Tag storage area: a table of G numbers ranked according to the tournament results. Tags identify records and point to entries in sequence array.
TTY	Teletypewriter
U	The unit strings rating of SMCIMG: the number of times all data from an original string from SMCSRT is processed by SMCIMG
Unit strings	The original strings produced by SMCSRT
Variable tables	Sort/Merge tables holding INFILE (file table), and KEYS (key table) information for the run

Way-of-merge	Number of strings being combined by the current merging operation
WIERD	Outputs special error messages
wkbksz	RUN command parameter: specifies size of data portion of work file buffer
Work file	Intermediate files used to process records and record pointers by Sort/Merge
Y labels	Labels in the fixed tables. All labels begin with a Y.
Z	Escape character. When used in place of a comma or field in RUN, KEYS, INFILE, or OUTFILE statements, this cancels the operator's current input and returns him to the start of the prompting (if any) for that statement.

SUBROUTINE HIERARCHY

The following is a list of callers and the procedures they call their callees, by phase. Within a phase, the arrangement breaks down some of the higher level routines. A brief description of subroutine functions is given in the Procedure Names section. This description of internal structure is provided to assist the analyst in understanding the internal operation of the Sort/Merge utility package. Specific parameters are subject to change as a result of program modification by CDC.

SMC PHASE

SMC – LOAD (GTFILE is used by LOAD)

LOAD – SMCMON/SMCEDT/SMCSRT/SMCIMG/SMCFMG

SMCMON PHASE

SMCMON – DISP/INIT/LOAD/REL/TYPOUT

INIT – BOMB/MONI/RELOC/SYFMLU/TYPOUT

REL – DULU/GORQT/HADOUT/LUFNO/RELFIL/RESAQI/SAVAQI/WIERD

TYPOUT – DISP/MONI/RESAQI/SAVAQI

BOMB – DISP/HADOUT/TYPOUT

SYFMLU – GFMLU

DULU – WIERD

GORQT – ACCEPT/HADOUT

HADOUT – RESAQI/SAVAQI/TPHEX/TYPOUT

LUFNO – TPDEC

WIERD – BOMB/TPDEC

ACCEPT – TYPIN/TYPOUT

TPHEX – HXBCD/MOVE/TYPOUT

TPDEC – TPHEX

TYPIN – BOMB/DISP/MONI/RESAQI/SAVAQI/TYPOUT

HXBCD – BINDEC/BINHEX

BINDEC – RESAQI/SAVAQI/WIERD

BINHEX – B2HXBT

EOS – CLSU/PUTSEQ

CLSU – TYRCT/WRTD/WRTT

PUTSEQ – BOMB/RESAQI/SAVAQI/WRTD

TYRCT – BIGADD/BIGSUB/LUFNO/TYRCTY

WRTD – BADBLK/BOMB/CLRBIO/DEF/DULU/PUTSEG/STOSEQ/WIERD/WRTDIN

WRTT – BADBLK/BOMB/DISP/MONI/STATUS

TYRGCTY – BTDEC

BADBLK – GORQT/HADOUT/HEXDMP/LUFNO

CLRBIO – CLR

DEF – BOMB/DEFFIL/LUFNO/RANDOM/RANNIT/TPHEX/WIERD

PUTSEG – CLRBIO/CLRFT/DEF/PTSGWT

STATUS – MONI

BTDEC – TPDEC

HEXDMP – BINHEX/TYPOUT

CLRFT – CLRBIO

PTSGWT – WRTD/WRTDIN

GETSEQ – BOMB/RDD/RESAQI/SAVAQI

RDD – BADBLK/GETSEG/RDDNIT/RTVSEQ/WIERD

GETSEG – BOMB/CLRBIO/CLRFT/RDD/RDDNIT/REL

GETU – BADBLK/BIGCNT/RDD/RDT/TYRCT

RDT – BADBLK/DISP/MONI/STATUS

PUTU – BIGCNT/MOVE/WRTD/WRTT

BOS – CLRFT/DEF

BIGB2D – BINDEC

SMCEDT PHASE

SMCEDT – BIGNUM/BOMB/CKYSIZ/INFILE/KEYS/LINK/MEM/NEWSCL/OUTFIL/RELOC/RUN/TPDEC/TYPOUT/XCK

BIGNUM – SCDIAG/TOKEN

INFILE – ALPHA/BIGADD/BINDEC/COMALF/COMBIG/COMMA/COMPOS/NEWSCL/PROMPT/SCDIAG/TOKEN

KEYS – ALPHA/COMALF/COMMA/COMPOS/KEYFWA/KRANGE/NEWSCL/PROMPT

LINK – WIERD

MEM – DETG

OUTFIL – ALPHA/COMALF/COMPOS/NEWSCL/PROMPT

RUN – ALPHA/COMALF/COMPOS/NEWSCL/PROMPT

SCDIAG – BLANK/BOMB/MOVE/NEWSCL/RESAQI/SAVAQI/TYPOUT

TOKEN – DIGTST/MOVE/SCLBYT

ALPHA – SCDIAG/TOKEN

COMALF – ALPHA/COMMA

COMBIG – BIGNUM/COMMA

COMMA – SCDIAG/TOKEN

COMPOS – COMMA/POSNUM

PROMPT – TYPOUT

SCLBYT – RESAQI/SAVAQI/TYPIN

POSNUM – BIGNUM

SMCSRT PHASE

SMCSRT – BTDEC/CLSU/GET/IEOR/INIT/PUT/PUTU/TOURN/TPDEC

GET – GMPKEY/IEOR/MOVE

IEOR – CLRFT/GETU/GORQT/MOVE/TPDEC

INIT – CLRFT/LINK/RELOC/TURNIT/TYPOUT

PUT – BOS/EOS/PUTU

TOURN – CMPKEY

LINK – WIERD

SMCIMG PHASE

SMCIMG – BTDEC/GET/INIT/MGINIT/MTOURN/PUT/TPDEC

GET – GETU

INIT – DETM/LINK/RELOC/TPDEC/TYPOUT

MGINIT – BOS/FTINIT/TURNIT

MTOURN – CMPKEY

PUT – CMPKEY/EOS/GORQT/HEXDMP/PUTU/TYPOUT

DETM – WIERD

LINK – WIERD

FTINIT – CLRFT/GETSEQ

SMGFMG PHASE

SMCFMG – BTDEC/GET/INIT/MTOURN/PUT/TPDEC/TYPOUT

GET – GETU

INIT – BUFALO/CLRFT/GETSEQ/GORQT/LINK/MOVE/RELOC/TPDEC/TURNIT/TYPOUT/WIERD

MTOURN – CMPKEY

PUT – CLSU/CMPKEY GORQT/HEXDMP/PUTU/TYPOUT

LINK – WIERD

PROCEDURE NAMES

ACCEPT	Sends messages to operator and accepts one-word replies
ALPHA	Decodes alphabetic characters for operator input messages
B2HXBT	Converts hexadecimal 10 through 15 to A through F
BADBLK	Informs operator that program found a bad data block; requests operator instructions
BIGADD	Handles output sums greater than 10,000
BIGB2D	Converts 32 bits of binary number to BCD format
BIGCNT	Updates two-part counter with breakpoint at 10,000
BIGNUM	Decodes eight-digit operator input for SMCEDT
BIGSUB	Handles output differences requiring special borrows
BINDEC	Converts 16 bits to binary number to BCD format
BINHEX	Converts binary number to hexadecimal format
BLANK	Blanks out the correctly entered parts of the request in the expected/received error message that is sent to the operator
BOMB	Outputs the fatal error message when Sort/Merge aborts
BOS	Begins sequence processing by defining a filnum on the logical unit, initializing file table and starting sequence directory processing
BTDEC	Outputs prefix with two-word BCD number
BUFALO	Allocates output buffer
CKYSIZ	Checks for sufficient core size to perform requested Sort/Merge

CLR	Clears the region of core specified by the input request
CLRBIO	Clears selected I/O buffer areas and parameters
CLRFT	Clears file table
CLSU	Closes and writes the current block
CMPxxx	Subroutines used to execute the tournament sorting process
COMALF	Decodes comma plus alphabetic character from operator's input message
COMBIG	Decodes eight-digit number and comma for SMCEDT
COMMA	Decodes commas in operator input messages
COMPOS	Computes address of error message
DEF	Prepares DEFFIL requests
DEFFIL	File manager request to define a file
DEFGLU	Processes errors for DEFFIL requests
DETG	Determines G (group size) for the tournament
DETM	Determines order of I-merges (final merging)
DIGTST	Tests if operator input is in decimal format
DISP	Dispatcher
DULU	Checks status (up or down) of logical unit
EOF	End-of-file processor
EOS	Ends sequence processing by closing the current block and completing the sequence directory
FTINIT	Initializes merge input file tables
GET	Gets records and moves them for tournament input for sorting
GETSEG	Get next segment (file)
GETSEQ	Calculates sequence logical unit and filnum
GETU	Gets a new buffer
GFMLU	Finds logical unit to be formatted by SYFMLU
GORQT	Decodes operator reply of GO (continue), QT (delete file), or BY (continue bypassing this type of error)
HADOUT	Generates $ prefix for hexadecimal output and outputs special messages
HEXDMP	Dumps buffer in $ format on output device
HXBCD	Convert hexadecimal to BCD
IEOR	Finds files and requests operator to ready them
INFILE	Decodes INFILE message inputs to SMCEDT
KEYFWA	Computes and stacks relative first word address
KEYS	Decodes KEYS message inputs to SMCEDT
KRANGE	Generates key word tables
LINK	Links program entry points to SMCMON
LOAD	Loads SMC or SMCMON
LUFNO	Identifies the file
MEM	Algorithm that computes core required for the run
MGINIT	Initializes intermediate merge operations
MOVE	Moves words in core
MTOURN	Tournament algorithm
NEWSCL	Points to new operator-entered field in input control messages to SMCEDT
OUTFIL	Decodes OUTFILE message inputs to SMCEDT
POSNUM	Sets up positive numbers less than or equal to 32K
PROMPT	Determines prompting level selected by operator and outputs messages appropriate to that level
PTSGWT	Writes segment list
PUT	Updates tables using tournament intermediate results (used by SMCSRT)
PUTSEG	Updates segment list
PUTSEQ	Builds sequence directory entries
PUTU	Puts current record in buffer if there is room or writes buffer and gets a new buffer for current record
RANDOM	Continues generating random numbers
RANNIT	Initializes random number sequence
RDD	Checks status of data blocks
RDDNIT	Sets up retrieval sequence call
RDT	Executes and checks FREAD operation
REL	Prepares to release file normally or abnormally
RELFIL	Release the filnum
RELOC	Relocates programs and tables during loading

RESAQI	Restores the A, Q, and I registers
RTVSEQ	Retrieves sequence information
RUN	Decodes RUN message inputs to SMCEDT
SAVAQI	Saves A, Q, and I register contents
SCDIAG	Returns expected/received operator selected input to operator and requests correct reentry of parameters. Used by SMCEDT
SCLBYT	Maintains byte counters while decoding operator inputs to SMCEDT
STATUS	Sets up status word
STOSEQ	Stores sequence
SYFMLU	Formats logical units in tables during initialization
TOKEN	Decodes format of operator input message to SMCEDT
TOURN	Tournament algorithm
TPDEC	Outputs prefix with BCD number
TPHEX	Outputs full hexadecimal number
TOURNIT	Updates TSA to initialize tournament
TYPIN	Reads formatted input from operator, checks it, and requests input if errors occur
TYPOUT	Sends formatted output message to operator
TYRCT	Calculates number of records passed
TYRCTY	Displays number of records passed
WIERD	Outputs special error messages
WRTD	Writes blocks for STOSEQ and checks write status
WRTT	Writes formatted data and check status of the operation
XCK	Cross checks operator input message parameters to SMCEDT

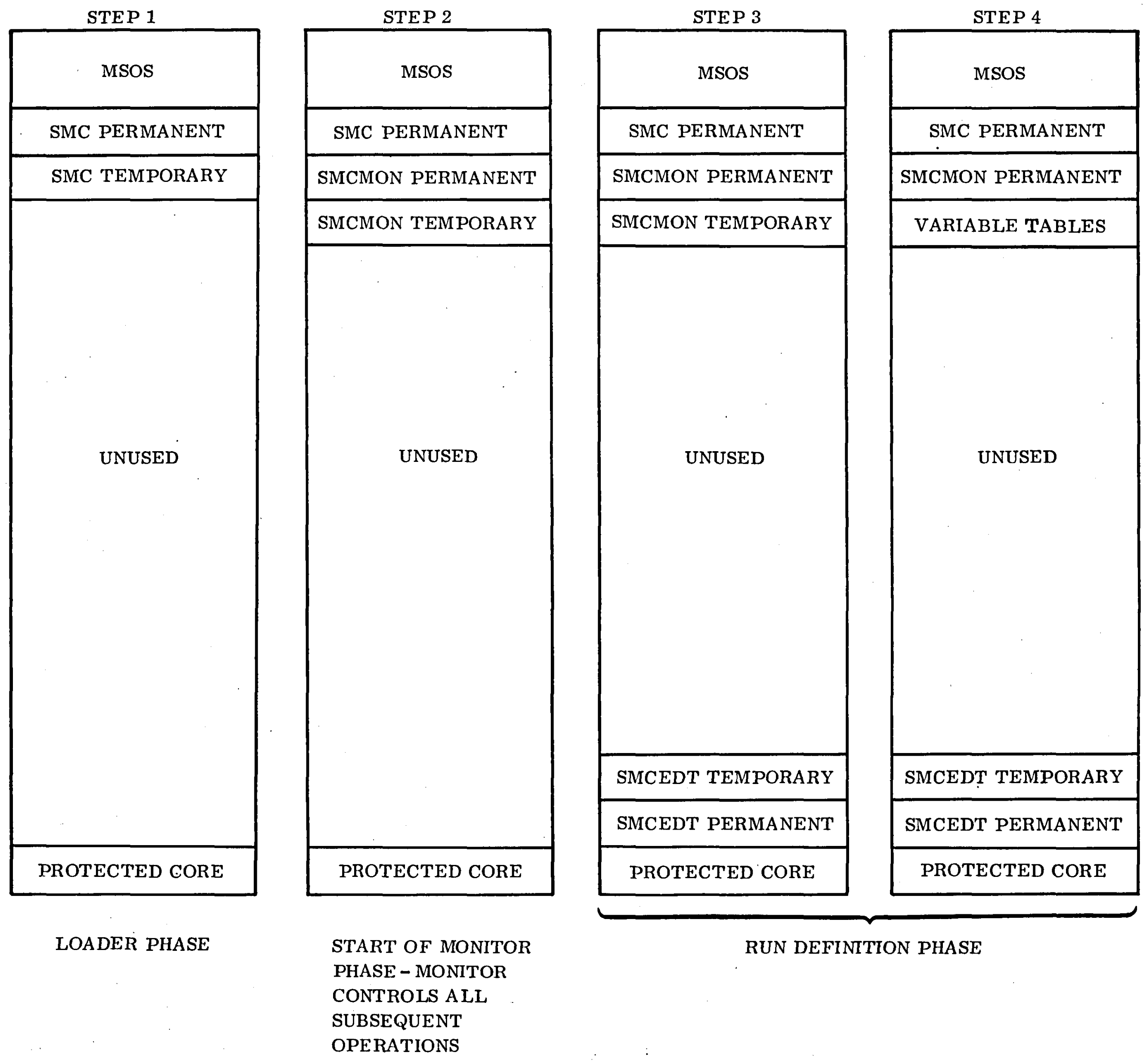
SORT/MERGE CORE USAGE

The memory layouts in figure C-1 are for a sort run, which involves internal sorting, intermediate merging, and the final merging operation.

Step	Action
1	*SMC brings SMC into core.
2	SMC has initialized itself and has loaded SMCMON over the SMC initialization logic.
3	SMCMON has initialized itself and has loaded SMCEDT high in core, so that tables may be built in the region between SMCMON temporary and SMCEDT temporary.
4	Variable tables have been built over SMCMON temporary (and perhaps some other core), but in this case the tables were not extensive enough to overlay the SMCEDT temporary region. This is the typical situation.
5	SMCSRT has been loaded.
6	SMCSRT has initialized itself and has partitioned remaining unprotected core. SMCSRT has occupied the space formerly used by SMCSRT temporary, SMCEDT temporary, and SMCEDT permanent.
7	SMCIMG has been loaded, replacing SMCSRT, which completed its processing. Communication between SMCSRT and SMCIMG is accomplished using the fixed tables (not shown) located at the top of SMCMON permanent.
8	SMCIMG has initialized itself and has partitioned remaining unprotected core. SMCIMG temporary, which is no longer needed, has been overlayed by buffers, etc.
9	SMCFMG has been loaded, replacing SMCIMG. Communication between SMCIMG and SMCFMG is accomplished by the fixed tables mentioned in step 7.
10	SMCFMG has initialized itself and has partitioned remaining unprotected core. SMCFMG temporary, which is no longer needed, has been overlayed by buffers, etc.

A sort-only run might involve only steps 1 through 6, and 9 and 10, with SMCFMG replacing SMCSRT in step 9. Also, a sort-only run might involve only steps 1 through 6. A copy run would use steps 1 through 6; however, step 6 would not use the core space between the input buffer and protected core.

A merge-only run would use only steps 1 through 4, and 9 and 10. In step 9, SMCFMG would be loaded without replacing a previous phase. In step 10, if there were a runtime deletion of input files, there would be some unused core at the top of the input buffers and the input file tables. There would also be unused core at the top of unprotected core.



0335

Figure C-1. Core Maps during Sort/Merge Processing (Sheet 1 of 3)

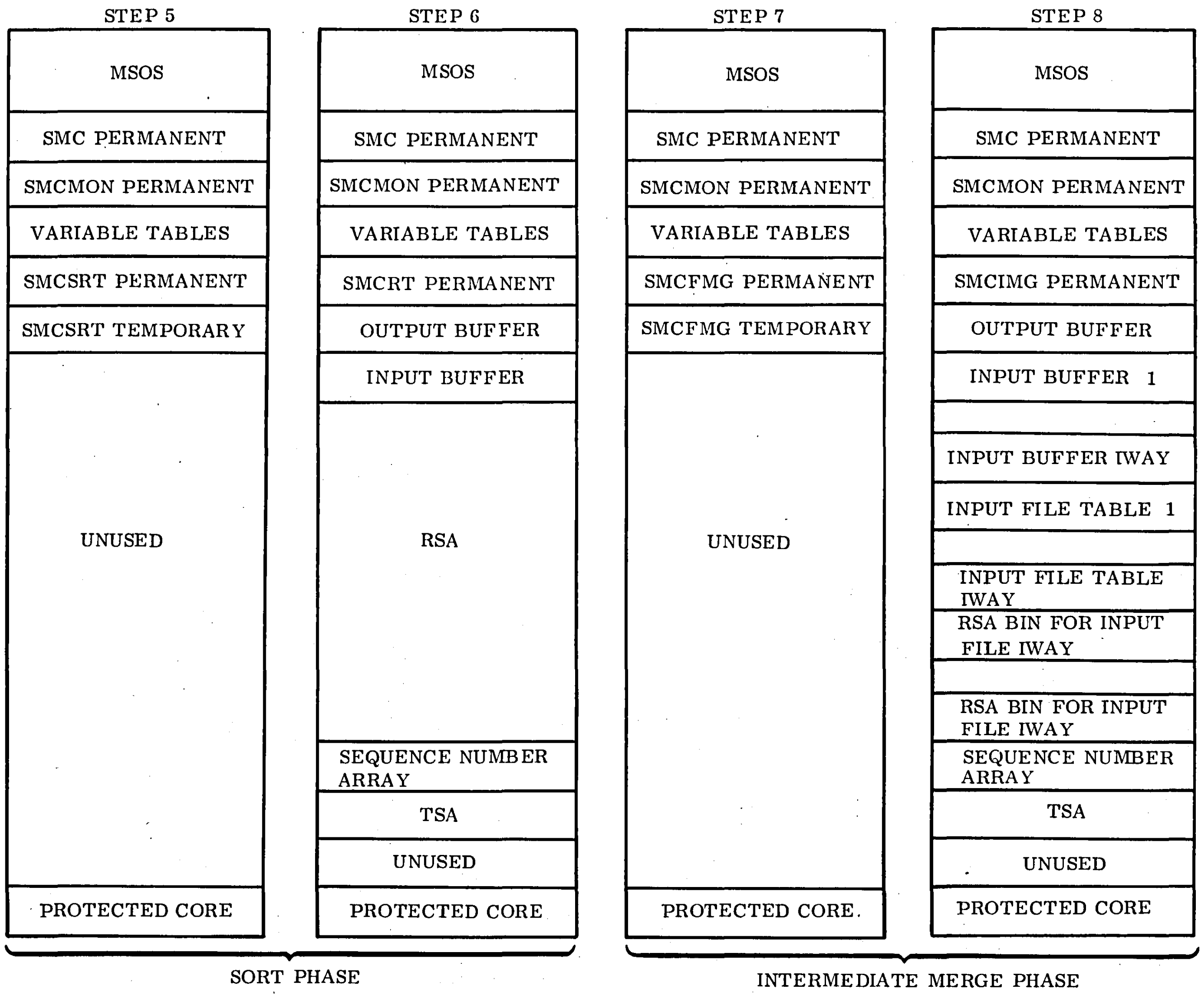


Figure C-1. Core Maps during Sort/Merge Processing (Sheet 2 of 3)

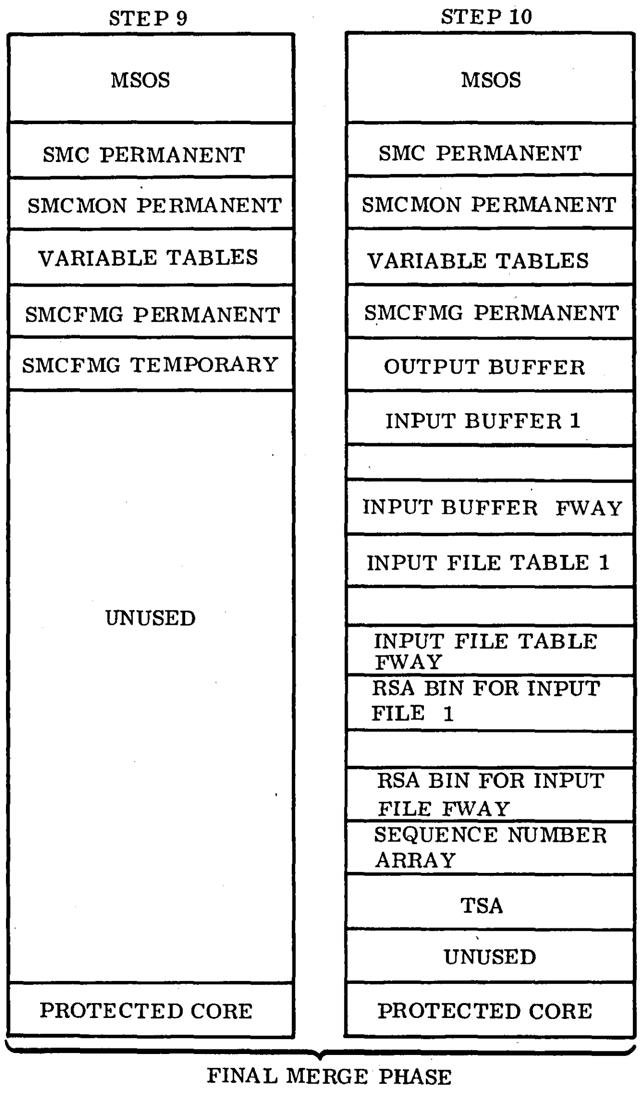


Figure C-1. Core Maps during Sort/Merge Processing (Sheet 3 of 3)

PROGRAM DESCRIPTION

The following description of internal structures is provided to assist the analyst in understanding the internal operation of the Sort/Merge utility package. Users need not normally be concerned with this information. Specific parameters in this section are subject to change as a result of program modifications by CDC.

Sort/Merge is executed in six separate phases:

- Phase 1 – Calling Sort/Merge initializer module (SMC)
- Phase 2 – Executing the main program (SMCMON)
- Phase 3 – Defining and checking the run definition (SMCEDT)
- Phase 4 – Sorting (SMCSRT)
- Phase 5 – Intermediate merging (SMCIMG)
- Phase 6 – Final merging (SMCFMG)

PHASE 1 – CALLING SORT/MERGE (SMC)

The job processor is activated as usual (i.e., *JOB). Then, before calling Sort/Merge, the operator selects the input and list devices using a *K,Ix,Ly statement. This causes the run definition to be accepted from logical unit x and the general output of all messages to be made to logical unit y. For example, if the conversational display terminal (CDT) is used both for run definition and for comments to the operator, and if the CDT is assigned to I/O channel 1, *K,I4,L4 may be used. Sort/Merge is then activated with an *SMC statement. Appendix C shows the phases of loading and executing all Sort/Merge programs.

After Sort/Merge is loaded and activated, the MSOS GTFILE routine is used to load the Sort/Merge monitor, SMCMON (phase 2). When SMCMON is loaded, SMC activates it. SMCMON controls all further operations. SMCMON immediately loads the editor SMCEDT (phase 3), which receives the operator's definition of the run, checks it for accuracy, and informs the operator of selected run parameters if the operator so requests.

Sort/Merge may be run in any of three interactive modes:

- Level 0 – No operator interaction
- Level 1 – Limited interaction
- Level 2 – Complete interaction by use of detailed informative and error messages.

Level 2 mode requires the most time and operator assistance, but has the advantage that extensive error recovery is available. On the other hand, if the file manipulation techniques have been fully debugged, and the input files are standardized and error-free, level 0 mode provides quick and efficient operation.

After the run is defined, SMCEDT is ejected as SMCMON loads the sorter, SMCSRT (phase 4). If merging is required (e.g., not all files could be sorted within core,and intermediate disk storage of files is necessary), SMCSRT is ejected by the Sort/Merge monitor and the intermediate merge program (SMCIMG) is loaded and executed (phase 5). Finally, in all cases, the Sort/Merge monitor loads and executes the final output processing program, SMCFMG (phase 6). Then control returns to the Sort/Merge monitor, which sends a message to the operator informing him that the sort/merge/copy request has been completed.

PHASE 2 – EXECUTING THE MAIN PROGRAM (SMCMON)

The Sort/Merge monitor (SMCMON) has an initial preparative operation (phase 2), but it also receives control after each other phase is completed. SMCMON contains logic and tables common to the other phases of Sort/Merge. It is responsible for calls to the following:

- SMCEDT, SMCSRT, SMCIMG, and SMCFMG
- The routines to display messages to the user
- The routines to accept replies from the user
- The logic for controlling physical and logical I/O operations
- The routine for comparing user-defined key fields
- Tables of user-supplied and dynamic parameters, including interphase communication

PHASE 3 – DEFINING AND CHECKING RUN DEFINITION (SMCEDT)

SMCEDT prompts (asks) the user to supply parameters defining the type of run desired (see section 4 for the format of the messages that prompt the operator). Note, however, that there is no prompting for the first user parameter that is required; i.e., the level of prompting that selects the interactive mode. The operator must select this before he receives the first request for input parameters.

The prompting levels are as follows:

Level	Interaction Mode
0	No prompting; the first error is fatal. An appropriate error message is sent, but the sort/merge/copy request is aborted and must be resubmitted.
1	The required statement type is named, and limited interactive error recovery is provided.

Level	Interaction Mode
2	The required statement type is named, its format is described, and the user is explicitly reminded that Sort/Merge is awaiting an operator reply. The full range of interactive error recovery is provided.

At the time it is supplied, each user parameter is checked and tabulated. For prompting levels 1 and 2, interactive error recovery is provided. The three following examples show the level of input checking and interaction:

1. The operator supplies an erroneous character in a RUN (input file size/number), KEYS (key definition), INFILE (input file definition), or OUTFILE (output file definition) statement.

   The program returns an EXPECTED. . . RECEIVED. . . message. The operator then re-enters the full statement with a proper character string at the point indicated.

2. The fixed tables are incompatible with file size.

   This is an unrecoverable condition and the run is aborted.

3. The input statements have been checked and accepted during a sort run.

   SMCEDT presents the operator with information about record sorting, and intermediate or final merging (G = n, iway = a, or fway = n). These are explained in the SMCSRT, SMCIMG, and SMCFMG sections below.

SMCEDT returns control to SMCMON after the input parameters have been checked and accepted.

If a set of parameters is to be used repeatedly, it is convenient and efficient to degug using level 2 prompting interactively, and then to transfer the parameters to a permanent medium such as paper tape.

PHASE 4 — SORTING (SMCSRT)

SMCSRT is used during copy and sort-only runs. In both cases, multiple user input fiels are permitted.

The program asks the user to identify each input file, one at a time, in the order it was described to SMCEDT. The user must direct SMCSRT to process the file with or without further operator intervention or to delete the file. SMCSRT does not request operator assistance for another input file until the program is finished with the file just specified.

For each user input file included in the run (whether copy or sort-only), skipcnt records are deleted from the run and docnt records are included in the run.

During a copy run, the input logical records are copied in the original order onto the user output file. When all input records have been copied, SMCSRT returns control to SMCMON and that program terminates the run.

In a sort-only run, SMCSRT also copies the input logical records included in the run. The logical records are not normally copied in their original order, but onto work files instead.

A queuing and partial sorting routine (tournament) delays and saves some logical records while advancing others, causing logical records to be copied in bursts (called sequences or strings) of user-defined order. Wherever possible, pointers are moved instead of records, minimizing actual record movement. SMCSRT moves each logical record twice, first from the input buffer to the queue, and then from the queue to the output buffer.

The group size G of the tournament is the number of logical records that the tournament can process (delay and sort) at one time.

Excluding the first sequence and the last sequence produced by the tournament, the average number of logical records per sequence is G*O, where O is a function of the inherent ordering of tournament input relative to the output order of the tournament.

Some sample O values are 1, 2, and infinity, for inverse, random, and perfect ordering of tournament input, respectively. Normally, O assumes a value greater than or equal to 1. SMCSRT usually produces more than one sequence.

SMCSRT moves records to work files in preparation for the recursive merging of the two or more sequences after each such output. SMCSRT returns to SMCMON with a flag that causes SMCMON to continue the run. To permit subsequent merging, both the sequences themselves and a sequence directory must be written by SMCSRT onto work files.

Whenever feasible, SMCSRT generates a single sequence on the user output file. In this case, SMCSRT returns control to SMCMON, with a flag causing SMCMON to terminate the run. Such a single output to the user output file occurs when both of the following conditions exist:

- Tournament input is less than G logical records. (All records can be contained in core for immediate sorting: merging of sorted partial outputs is unnecessary.)
- blksiz for the user output file is greater than or equal to wkbksz plus the largest blksiz of the user input files.

The sorting process always lists the count of logical records output by SMCSRT, whether these are work files or the final user output file. The count is saved both for sort-only runs and for copy runs.

When the SMCSRT phase is a prelude to merging, the program also lists s, the number of sequences produced by SMCSRT.

PHASE 5 — INTERMEDIATE MERGING (SMCIMG)

MERGING

SMCIMG is not used either in a copy run or a merge-only run.

Intermediate merging is required for a sort-only run when neither SMCSRT alone nor SMCSRT followed by SMCFMG would produce a single sequence on the user output file from the user input files.

SMCIMG merges the sequences produced by SMCSRT until few enough sequences exist to allow SMCFMG to merge the remaining sequences into one final sequence on the user output file. SMCIMG deals only with work files, writing an entry in the sequence directory for each sequence of SMCIMG oututs; i.e., SMCIMG adds to the sequence directory begun by SMCSRT.

SMCIMG may have to merge strings produced earlier by SMCIMG, as is shown in the second example in section 3. The strategy to perform the recursive merging operations may be considered to be a tree structure formed by the pattern of merges performed first by SMCIMG and then by SMCFMG. Prior to any merging, SCMIMG determines and optimizes this tree structure; i.e., factors of core size and numbers of merges are used as parameters to determine the most efficient method of successively merging files.

SMCIMG first combines 1STWAY sequences, then it combines the IWAY (the maximum possible for SMCIMG) sequences. The shortest strings (determined by the number of inputs to the tournament) are combined first.

The result of the preceding strategy is minimization of U, the unit strings rating of all merging performed by SMCIMG; i.e., how many times all data from an original string from SMCSRT is processed by SMCIMG. Before the start of any merging, SMCIMG computes and lists 1STWAY and U. Note that on the first pass, records themselves are sorted in bins. After that time, pointers to records are successively merged, so that the records themselves need not be moved. The following example makes the process clearer:

Consider the second sample run in section 3 (a sort run with level 1 prompting). Given available core size, work block size (wkbksz), record length (reclth), and number of records (docnt), the tournament inputs are:

- Core size (internally known to the program)
- wkbksz (determined by the operator using appendix G criteria): 3000 words
- reclth (fixed by the records themselves): 400 words
- docnt (operator specifies number of records): 300 words

The ideal size tournament for these parameters is G = 13. Therefore, the first sorting must handle 300/13 = 23 + 24 sorts (sequences). On this sort, the records themselves are compared by the tournament (sort criteria uses a C,A,76,5 key; i.e., character key, ascending from A to Z, and 76*5 = 380 characters long). On subsequent mergers, 13 record string pointers (not the records themselves) are merged: 13 by 13 into 26 entry strings; then 26 by 26 into 52 entry strings; then 52 by 52 into 104 entry strings; and finally 104, 104, and 92 into a 300-entry string.

Note that each earlier string also had one string with less than the normal number of entries (e.g., one string with one entry in the 13-entry strings to make up the full 300 records).

When the user files have the same effective transfer rate as the work files, total run time is roughly 1 (for SMCSRT) + U/S (for SMCIMG) + 1 (for SMCFMG) = 2 + U/S * SMCSRT time.

In a sort-only run, SMCSRT and SMCFMG each involves at most one pass of all the input data, but SMCIMG could involve several passes. Therefore, at best, SMCIMG is not used and, at worst, SMCIMG accounts for most of the run time.

When merging records, SMCIMG uses an adaptation of the tournament of SMCSRT. The SMCIMG queue is being fed by several input sequences at the same time, in contrast with the serial input of probably unsorted files to the SMCSRT queue.

Since the input to the SMCIMG queue is presumably sorted, the separate input buffers constitute the actual logical record delay area, and, within a single merge, each logical record is moved only once from the corresponding input buffer to the output buffer.

One string should always be produced by each merge, since the input to the merge is presumably sorted.

SEQUENCE CHECKING

If the user selects sequence checking to detect hardware or software malfunctions, SMCIMG compares each logical record about to be output (residing in an input buffer) with the logical record previously output (residing in the output buffer) within the same merge. This comparison is made before moving the new record to the output buffer. If an error occurs, the last record output rather than the new record might be the cause of the sequence error. The user must determine which record is in error. Therefore, when SMCIMG detects a sequence error, it sends a message to the operator giving him the option of deleting the new record. Use of this option avoids repeated sequence errors on subsequent merges involving the same record.

SMCIMG returns control to SMCMON when the number of sequences to be merged is less than or equal to FWAY, the maximum number of sequences that SMCFMG can handle. Just before returning to SMCMON, SMCIMG lists the number of records deleted and the number of records output. SMCMON then loads and gives control to SMCFMG.

PHASE 6 — FINAL MERGING (SMCFMG)

SMCFMG is used in merge-only runs and it is the normal last phase of a sort run (SMCSRT sometimes completes a sort-only run). Copy runs do not use SMCFMG.

SMCFMG performs a single merging operation in which the input files are either all user files (for merge-only runs) or all work files (for the final merge of a sort run).

Therefore, SMCFMG is much like SMCIMG, although it lacks multiple merge logic. On the other hand, it does contain logic for processing user input files.

The SKIPCNT and DOCNT features are available for the user input to a merge-only run.

The merge tournament logic and sequence checking logic is the same as is used in SMCIMG.

After the final merging and before returning control to SMCMON, SMCFMG announces the number of records deleted and the number of records output. Also, for a final merge, even if no records were deleted, SMCFMG checks the number of records output by SMCFMG against the output count for SMCSRT and sends a message showing any disagreement between the two counts.

After receiving control from SMCFMG, SMCMON returns control to MSOS, ending the run.

SAMPLE TOURNAMENT TREE STRUCTURE

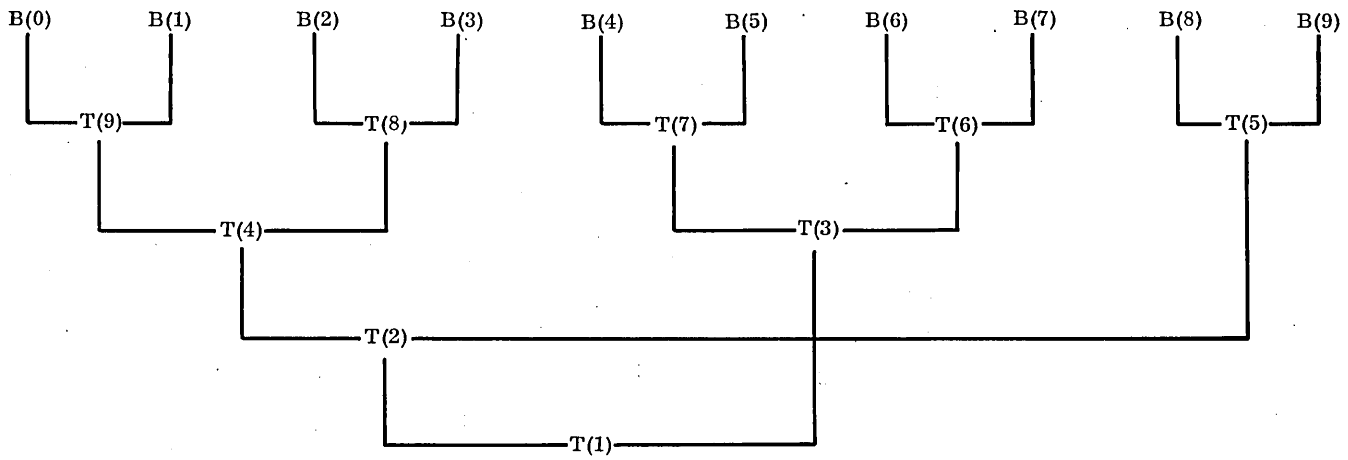


Figure E-1. Sample Tournament Tree Structure

TOURNAMENT INITIALIZATION

The tournament region is filled with winning dummy records; i.e., phony records identified by use of a string number less than G. These records therefore win over any real records, which must have a string number greater than or equal to G.

The tags are set up in such a way as to force the tournament to structure itself properly as the real records are submitted, one by one.

In the initialization example shown in figure F-1 (which corresponds to the tree structure in appendix E), there is an apparently anomolous condition: T(2) indicates that B(2) was a loser in the last contest associated with T(2). Inspection of the sample tree structure shows that T(2) could not be concerned with B(6). This anomolous condition disappears as real records are submitted to the tournament, since each real record except the Gth initially loses to a winning dummy and therefore gets recorded in a necessarily relevant tag.

TAG:	T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	T(7)	T(8)	T(9)
CONTENTS:	8	6	4	2	9	7	5	3	1

Figure F-1. Initialization Example

OPTIMIZING THE WORK BLOCK SIZE

Version 1.0 of Sort/Merge does not double buffer. Therefore, the following equations compute an approximate time for a Sort/Merge sort run, assuming there is no overlapping of input, compute, and output operations.

Let CTRANS	= Number of words of core for transient tables, logic, buffers
C	= Number of words of core available to Sort/Merge
CRES	= Number of words of core needed for the resident parts of the phases SMC and SMCMON, including fixed tables
K	= Number of sort keys (from the KEYS statement)
F	= Number of input files (SKIPCNT and DOCNT)

Assume that five words are needed for each average key field (a minimum of two words and a maximum of seven words).

Currently, SMCEDT uses eight words to tabulate the parameters of each input file.

Then CTRANS = C-CRES-5*K-8*F

Let G	= Number of real bins in the internal sort tournament
CSRT	= Number of words needed for resident SMCSRT logic
BIN	= Maximum user input block size
BWORK	= Work block size (WKBKSZ in the RUN and INFILE statements)
R	= Logical record size (RECLTH)

Then G = (CTRANS-CSRT-BIN-BWORK+1)/(R+2)

Let IWAY	= Maximum way-of-merge for SMCIMG
CIMG	= Number of words needed for resident SMCIMG logic

Currently, 30 words are needed for each file table, two words are needed for a merge tournament bin, and one word is needed for each merge tournament sequence number and tag.

Then IWAY = (CTRANS-CIMG-BWORK+29)/(BWORK+34)

Let FWAY	= Maximum way-of-merge for SMCFMG
CFMG	= Number of words needed for resident SMCFMG logic
BOUT	= User output block size

Then FWAY = (CTRANS-CFMG-BOUT+1)/(BWORK+34)

Let TREE(X)	= Average number of tournament levels encountered per logical record for a tournament with G=X.
L2(X)	= LOG_2X

Then TREE(X) = 1 + L2(X)+((2**L2(X))/X)

The CPU time consumed within SMCSRT, SMCIMG, and SMCFMG is accumulated on a per block basis for MSOS calls, and on a per logical record basis for blocking, deblocking, and tournament. Other costs such as per sequence costs are insignificant.

Let TCPUS	= CPU time used by SMCSRT per logical record
ACPUS BCPUS CCPUS DCPUS	= Four separate constants
BINAVG	= Average user input block size

Then TCPUS = (ACPUS/BINAVG+DCPUS/BWORK)*R+ BCPUS+CCPUS*TREE(G)

Let TCPUI	= CPU time used by SMCIMG per logical record per pass
ACPUI BCPUI CCPUI DCPUI	= Four separate constants

Then TCPUI = (ACPUI+DCPUI)*R/BWORK+BCPUI+CCPUI+ TREE(IWAY)

Let TCPUF	= CPU time used by SMCFMG per logical record
ACPUF BCPUF CCPUF DCPUF	= Four separate constants

Then TCPUF = (ACPUF/BWORK+DCPUF/BOUT)*R+ BCPUF+CCPUF*TREE(FWAY)

For many devices there is a certain access time per block, followed by a transfer time.

Let TINUS	= I/O time per logical record for user input
TINWK	= I/O time per logical record for work input per pass
TOUTWK	= I/O time per logical record for work output per pass
TOUTUS	= I/O time per logical record for user output

AINUS
AINWK
AOUTWK
AOUTUS = Four separate constants

Then TINUS = AINUS/BINAVG*R

TINWK = AINWK/BWORK*R

TOUTWK = AOUTWK/BWORK*R

TOUTUS = AOUTUS/BOUT*R

Let S = Number of strings written by SMCSRT

NRGC = Number of user input logical records

ORDER = Rating of inherent order in user input: 1, 2, infinity for inverse, random, perfect, respectively

Then S = NREC/(ORDER*G)

Let U = Unit strings rating of SMCIMG

P = U/S ; i.e., the integer portion of the number of SMCIMG merge passes

DFAN = Inadequacy of the fanout after P SMCIMG passes

DNM = Number of extra IWAY merges to increment the fanout by DFAN strings

Then P = $\log_{IWAY}(S/FWAY)$

DFAN = S - (IWAY**P)*FWAY

$$DNM = \frac{DFAN}{IWAY-1},$$

$$U = P*S + \begin{cases} DNM\ *IWAY & \text{if DNM is an integer} \\ DNM\ +DFAN & \text{if DNM is not an integer} \end{cases}$$

Total time, T, for the Sort/Merge run is:

```
T = NREC*(TINUS+TCPUS+TOUTWK
        +(U/S)*(TINWK+TCPUI+TOUTWK)
        +TINWK+TCPUF+TOUTUS)
```

The constants used in the preceding equations are dependent upon the specific I/O device and CPU involved. The coefficients should be determined by programming the preceding equations and fitting the equations to actual timings.

Once the coefficients are known, the user may wish to prepare a program to accept parameters such as K,F,BIN,R, BOUT,BINAVG,NREC,ORDER. The program would vary BWORK, and hold all other symbols as constants.

The value of BWORK for which T is minimized would be output by the program for use in running Sort/Merge.

SORT/MERGE MERGING STRATEGY

With maximal merges, except for 1STWAY, Sort/Merge extends the tree from the output of the final merge (performed by SMCFMG) to the original strings (generated by SMCSRT). SMCIMG determines to optimal tree structure prior to merging any of the SMCSRT sorted strings. The calculations below show how U (unit strings rating) is minimized by minimized tree height (the number of merging levels, which is four in the sample shown in figure H-1), and by merging the shortest strings first.

$$\text{P} = \log_3(57/4) = 2$$

$$\text{DFAN} = 57-(3**2)*4 = 21$$

$$\text{DNM} = \frac{21}{3-1} = 10.5$$

$$\text{U} = 2*57+ 10.5 +21 = 146$$

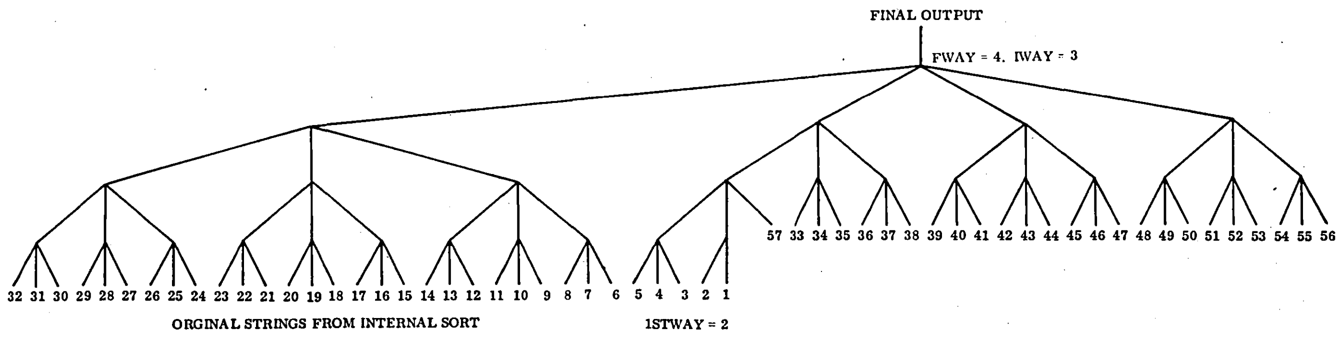


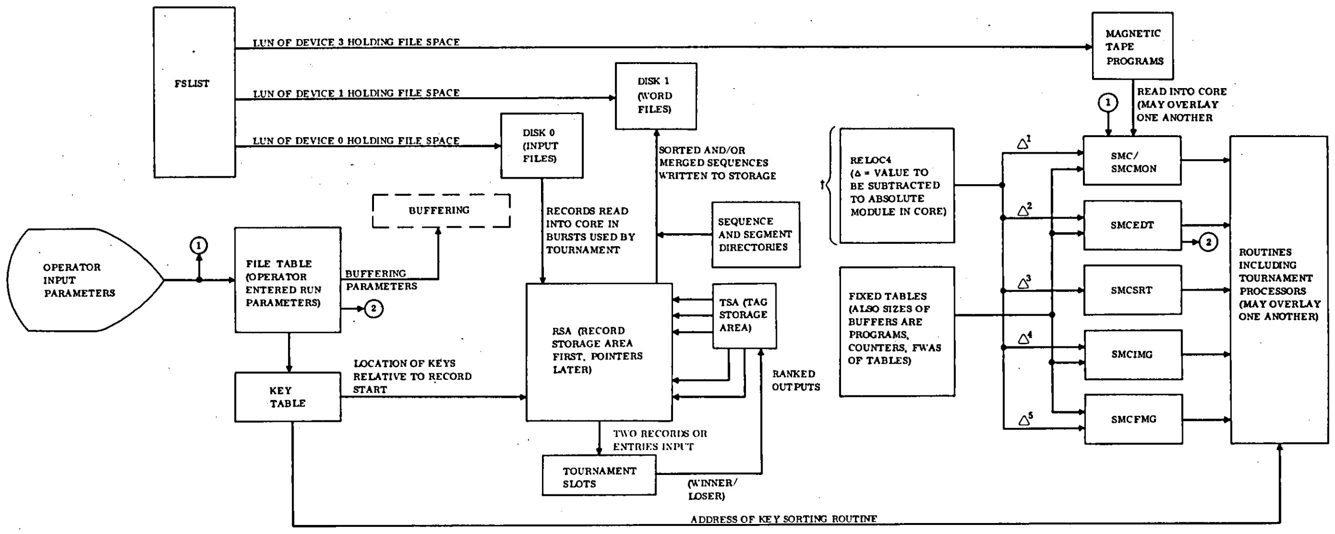
Figure H-1. Merge Tree

TABLE FORMATS AND USAGE

The following descriptions of internal structure are provided to assist the analyst in understanding the internal operation of the Sort/Merge utility package. Normally, users need not be concerned with the detail level of this appendix. Specific parameters in the section are subject to change as a result of program modifications by CDC.

The Sort/Merge module uses several tables resident in core or on disk to perform sort/merge/copy tasks. The relation of the tables to one another is shown in figure I-1. A summary of the tables, their location, and their use follows:

Location	Name	Use
Core	FSLIST	Gives identification of each logical unit that holds file space. Table is actually part of resident MSOS.
Core	RELOC4	Holds relocation values for the major programs: SMCMON, SMCEDT, SMCSRT, SMCIMG, and SMCFMG
Core	Fixed tables (y tables)	Hold various parameter values for run values and for programs while in core
Core	Variable Tables	
	File table	Holds parameters for all user and work files
	Key tables	Hold keys and key locations in each file
Core	Tournament Tables	
	RSA (record storage area, i.e., bins)	Holds records (internal sort) or record pointers (intermediate or final merge)
	Sequence number array	Pointers to RSA records as written in sequences of records
	TSA (tag storage area)	Bin indices
Disk	Sequence directory	Parameters for each work file, including filnum
Disk	Segment directory	Links filnums for work files that extend to a second disk



FSLIST
LUN OF DEVICE 3 HOLDING FILE SPACE
LUN OF DEVICE 1 HOLDING FILE SPACE
LUN OF DEVICE 0 HOLDING FILE SPACE
MAGNETIC TAPE PROGRAMS
READ INTO CORE (MAY OVERLAY ONE ANOTHER
DISK 1 (WORD FILES)
DISK 0 (INPUT FILES)
SORTED AND/OR MERGED SEQUENCES WRITTEN TO STORAGE
BUFFERING
RECORDS READ INTO CORE IN BURSTS USED BY TOURNAMENT
SEQUENCE AND SEGMENT DIRECTORIES
RELOC4 (Δ = VALUE TO BE SUBTRACTED TO ABSOLUTE MODULE IN CORE)
†
OPERATOR INPUT PARAMETERS
FILE TABLE (OPERATOR ENTERED RUN PARAMETERS)
BUFFERING PARAMETERS
KEY TABLE
LOCATION OF KEYS RELATIVE TO RECORD START
RSA (RECORD STORAGE AREA FIRST. POINTERS LATER)
TSA (TAG STORAGE AREA)
RANKED OUTPUTS
TWO RECORDS OR ENTRIES INPUT
TOURNAMENT SLOTS
(WINNER/ LOSER)
FIXED TABLES (ALSO SIZES OF BUFFERS ARE PROGRAMS, COUNTERS, FWAS OF TABLES)
Δ1
Δ2
Δ3
Δ4
Δ5
SMC/ SMCMON
SMCEDT
SMCSRT
SMCIMG
SMCFMG
ROUTINES INCLUDING TOURNAMENT PROCESSORS (MAY OVERLAY ONE ANOTHER)
ADDRESS OF KEY SORTING ROUTINE
1
2

†ABSOLUTE ADDRESSES OF PROGRAMS WHEN BEING EXECUTED IN CORE

Figure I-1. Sample Relation of Sort/Merge Tables, Programs, and User Records

FSLIST

This file space list table (figure I-2) is a core-resident part of MSOS.

For each logical unit that holds file space for the file manager, there is a corresponding entry in FSLIST. Therefore, n such logical units would correspond to n contiguous entries in FSLIST.

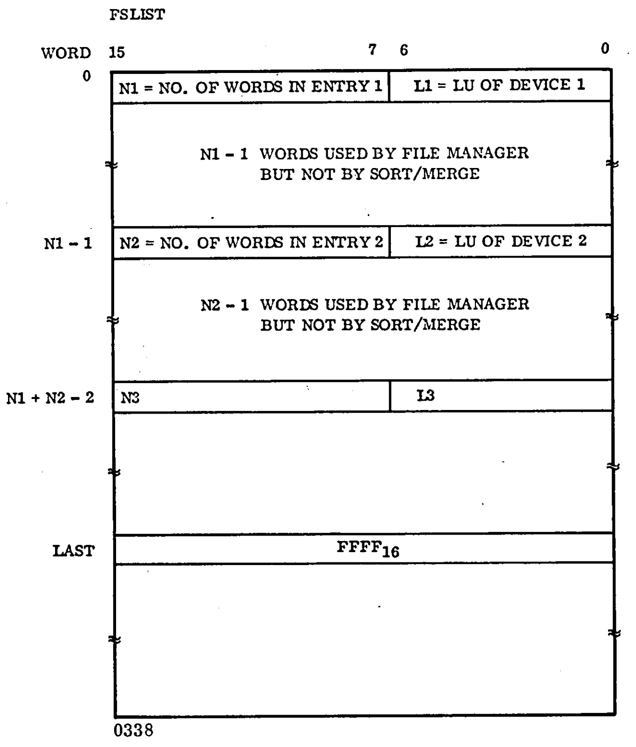
Sort/Merge uses FSLIST to determine which logical units hold file space. This information is used when Sort/Merge attempts to define a new segment for a work file (c.f. segment list).

Figure I-2 is a portion of FSLIST used by Sort/Merge. The table is vertical in format, with all information for each file space device held in contiguous words.

RELOC4

RELOC4 is the relocation values table. RELOC is the routine that adds a relocation factor to each of the relocatable programs designated by the table RELOC4. The table is used for SMCMON, SMCEDT, SMCSRT, SMCIMG, and SMCFMG. The relocation factor used is the first word address of a reference point labeled HERE. Each entry in RELOC4 has the value:

FWA of some relocatable program – FWA of HERE



FSLIST
WORD
15
7
6
0
0
N1 = NO. OF WORDS IN ENTRY 1
L1 = LU OF DEVICE 1
N1 - 1 WORDS USED BY FILE MANAGER BUT NOT BY SORT/MERGE
N1 - 1
N2 = NO. OF WORDS IN ENTRY 2
L2 = LU OF DEVICE 2
N2 - 1 WORDS USED BY FILE MANAGER BUT NOT BY SORT/MERGE
N1 + N2 - 2
N3
L3
LAST
FFFF16
0338

Figure I-2. FSLIST Format

FIXED TABLES

These tables are grouped in a fixed format; i.e., the tables have the same size and format for each run, but values for the entries differ from run to run.

Entries in the 98 word fixed table are grouped into six categories:

- Parameters that are part of a general definition of the current run
- Addresses of items in variable tables
- Memory limits
- Phase sizes
- Dynamic variables (e.g., a record count)
- Logic addresses

VARIABLE TABLES

Since the number and types of input files and key fields differ from run to run, a variable amount of core must be used to describe these variables.

The file table has two sections. The first portion is built by SMCEDT; the latter portion is built later during the run. Only the portion built by SMCEDT is a part of the variable tables. The file table expansions are later generated higher in core but are derived from the unexpanded file tables in the variable tables.

Logic overlays (SMCSRT, SMCIMG, SMCFMG) are brought into core immediately next to and above variable tables (see appendix C).

FILE TABLE

There is one file table for each user file and for each work file. The table stores both static and dynamic parameters. The first group of parameters is set up when input parameters are processed:

- Type/ID: storage device/skip and do flags/ASCII or Binary/LUN/work or data file flags/extended or normal file
- Filnum
- Record and buffer length
- Skip and do counts

The remaining parameters are added as the file is processed

- I/O buffer address
- Record, block, and error counts

- Deblocking data (current FRB from which data is being drawn)
- Work area information
- Status of file when file manager last used it
- File address of FRB containing current record

KEY TABLE

SMCEDT sets the key table with the user's key definitions as these are read and analyzed from the KEYS message input.

A completed key table describes each user key field in a fashion chosen for efficient use by CMPKEY, the only key table user other than SMCEDT. (If there were other users of the key table, CMPKEY efficiency would still be the prime influence on key table design because CMPKEY is used much more often than any other Sort/Merge logic.)

Key table entries have one of three formats:

1.	Fixed-length key:	Word 0 =	FWA of key minus FWA of logical record
		Word 1 =	FWA minus 1 of logic tailored to type and order
2.	Variable-length key:	Word 0 =	FWA of key minus FWA of logical record
		Word 1 =	FWA minus 1 of logic tailored to type and order
		Word 2 =	LWA+1 of key minus FWA of logical record

Additional words are needed to describe keys that use multi-pass processing (see the example that follows).

3.	Key table terminator:	Word 0 =	$FFFF_{16}$

The key-table terminator appears at the end of each key table to indicate the end of the table.

The fixed-length key format is used to describe a key of fixed-length, which is one of the following key types:

- Floating point
- Signed binary
- Logical binary
- Upper character (within a word)
- Lower character (within a word)

Since each of these key types is of fixed length, the entry need only mention the key type, the relative start, and the order (ascending or descending). It is assumed that the user-defined key fields are oriented in the same way relative to the start of each logical record.

Word 0 designates the start of the key field relative to the start of any logical record containing that key field. Word 1 designates the address of the routine that processes that type and order of the key field.

For a character key field on a word boundary with a word length of two or more, the variable-length key format is used. Words 0 and 1 are used as before, but word 2 is added to designate the end of the key field relative to the start of any logical record containing that key field.

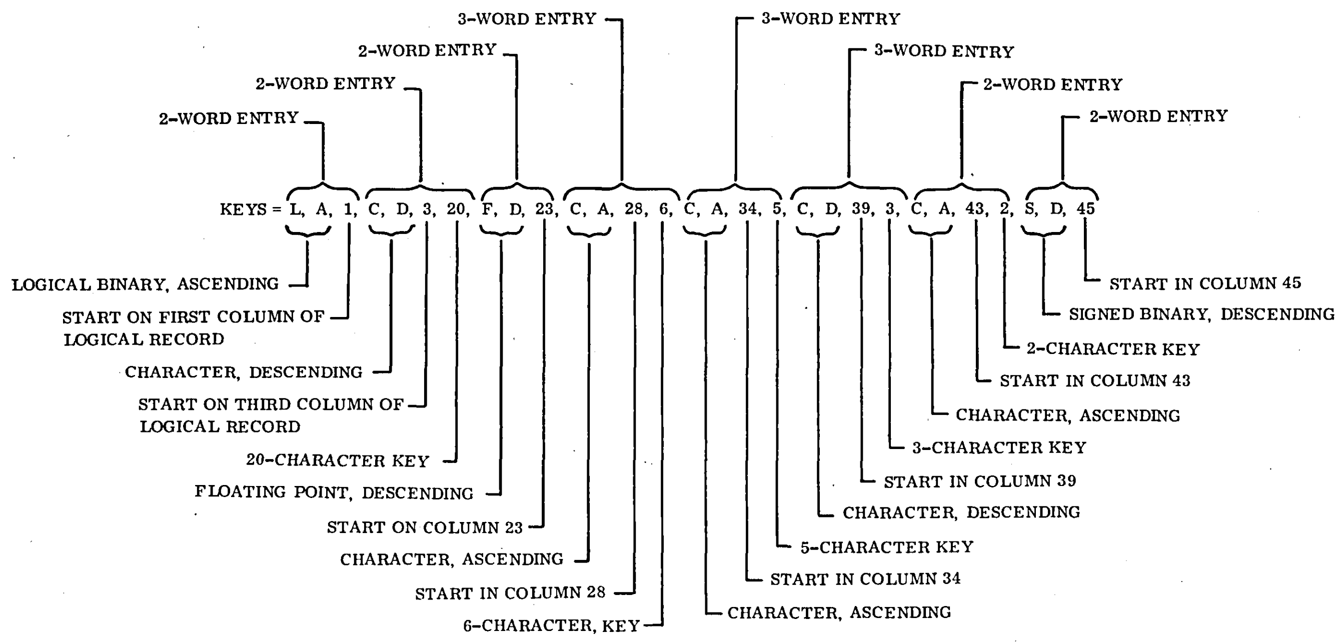
The following key table rules list the routines and constraints used for key processing.

1. F,A/D, even column is illegal (F = floating point, A/D = ascending or descending).
2. F,A/D, odd column uses CMPFA/CMPFD.
3. S,A/D, even column is illegal (S = signed binary).
4. S,A/D, odd column uses CMPSA/CMPSD.
5. L,A/D, even column is illegal (L = logical binary).
6. L,A/D, odd column uses CMPLA/CMPLD.
7. C,A/D, even column, 1 uses CMPCLA/CMPCLD (C = character).
8. C,A/D, odd column, 1 uses CMPCUA/CMPCUD.
9. C,A/D, even column, 2 uses CMPCLA/CMPCLD followed by CMPCUA/CMPCUD.
10. C,A/D, odd column, 2 uses CMPLA/CMPLD.
11. C,A/D, even column, 3 uses CMPCLA/CMPCLD followed by CMPLA/CMPLD.
12. C,A/D, odd column, 3 uses CMPLA/CMPLD followed by CMPCUA/CMPCUD.
13. C,A/D, even column 4 uses CMPCLA/CMPCLD followed by CMPLA/CMPLD followed by CMPCUA/CMPCUD.
14. C,A/D, odd column, 4+2n for n = 0,1,2... uses CMPWA/CMPWD.
15. C,A/D, even column, 5+2n for n = 0,1,2... uses CMPLA/CMPLD followed by CMPWA, CMPWD.
16. C,A/D, odd column, 5+2n for n = 0,1,2... uses CMPWA/CMPWD followed by CMPCUA/CMPCUD.
17. C,A/D, even column, 6+2n for n = 0,1,2... uses CMPLA/CMPLD followed by CMPWA/CMPWD followed by CMPCUA/CMPCUD.

The operator input KEYS statement generates the key tables, as shown in figure I-3.

TOURNAMENT ARRAYS

Tournaments take a number of input records (n) and order them by comparing two records at a time. Using an operator-designated key, the n records are successively



Word	Contents	Comments	Word	Contents	Comments
0	0	L, A, 1 entry	15	-1 + fwa of CMPCUA	
1	-1 + fwa of CMPLA		16	17	C, A, 34, 5 entry
2	1		17	-1 + fwa of CMPWA	
3	-1 + fwa of CMPWD	C, D, 3, 20 entry	18	19	
4.	11		20	-1 + fwa of CMPLD	C, D, 39, 3 entry
5	11	F, D, 23 entry	21	20	
6	-1 + fwa of CMPFD		22	-1 + fwa of CMPCUD	
7	13		23	21	C, A, 43, 2 entry
8	-1 + fwa of CMPLA		24	-1 + fwa of CMPLA	
9	14	C, A, 28, 6 entry	25	22	S, D, 45 entry
10	-1 + fwa of CMPWA		26	-1 + fwa of CMPSD	
11	16		27	$FFFF_{16}$	End of key table
12	16				
13	-1 + fwa of CMPLA	C, A, 34, 5 entry			
14	16				

NOTE: fwa = fixed word address

Figure I-3. KEYS Statement and Table

compared until the records are totally ranked; i.e., if a five-character letter key is used, with ascending order specified, comparisons are made until each record appears in its alphabetically ranked slot, as in a dictionary. Once every initial tournament (which compares the records themselves) is sorted alphabetically, the outputs may be merged in a hierarchy of merging operations. This hierarchy is the merge tree structure described earlier. For the example used before (second example in section 3), each of the 300 input records is first sorted using the keys by 24 tournaments, 13 records being sorted per tournament. These records are placed in bins called a record storage area (RSA), and written to disk in a single sorted sequence (string).

Since it may be necessary to have several sequences (tournaments), two other tables are necessary: the sequence number array, which has pointers to each record in the sequence, and the tag storage area (TSA), which is used for merging operations.

After the first sorting (assuming intermediate merging is necessary because not all the records could be sorted in core during a single pass or tournament), the RSA does not contain records but only pointers to records. At all times, the TSA contains the ranked outputs of the tournaments following merge operations. Merges occur only with sorted outputs so the two compared items (one from each string) need to be compared only to the same ranked item; e.g., the following are compared:

AB. . .	CA. . .	AB wins over CA
BA. . .	EN. . .	BA wins over CA
ED. . .	ES. . .	CA wins over ED
FI. . .	NE. . .	ED wins over EN
HA. . .	TO. . .	EN wins over FI
		ES wins over FI
String 1A	String 2A	FI wins over NE
		HA wins over NE
		TO is the residue
		String 1B

The formats of the RSA, sequence number array, and TSA are described below. Their relationship is shown in figure I-4.

RECORD STORAGE AREA (RSA)

The RSA is an array of bins used for the internal sorting operation or for the intermediate or final merging operation. For the internal sorting operation, each bin holds one logical record; for the intermediate merging and the final merging operation, each bin holds a two-word line with the contents defined below.

Word	Contents
0	FWA of a logical record residing in an input buffer
1	FWA of the file table corresponding to the input buffer and to the file from which the logical record came

The rank of each RSA bin relative to the other RSA bins is recorded in a binary tree structure called the TSA. Because this binary tree structure must have an even number of RSA bins for its oepration, an imaginary RSA bin may be used (see Sequence Number Array below) when the number of real bins is odd.

SEQUENCE NUMBER ARRAY

This table of one-word binary numbers has one entry for each RSA bin. Its entries are used as the major key fields in comparing the logical records associated with the RSA.

When the number of RSA bins is even, there is exactly one sequence number for each RSA bin. To preserve the even binary tree structure when the number of RSA bins is odd, a sequence number of 7FFF is added. This losing-dummy sequence number is held by the last word of the sequence number array.

TAG STORAGE AREA (TSA)

This table stores indices that link the sequence number array to the RSA bins. Suppose that the RSA bins, real and imaginary, are numbered 0, . . . , G-1. Each such number is called a bin index and each G bin index designates an RSA bin, as well as the corresponding entry in the sequence number array.

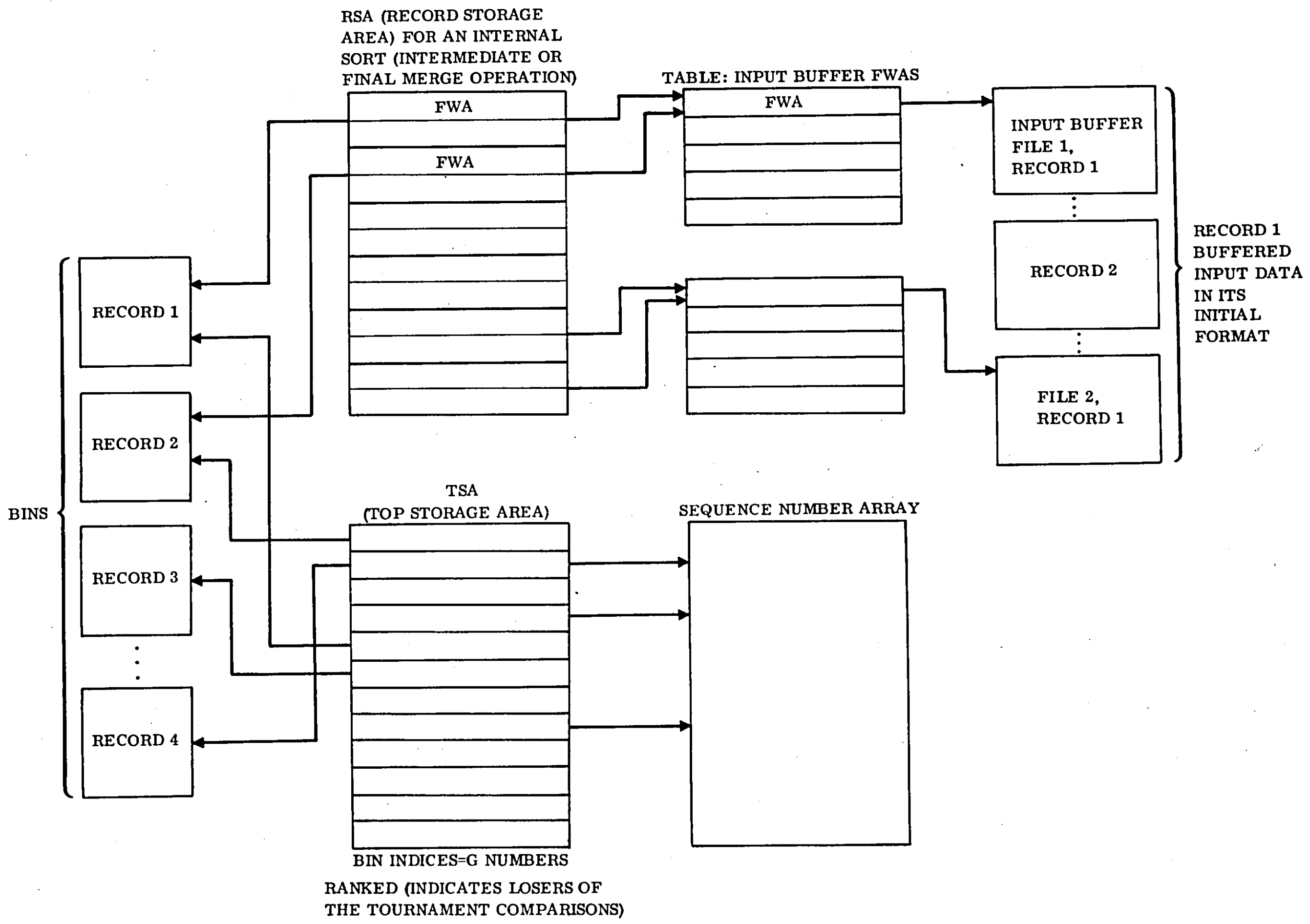
Each TSA entry contains a bin index to designate the loser of one contest between two RSA bins in a binary tree structure of such contents; i.e., in a tournament. Therefore, at any time there will be G-1 one-word entries in the TSA to indicate the relative rank of each RSA bin. At the end of all tournaments, the TSA will have all records ranked (sorted) according to the operator-entered sorting criteria (key).

DISK-RESIDENT TABLES

SEQUENCE DIRECTORY

Each sequence output on a work file has an entry in the sequence directory, which is a work disk file. The entries are actually file records. The format of the entry is:

Word	Contents
0	Length of the entry, including this length word
1	Number of the sequence this entry concerns
2	Flags and logical unit number upon which the first segment of the sequence resides (c.f. logical unit of file table for format of this word).
3	FILNUM (disk file identification) of the first segment of the sequence


RSA (RECORD STORAGE AREA) FOR AN INTERNAL SORT (INTERMEDIATE OR FINAL MERGE OPERATION)
FWA
FWA
TABLE: INPUT BUFFER FWAS
FWA
INPUT BUFFER FILE 1, RECORD 1
RECORD 2
FILE 2, RECORD 1
RECORD 1 BUFFERED INPUT DATA IN ITS INITIAL FORMAT
BINS
RECORD 1
RECORD 2
RECORD 3
RECORD 4
TSA (TOP STORAGE AREA)
SEQUENCE NUMBER ARRAY
BIN INDICES=G NUMBERS
RANKED (INDICATES LOSERS OF THE TOURNAMENT COMPARISONS)

NOTE: EACH TOURNAMENT MAY USE ONLY $\frac{1}{m}$ th OF THE TOTAL NUMBER OF RECORDS WHEN M IS A (NON-EVEN) DIVISOR OF THE TOTAL NUMBER OF RECORDS, N; FOR EXAMPLE, N = 300, M = 13, 26, 52, 104.

Figure I-4. Tournament Tables

SEGMENT DIRECTORY

When a work disk file runs out of space, Sort/Merge attempts to define a new filnum on another disk drive. The two FILNUMs, one exhausted and one fresh, may be viewed as segments of the same logical file.

The segment directory is a disk file containing entries linking segments (filnums) together. The entries are actually file records. The segment list is flagged in its file table as a user file to avoid recursion should the segment list logic try to extend the segment list across two devices. Format of the segment directory is:

Word	Contents
0	Length of the entry, including this length word
1	Flags and logical unit number of the old segment (logical unit number of the file table for the format of this word)
2	filnum of the old segment
3	Flags and logical unit number of new segment (logical unit number of file table for format of this word)
4	filnum of new segment

The above entries, which are added to the segment list as needed, remain in chronological order. However, during a Sort/Merge sort run, a FILNUM for a work file may be released and redefined several times. Therefore, the same

FILNUM may appear several times as an old FILNUM or as a new FILNUM within the segment list. However, using the chronolgical order, Sort/Merge resolves these ambiguities when looking up extension segments.

The sequences and their directory are the only files eligible for extension via the above mechanism.

Figure I-5 shows the relations of the files to the directories.

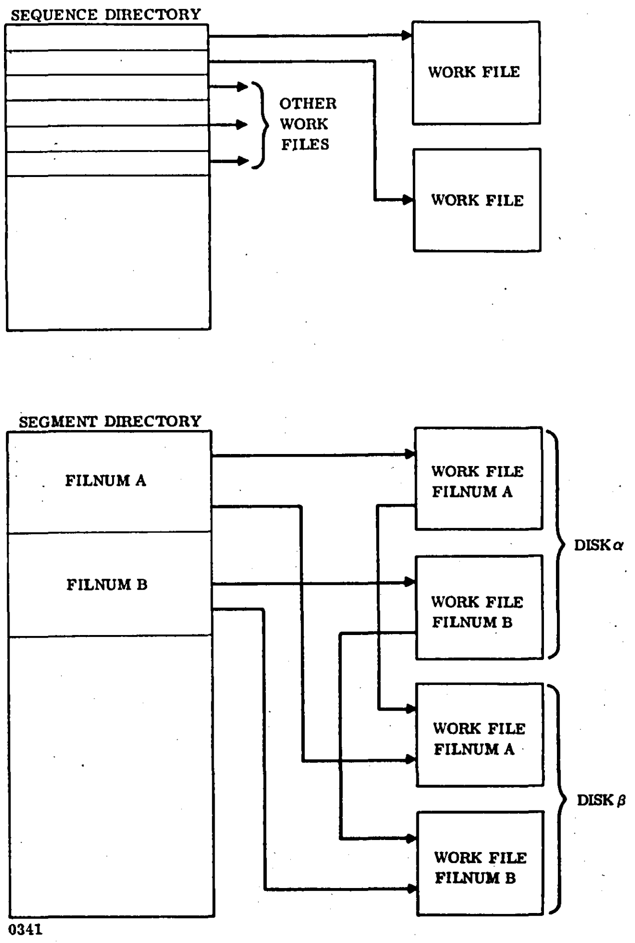


Figure I-5. Disk Resident Tables

FILE MANAGER USE BY SORT/MERGE

The following is a brief introduction to the file manager. For specific information, consult the MSOS File Manager Reference Manual.

NOTE

If an MSOS file is specified in the OUTFILE command, that file is released and redefined as an MSOS sequential file. This ensures that the output is stored beginning at record 1.

CAUTION

Only MSOS sequential files may be used for Sort/Merge output.

From the point of view of the file manager, mass storage is subdivided as follows: File space includes the logical unit which includes the file which includes the FRB (file record block) which includes the file record. Each file must reside entirely on one logical unit.

Each file is described by an FIS (file information segment), a group of 16 contiguous words recorded on mass storage. On mass storage, for efficiency, the FISs are blocked into 96-word FIS blocks, each containing up to five FISs. The FISs locate the files in the file record blocks.

Files are written in single or linked 96-word file record blocks (FRBs). Each FRB contains a three-word header followed by n records of variable or fixed length. If the record is longer than 93 words, linked sectors compose a single FRB. Conversely, several short records can be written on a single 96-word FRB. An example of an FRB is shown in figure J-1.

A file record consists of one header word, a two-word recorder-pointer if the file is indexed-linked, and zero or more data words. Assume that the file is not indexed-linked. A sample file record is shown in figure J-2.

NOTE

Sort/Merge skips removed file records without notice to the operator.

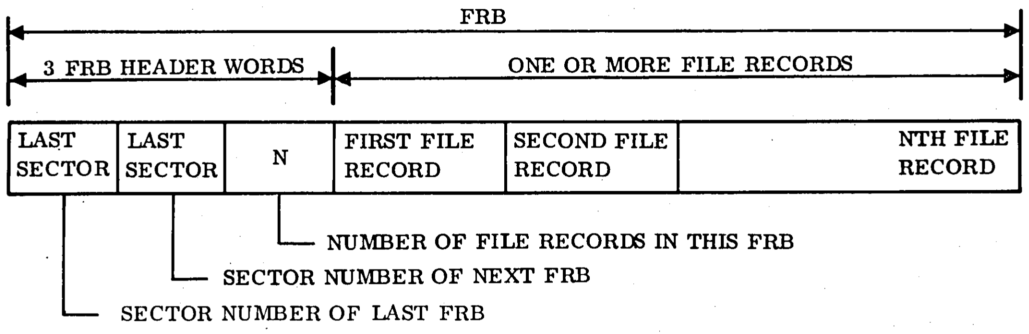


Figure J-1. Sample File Record Block

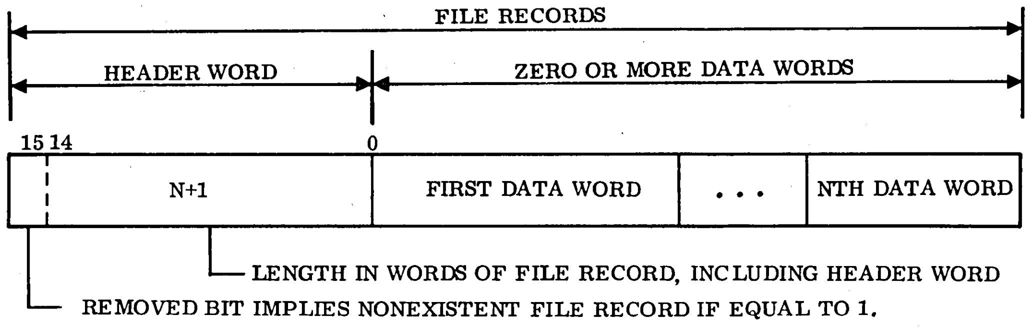

FILE RECORDS
HEADER WORD
ZERO OR MORE DATA WORDS
15 14
0
N+1
FIRST DATA WORD
. . .
NTH DATA WORD
LENGTH IN WORDS OF FILE RECORD, INCLUDING HEADER WORD
REMOVED BIT IMPLIES NONEXISTENT FILE RECORD IF EQUAL TO 1.

Figure J-2. Sample File Record

Sort/Merge subdivides file records into logical records, as if the data words of the file record were a block of magnetic tape. See figure J-3 for an example.

The file manager rounds the user-specified FRB size up to the nearest multiple of sector size. As a user of the file manager, Sort/Merge specifies that the maximum file record size should be blksiz+1. Therefore, the actual FRB size is

$$\frac{3+1+\text{blksiz}}{96} * 96 \text{ words}$$

Therefore, to conserve mass storage space, blksiz+4 should be equal to or slightly less than a multiple of 96 words.

The file manager uses an FIS directory (also held on mass storage) to find the files. The FIS directory is indexed by means of a scatter (hashed) code computed from filnum, the unique numeric file identifier. Since the FIS directory consists of pointers to FIS blocks, the file manager proceeds quickly from a filnum identifier to a search of the correct FIS block in the attempt to find a matching FIS.

When a filnum is released, the FRBs are returned to allocatable file space. The FIS, however, remains intact, although it is flagged as released.

When a filnum is defined, the file manager first tries to find a corresponding FIS in core. If that fails, the file manager consults the FIS directory. The FIS directory search has one of three results:

1. If a matching but previously relreased FIS is found, it is reused after it is initialized to the attributes of the new filnum.

2. If a matching still-defined FIS is found, the filnum definition attempt is stopped since each filnum must have a unique definition.

3. If a matching FIS did not exist, a new FIS is created.

The file manager supports a multitude of file organizations and requests. However, Sort/Merge assumes that it is only dealing with sequential files, and Sort/Merge only uses four file manager requests: DEFFIL, STOSEQ, RTVSEQ, and RELFIL.

DEFFIL defines a file with user specification of filnum, maximum file record length, and logical unit.

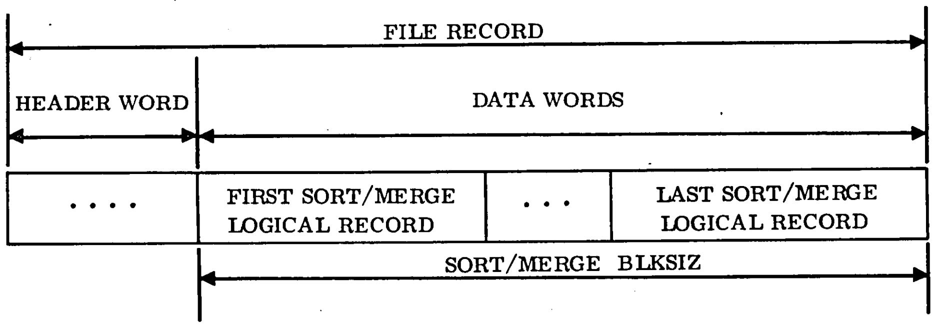
STOSEQ stores a user-specified buffer as a file record following the current last file record of a sequentially-organized user-specified filnum.

RTVSEQ can be used on any file organization and reads the next file record into a user-specified buffer from a user-specified filnum, regardless of file organization. However, Sort/Merge deblocks that file record so that is appears to contain no record pointers; i.e., as if the file is not indexed-linked.

RELFIL releases a user-specified filnum. The FIS remains on disk for possible future reuse, and the FRBs are returned to allocatable file space.

NOTE

Sort/Merge uses the master file combination of zero on all retrievals. File records in removed status are skipped without notice to the retrieving program. File-locked status is ignored during retrieval but is treated as an error during storage. Sort/Merge checks the status of each file manager request after it is completed.


FILE RECORD
HEADER WORD
DATA WORDS
. . . .
FIRST SORT/MERGE LOGICAL RECORD
. . .
LAST SORT/MERGE LOGICAL RECORD
SORT/MERGE BLKSIZ

0343

Figure J-3. Subdivision of File Records into Logical Records

INSTALLATION OF SORT/MERGE

The Sort/Merge module consists of six major programs: SMC, SMCMON, SMCEDT, SMCSRT, and SMCFMG. Each of these programs has its own deck, and each deck includes all the supporting routines necessary for each program to process its phase of the sorting, merging, or copying operation.

The installation takes place under the control of the job processor and the library editor. The decks used are all part of Sort/Merge version 1.0. The deck identifications are shown in table K-1.

The listing in table K-2 shows the skeleton of the installation tape for absolutizing the binary records of the Sort/Merge programs and for editing and installing them on the program library. The installation tape is entered via the input device (logical unit Ix) and loaded by typing *V,lu (after *BATCH). The first record read from the tape is a *JOB statement.

TABLE K-1. SORT/MERGE PROGRAM IDENTIFICATIONS

Program	Identification		Function
SMC	S01	SMC 1.0	Loader/Initializer
SMCMON	S02	SMC 1.0	Monitor
SMCEDT	S03	SMC 1.0	Editor for operator-input statements
SMCSRT	S04	SMC 1.0	Sort (initial)
SMCIMG	S05	SMC 1.0	Intermediate merging
SMCFMG	S06	SMC 1.0	Final merging and output
FLOTN	S08	SMC 1.0	Floating point
PARABN	S07	SMC 1.0	Interface to CYBER 18/ 1700 programs
COMNFP	S09	SMC 1.0	

TABLE K-2. INSTALLATION TAPE LISTING

Listing				Comments
0001	*JOB, SMCINS, SORT MERGE INSTALLATION			Call job processor
0002	*CTO,SORT/MERGE 1.0 INSTALLATION			
0003	*CTO,COPYRIGHT CONTROL DATA CORPORATION 1976			
0004	*CTO, REVISION DATE 14 JULY 1976			Call library editor
0005	*LIBEDT			
0006	*K,I6			Specify input on logical unit 6 and binary on logical unit 8
0007	*K,P8			
0008	*L,SMC			Add SMC
0009	SMC	DECK-ID S01 SMC 1.0	SUMMARY-108	
0010	*P,F			Generate absolute record for SMCMON in 96 word blocks
0011	SMCMON	DECK-ID S02 SMC 1.0	SUMMARY-108	
0012	FLOTN	DECK-ID S08 SMC 1.0	SUMMARY-108	
0013	PARABN	DECK-ID S07 SMC 1.0	SUMMARY-108	
0014	COMNFP	DECK-ID S09 SMC 1.0	SUMMARY-108	
0015	*T			Transfer files to disk
0016	*K,I8			Input from binary device
0017	*N,SMCMON,,,B			Enter update SMCMON in library

TABLE K-2. INSTALLATION TAPE LISTING (Continued)

Listing				Comments
0018	*K,I6			
0019	*P,F			Generate absolute record for SMCEDT
0020	SMCEDT	DECK-ID S03 SMC 1.0	SUMMARY-108	
0021	*T			
0022	*K,I8			Transfer updated SMCEDT to disk
0023	*N,SMCEDT,,,B			
0024	*K.I6			
0025	*P,F			Generate absolute record for SMCSRT
0026	SMCSRT	DECK-ID S04 SMC 1.0	SUMMARY-108	
0027	*T			
0028	*K,I8			Transfer updated SMCSRT to disk
0029	*N,SMCSRT,,,B			
0030	*K,I6			
0031	*P,F			Generate absolute record for SMCIMG
0032	SMCIMG	DECK-ID S05 SMC 1.0	SUMMARY-108	
0033	*T			
0034	*K,I8			Transfer updated SMCIMG to disk
0035	*N,SMCIMG,,,B			
0036	*K,I6			
0037	*P,F			Generate absolute record for SMCFMG
0038	SMCFMG	DECK-ID S06 SMC 1.0	SUMMARY-108	
0039	*T			
0040	*K,I8			Transfer updated SMCFMG to disk
0041	*N,SMCFMG,,,B			
0042	*K,I6			
0043	*Z			Terminate library processing for Sort/Merge
0044	*CTO, SORT/MERGE 1.0 IS INSTALLED			
0045	*U			
0046	*END			

DIAGNOSTIC MESSAGES

Table L-1 contains an alphabetical listing of all Sort/Merge diagnostic messages. In some cases parameters are an integral portion of the message. A full explanation of the parameters is found in Messages, section 4.

Included in table L-1 are run parameter (RP) statements and information-only (I) statements.

TABLE L-1. SORT/MERGE DIAGNOSTIC MESSAGES

Run Parameter (RP)/ Information Only (I)	Statement	Meaning
(I)	ABNORMAL ERROR = <n>	Miscellaneous errors
(I)	BLKSIZ/RECLTH ≠ <parameters>	The operator may direct the program to reread the file, to delete it, or to continue without operator interaction for the record size type of error.
(I)	COPY BEGINS	SMCSRT is loaded and is starting a copy run.
	DEFFIL REQIND = <parameters>	Bad user-defined output file status; run aborted
(I)	DELETES = <n>	The phase is ended and n records have been deleted
(I)	DONE = <n>	Number of records deblocked and processed
(I)	EDIT BEGINS	Edit is ready for operator inputs using prompting messages.
	EXPECTED <parameter> FOUND <character>	Editor did not find the type of parameter that should have been entered. Operator may be able to correct the error.
(I)	FINAL MERGE BEGINS	Sort-only run: SMCFMG is ready to start the final merging process.
	FREAD STATUS = <parameters>	Operator may direct the program to reread the file, to delete it, or to continue without operator interaction for format read errors.
	FWRITE STATUS = <parameters>	Operator may direct the program to rewrite the file, to abort the run, or to continue without operator interaction for format write errors.
(I)	FWAY = <n>	Sort-only run. Maximum number of wkbksz string input buffers that can be used during final merging
(I)	G = <n>	Sort-only run. Input has been checked and accepted; <n>-1 indicates largest number of records that can be sorted in core (if more than n records, Sort/Merge and mass storage are required).
(RP)	INFILE <n> = <parameters>	Operator should supply the input file parameters.

TABLE L-1. SORT/MERGE DIAGNOSTIC MESSAGES (Continued)

Run Parameter (RP)/ Information Only (I)	Statement	Meaning
(I)	INTERMEDIATE MERGE BEGINS	Sort-only run. SMCIMG is ready to start its merging process.
(I)	1STWAY = <n>	Sort-only run. SMCIMG has optimized merge strategy – <n> is the number of strings used for first merge.
(I)	INTERNAL SORT BEGINS	SMCSRT is loaded and is starting a sort-only run.
	INTERPHASE RECORD COUNTS DISAGREE	Number of output records does not equal the number of input sort records.
(I)	IWAY = <n>	Sort-only run. Maximum number of wkbksz input string buffers that can be used during intermediate merging
(RP)	KEYS = <parameters>	Operators should supply the file keys.

	LUN = <k>
	FILNUM = <f>
		The following messages concern file f from logical unit <k>.

(I)	MERGE-ONLY BEGINS	Merge-only run. SMCFMG is ready to start the final merging process.
	<n> = 12	Erroneous SMCFMG fixed table size; run aborted
(RP)	OUTFILE <n> = <parameters>	Operator should supply the output file parameters.
	OVERSIZE BLOCK <parameters>	Operator may direct the program to reread the file, to delete it, or to continue without operator interaction for this block size type error.
(I)	PASSED = <n>	The specified file composed of <n> records was either down or skipped.
(I)	READY FILE = <n>	The user should ready the file, or direct Sort/Merge to delete or bypass the file.
(I) (I)	RECORDS IN = <m> RECORDS OUT = <n>	RECORDS IN specifies the number of records sent from unblocking to processing; RECORDS OUT specifies the number of records for the converse. Unless a record is deleted or lost (hardware error), m is equal to n.
	RELFIL REQIND = <parameters>	The release file operation failed. Operator may direct program to retry the release or to continue with or without operator interaction for this type of error.
	RTXSEQ REQIND = <parameters>	Operator may direct the program to again retrieve the file, to delete it, or to continue without operator interaction for this type of retrieval error.
(RP)	RUN = <parameters>	Operator should supply the run parameters.

TABLE L-1. SORT/MERGE DIAGNOSTIC MESSAGES (Continued)

Run Parameter (RP)/ Information Only (I)	Statement	Meaning
(RP)	SEGMENT LIST ERROR	Sort-only run. Work file accountability lost; run aborted
	SEQ DIR ERROR	Sort-only run. Sequence directory read/write error; run aborted
	SEQUENCE ERROR	Latest record should have preceded previous record in key merging. Operator may direct program to delete the record or to continue with or without operator interaction for this type of error.
(I)	SEQUENCES = <n>	All completed logical records have been grouped in n sequences following a completed internal sort procedure.
(I)	SMC BEGINS	Program is beginning.
(I)	SMC ENDS	Program has been completed.
	STOSEQ REQIND = <parameters>	The operator may direct the program to again store the file, to abort the run, or to continue without operator interaction for this type of store error.
	TOO LITTLE CORE	Requested inputs cannot be processed in available core.
	TOO LITTLE DISK	Sort-only run. Inadequate disk space; run aborted
	TYPE-IN ERROR	Error in trying to interpret operator's command.
(I)	U = <n>	Sort-only run. SMCIMG has optimized merge strategy; n is the unit strings rating.

COMMENT SHEET

MANUAL TITLE CONTROL DATA® Sort/Merge Version 1.0 Reference Manual

PUBLICATION NO. 96769260 REVISION A

FROM NAME:

BUSINESS ADDRESS:

COMMENTS: This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number.

CUT ALONG LINE

STAPLE

STAPLE

FOLD

FIRST CLASS
PERMIT NO. 333

LA JOLLA, CA.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
PUBLICATIONS AND GRAPHICS DIVISION
4455 EASTGATE MALL
LA JOLLA, CALIFORNIA 92037

CUT ALONG LINE

FOLD

STAPLE

STAPLE