60497500

CONTROL DATA CORPORATION

# SORT/MERGE VERSIONS 4 AND 1 REFERENCE MANUAL

**CDC® OPERATING SYSTEMS:**
**NOS 1**
**NOS/BE 1**
**SCOPE 2**

60497500

CONTROL DATA CORPORATION

# SORT/MERGE VERSIONS 4 AND 1 REFERENCE MANUAL

**CDC® OPERATING SYSTEMS:**
**NOS 1**
**NOS/BE 1**
**SCOPE 2**

| REVISION RECORD | |
|---|---|
| **REVISION** | **DESCRIPTION** |
| A | Manual released. |
| (11-01-75) | |
| B | This revision documents version 4.4 of Sort/Merge; changes include feature CP156, Performance |
| (03-01-76) | Enhancements. |
| C | This revision documents version 4.5 of Sort/Merge. Changes include feature CP173, use of the |
| (11-15-76) | Common Memory Manager for space allocation. This manual is at PSR level 439. |
| D | This revision documents feature CP173I at PSR level 446. Changes include key comparison |
| (03-01-77) | technique for macro sorts. |
| E | This revision documents version 4.6 of Sort/Merge at PSR level 472. Changes include feature |
| (04-15-78) | 162, interface with BAM 1.5. |
| F | This revision documents the Sort/Merge interface with FORTRAN 5 at PSR level 528. |
| (12-02-80) | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| Publication No. 60497500 | |

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

Address comments concerning this manual to:

**CONTROL DATA CORPORATION**
*Publications and Graphics Division*
**215 MOFFETT PARK DRIVE**
**SUNNYVALE, CALIFORNIA 94086**

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 |
|---|---|
| Cover | - |
| Title Page | - |
| ii | F |
| iii/iv | F |
| v | F |
| vi | F |
| vii | F |
| 1-1 thru 1-3 | F |
| 2-1 | B |
| 2-2 | F |
| 2-3 | C |
| 2-4 | E |
| 2-5 | F |
| 2-6 | C |
| 3-1 | F |
| 3-2 | D |
| 3-3 thru 3-5 | A |
| 3-6 | E |
| 3-7 | E |
| 4-1 | F |
| 4-2 | F |
| 4-3 | C |
| 4-4 | C |
| 4-5 | F |
| 4-6 | F |
| 4-7 | E |
| 4-8 | E |
| 4-9 | F |
| 4-10 | E |
| 4-11 | A |
| 4-12 | B |
| 4-13 | A |
| 4-14 | A |
| 4-15 | F |
| 4-16 | F |
| 4-17 thru 4-23 | A |
| 4-24 thru 4-27 | F |
| 5-1 thru 5-3 | E |
| 5-4 | C |
| 5-5 | C |
| 5-6 | B |
| 5-7 | E |
| 5-8 | B |
| 5-9 | A |
| 5-10 | A |
| 5-11 | F |
| 5-12 | E |
| 5-13 thru 5-15 | D |
| 6-1 | F |
| 6-2 | E |
| 6-3 thru 6-10 | F |
| 6-10.1 | F |
| 6-10.2 | F |
| 6-11 | A |

| Page | Revision |
|---|---|
| 6-12 | A |
| 6-13 | D |
| 6-14 | A |
| 6-15 | D |
| 6-16 | A |
| 7-1 thru 7-15 | F |
| A-1 thru A-4 | F |
| B-1 thru B-4 | E |
| B-5 thru B-7 | F |
| B-8 | E |
| B-9 | E |
| B-10 | F |
| B-11 thru B-13 | E |
| C-1 | F |
| C-2 | E |
| C-3 | A |
| D-1 thru D-11 | E |
| D-12 thru D-15 | A |
| E-1 thru E-15 | A |
| F-1 thru F-3 | B |
| G-1 thru G-5 | E |
| H-1 | F |
| H-2 | F |
| Index-1 thru -6 | F |
| Comment Sheet | F |
| Mailer | - |
| Back Cover | - |

# PREFACE

Sort/Merge high-speed record processing facilities are available through the use of control statements and directives. User programs can call Sort/Merge with the COMPASS assembly language macros, the FORTRAN interface routine calls, or through the COBOL language.

The user is assumed to be familiar with the operating system on which Sort/Merge is to be run as well as with the calling language.

The Sort/Merge system is available under the following operating systems:

Sort/Merge Version 4.6 operates under NOS 1 for the CONTROL DATA® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74 and 6000 Series Computer Systems.

Sort/Merge Version 4.6 operates under NOS/BE for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems.

Sort/Merge Version 1.0 operates under SCOPE 2.1 for the CONTROL DATA CYBER 170 Model 176, CYBER 70 Model 76 and 7600 Computer Systems.

Documents of primary interest (in alphabetic order) to Sort/Merge users include the following:

| Publication | Publication Number |
|---|---|
| COBOL Version 5 Reference Manual | 60497100 |
| COMPASS Version 3 Reference Manual | 60492600 |
| CYBER Record Manager Basic Access Methods 1.5 Reference Manual | 60495700 |
| FORM Version 1 Reference Manual | 60466200 |
| FORTRAN Extended Version 4 Reference Manual | 60497800 |
| FORTRAN Version 5 Reference Manual | 60481300 |
| NOS Version 1 Reference Manual, Volume 1 of 2 | 60435400 |
| NOS/BE Version 1 Reference Manual | 60493800 |
| SCOPE Version 2 Reference Manual | 60342600 |
| SCOPE Version 2 Record Manager Reference Manual | 60454690 |
| Sort/Merge Version 4 and 1 Instant | 60497600 |
| Sort/Merge Version 4 Users Guide | 60482900 |

The following documents (in alphabetic order) are of secondary interest to users of Sort/Merge:

| Publication | Publication Number |
|---|---|
| CYBER Record Manager Basic Access Methods 1.5 Users Guide | 60495800 |
| NOS Version 1 Manual Abstracts | 84000420 |
| NOS/BE Version 1 Manual Abstracts | 84000470 |
| Software Publications Release History | 60481000 |

The NOS manual abstracts and the NOS/BE manual abstracts are instant-sized manuals containing brief descriptions of the contents and intended audiences of all NOS and NOS product set manuals, and NOS/BE and NOS/BE product set manuals, respectively. The abstracts manuals can be useful in determining which manuals are of greatest interest to a particular user. The Software Publications Release History serves as a guide in determining which revision level of software documentation corresponds to the Programming System Report (PSR) level of installed site software.

CDC manuals can be ordered from Control Data Corporation Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

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 parameters.

# CONTENTS

# INTRODUCTION 1

The primary function of Sort/Merge is the manipulation and rearrangement of records into a prescribed order, according to a user specification.

CDC offers guidelines for the use of the software described in this manual. These guidelines appear in appendix H. Before using the software described in this manual, the reader is strongly urged to review the content of this appendix. The guidelines recommend use of this software in a manner that reduces the effort required to migrate application programs to future hardware or software systems.

## COMPARISON OF SORT/MERGE 1 AND SORT/MERGE 4

Some of the material in this manual applies only to Sort/Merge Version 1; other material applies only to Sort/Merge Version 4. The difference is indicated as follows:

Material shaded in this manner is applicable only to Sort/Merge Version 1, supported under SCOPE 2.

Material shaded in this manner is applicable only to Sort/Merge Version 4, supported under NOS and NOS/BE.

Material presented without shading is appliable to both Version 1 and Version 4 of sort/Merge.

### SORT/MERGE 1

Sort/Merge resides in the library of the SCOPE 2 operating system. Input for this version consists of the following:

Control statements and directives (Sort/Merge Version 3 directives are accepted when 6C is specified on the SORTMRG control statement; see section 4)

Macro calls

Input files residing on mass storage and/or magnetic tape.

Intermediate storage areas or intermediate merge files reside on mass storage only.

Output is directed to mass storage or magnetic tape.

### SORT/MERGE 4

Sort/Merge resides in the libraries of the NOS and NOS/BE operating systems. Input for this version consists of the following:

Control statements and directives (Sort/Merge Version 3 directives are accepted, with the limits specified in appendix C, when 6C is specified on the SORTMRG control statement).

Macro calls.

Input files residing on mass storage and/or magnetic tape.

Intermediate storage areas or intermediate merge files can reside on either mass storage or magnetic tape.

Output is directed to mass storage or magnetic tape.

## SORT/MERGE CAPABILITIES

The modular construction of the Sort/Merge program enhances the record processing options by providing the following capabilities:

| | |
|---|---|
| Sort-only | Sorts records from one or more input files into an order specified by the user. |
| Merge-only | Combines from 2 to 100 presorted input files into 1 output file. |
| Sort and Merge | Sorts from 1 to 100 input files into an order or sequence specified by the user and merges the output with from 1 to 100 presorted files. No more than 100 files can be used in both phases combined. |

## RECORDS AND FILES

An understanding of Record Manager requirements for records and files in terms of structure and format as presented in appendix D is essential for proper application of the Sort/Merge program. More detail may be found in the Record Manager reference manuals.

## CHECKPOINT/RESTART

A Sort/Merge job can be terminated at any time during program execution as a result of machine malfunction, as well as operator or program error. Sorted and merged records would be lost if abnormal termination occurred during a final pass, necessitating that the job be resubmitted from the beginning.

The checkpoint/restart system facility stores on magnetic tape the total environment of a job abnormally terminated so the job can be restarted from a checkpoint, rather than from its beginning. Total environment includes local files associated with the control point of the job. For mass storage files, the complete file is captured as well as the relative position within the file. For magnetic tape files, only the relative position on the tape is captured, so the tape can be properly repositioned during restart.

Check/restart cannot handle:

- Roll-out jobs
- Random files
- Multi-file reels
- Tape sorts

To use the check point/restart facility, the checkpoint/restart option must be specified on the OPTIONS directive or macro for execution after a user specified number of records have been read from the input file or written to the output file or on the end of each reel of input or output.

Each time a checkpoint dump is taken, a file is written containing all information needed to restart the job at that point. Each checkpoint dump is numbered automatically in ascending order by the system. Several checkpoint dumps should be taken during a long Sort/Merge job so the user can return to any one of them to restart the job.

Consult the NOS or NOS/BE reference manual for information about requesting a file to be used as a checkpoint file, and for information about restarting a checkpoint job.

## SORT/MERGE DAYFILE MESSAGES

Upon successful completion of a Sort/Merge run, tallies are printed in the dayfile, providing statistics for that run. The following messages are output, indicating how many records were inserted, deleted, sorted, and output.

```
** INSERTIONS DURING INPUT     ********** n
** DELETIONS DURING INPUT      ********** n
** TOTAL RECORDS SORTED        ********** n
** INSERTIONS DURING OUTPUT    ********** n
** DELETIONS DURING OUTPUT     ********** n
** TOTAL RECORDS OUTPUT        ********** n
```

The following message is printed if intermediate merge files are used:

```
** MERGE ORDER USED WAS        ********** n
```

The following message is printed when records occur out of sequence on a merge run without the VERIFY option:

```
** RECORDS OUT OF SEQUENCE    **********n
```

After all tally messages, one of the following messages is printed, depending on the kind of run:

```
** END SORT RUN

** END MERGE RUN
```

# SORT KEYS 2

A sort key is a field of information within each record in a sort or merge input file that is used by Sort/Merge to determine the order in which records will be written to the output file.

An example of a record with sort keys is a typical record in a personnel file. Each such record might contain the name, age, department number, salary, and employment date of an individual employee.

This file could be sorted by any one of these fields or by a combination of fields. For example, the file could be sorted by the name field, or it could be sorted by age (as the major key) and salary (as the minor key). In the latter case, if two or more employees were the same age, their records would appear on the output file in order by salary.

## SORT KEY DESCRIPTION

Every field to be used as a sort key must be described by the user. The user of the directive version describes fields with the FIELD directive and the KEY directive (section 4); the user of the macro version describes fields with the KEY macro (section 5).

Sort key descriptions include the following information:

Key length

Starting location of key within record

Type of data found in key field

Sort order

Collating sequence to be used (for character keys only)

### KEY LENGTH AND POSITION

Key field length is specified on the FIELD directive or KEY macro as the number of bytes and bits in the field; the default byte is a 6-bit character. The length of any single key must not exceed minimum record length for any file.

Starting position of a sort key field may be anywhere within the record, but it must be the same for all records of all files to be sorted or merged. Character coded keys that span a word boundary must begin on a character boundary. Keys in variable length records must lie wholly within the fixed length portion of the record; if the last character of any key is not within the minimum record length, sort order is undefined. If more than one sort key field exists, fields may overlap.

## KEY TYPE

Data in sort key fields can be of any of the following types:

LOGICAL — Unsigned binary integers of any length (any number of bits); they are assumed to be non-negative, and are sorted by magnitude. Under certain conditions, described later in this section, data of other types can be sorted more efficiently when it is specified as logical type.

INTEGER — 60-bit integers; they can start anywhere within the record (not necessarily on a character or word boundary), and are sorted by numeric value. Integer data includes numbers of FORTRAN Extended INTEGER type when written by a binary write. Under certain conditions, described later in this section, data of other types can be sorted more efficiently when it is specified as type INTEGER.

The range of INTEGER keys must not exceed $2^{59}-1$($2^{59}-1$ = 576 460 752 303 423 487). If two sort keys differ by more than $2^{59}-1$, improper sorting may take place with no diagnostic issued. -0 is considered equal to +0.

FLOAT — 60-bit normalized or unnormalized floating point numbers; they can start at any bit position within the record, and are sorted by numeric value. When written by a binary write, ALGOL numbers of ≠REAL≠ or ≠INTEGER≠ type, FORTRAN Extended numbers of REAL type, and COBOL COMPUTATIONAL-1 and COMPUTATIONAL-2 numbers are all valid FLOAT keys.

The full range of floating point numbers is permitted in FLOAT keys. This range is:

$$-10^{322} \leqslant x \leqslant -10^{-293} \quad \text{or} \quad x = -0\ (=+0) \quad \text{or} \quad 10^{-293} \leqslant x \leqslant 10^{322}$$

Infinite numbers, positive or negative, and indefinite numbers are also allowed. A positive indefinite number is treated as though it had a value near 140 737 488 355 327.5, which cannot be represented exactly as a floating point number, and a negative indefinite number is treated as though it had a value near -140 737 488 355 327.5, which cannot be represented exactly as a floating point number.

INTBCD — Key fields written in internal BCD character code (appendix A); they can be any integral number of characters in length (any integral multiple of 6 bits). Some tapes written by CDC 3000 Series computers are internal BCD, but the code is not standard in the operating systems which support Sort/Merge. INTBCD keys are sorted according to the specified collating sequence.

DISPLAY — Key fields written in display code (appendix A); they can be any integral number of characters in length (any integral multiple of 6 bits). DISPLAY keys are sorted according to the specified collating sequence. The special case of signed numeric data is discussed below.

## COLLATING SEQUENCE

Collating sequence applies only to character data, not to numeric data. Unless the key is specified as INTBCD or DISPLAY, a collating sequence is not needed. The collating sequence can be any of the following:

| | |
|---|---|
| ASCII6 | The American Standard Code for Information Interchange collating sequence (appendix A) |
| COBOL6 | A CDC collating sequence (appendix A) |
| DISPLAY | The CDC display code collating sequence (appendix A) |
| INTBCD | Internal BCD collating sequence (appendix A) |

In addition, the user can specify a collating sequence with the SEQUENCE and EQUATE directives or macros. The collating sequence chosen need not correspond to the character set used in coding the data; collating sequence is independent of character set. The character set determines the translation that will take place from a 6-bit binary value occupying a character position to one of the letters, digits, and special characters available as graphics. The collating sequence, on the other hand, determines the precedence given to each character already translated, when the key is sorted.

For example, a file written using the internal BCD character set can be sorted by any of the collating sequences. Three records in a file have key fields whose contents are as follows:

| Record | Key (octal) |
|---|---|
| Record 1 | 0707030303 |
| Record 2 | 4331626333 |
| Record 3 | 1212121313 |

When INTBCD is specified as the character set and INTBCD as the collating sequence, records are written to the output file in the following order:

| Record | Key (octal) | Key (INTBCD) | Collating Position of Characters in Key (INTBCD) |
|---|---|---|---|
| Record 1 | 0707030303 | 77333 | 07,07,03,03,03 |
| Record 3 | 1212121313 | :::== | 12,12,12,13,13 |
| Record 2 | 4331626333 | LIST. | 43,31,62,63,33 |

The same records sorted according to INTBCD character set and DISPLAY collating sequence would be written in the following order:

| Record | Key (octal) | Key (INTBCD) | Collating Position of Characters in Key (DISPLAY) |
|---|---|---|---|
| Record 3 | 1212121313 | :::== | 00,00,00,54,54 |
| Record 2 | 4331626333 | LIST. | 14,11,23,24,57 |
| Record 1 | 0707030303 | 77333 | 42,42,36,36,36 |

If these records were sorted again according to the DISPLAY character set and the DISPLAY collating sequence, they would be written in the following order:

| Record | Key (octal) | Key (DISPLAY) | Collating Position of Characters in Key (DISPLAY) |
|---|---|---|---|
| Record 1 | 0707030303 | GGCCC | 07,07,03,03,03 |
| Record 3 | 1212121313 | JJJKK | 12,12,12,13,13 |
| Record 2 | 4331626333 | 8Y]%0 | 43,31,62,63,33 |

## SORT ORDER

The order for sorting keys can be specified as ascending or descending.

Ascending — Numeric keys are sorted so that the record having the key with the highest value is written last on the output file. Character keys are sorted according to the collating sequence specified.

Descending — Numeric keys are sorted so that the record having the key with the lowest value is written last on the output file. Character keys are sorted according to the collating sequence specified, taken in reverse.

## SIGNED NUMERIC DATA

Signed numeric data is integer data stored internally in display code, rather than in 60-bit integer format. Signed numeric data is specified either by the word SIGN on the sort or merge key descriptor or by ≠SIGN≠ or ≠SEPARATE≠ on the SMKEY call. Embedded blanks cannot appear within a signed numeric data field. In contrast to data specified as type DISPLAY that sorts according to display code values, signed numeric data sorts according to the magnitude and value of the integer the display code represents.

The sign of the integer can be specified as any of the following:

Sign overpunch representation of the last digit in the field. The display code value of the low order digit is equivalent to the key punch code resulting when a digit is overstruck with a + or - sign, as shown in the list below. The sign of the overpunch specifies the sign of the integer. The overpunch can be omitted implying a positive integer.

Sign overpunch representation of the first digit in the field. The display code value of the high order digit is equivalent to the key punch code resulting when a digit is overstruck with a + or - sign, as shown in the list below. The sign of the overpunch digit specifies the sign of the integer.

SEPARATE specifies that the positive and negative signs are the characters + and - respectively. If SEPARATE is specified, the + or - character can be the first or last character of the field.

When the character is shown in printed output or is received in input as a card image, the signed digit appears as specified in the third column of the sign overpunch code list. When the item is to be received as input from a card, the signed digit must be punched as specified in the second column of the list. Positive or unsigned integers can be mixed with integers represented by a sign overpunch in an input file. The negative sign is represented by a - overpunch in row 11; the positive sign is represented by the absence of an overpunch or the presence of a + overpunch in row 12.

Sign overpunch codes are:

| Sign and Digit | Hollerith Punch | Output Representation |
|---|---|---|
| +9 | 12-9 | I |
| +8 | 12-8 | H |
| +7 | 12-7 | G |
| +6 | 12-6 | F |
| +5 | 12-5 | E |
| +4 | 12-4 | D |
| +3 | 12-3 | C |
| +2 | 12-2 | B |
| +1 | 12-1 | A |
| +0 | 12-0 | < |
| -0 | 11-0 | v |
| -1 | 11-1 | J |
| -2 | 11-2 | K |
| -3 | 11-3 | L |
| -4 | 11-4 | M |
| -5 | 11-5 | N |
| -6 | 11-6 | O |
| -7 | 11-7 | P |
| -8 | 11-8 | Q |
| -9 | 11-9 | R |

Positive integers can be mixed with integers represented by a sign overpunch in an input file.

## MULTIPLE KEYS

At least one sort key must be specified for each sort or merge run; as many as 100 keys can be specified. When multiple keys are specified, they can differ as to type, collating sequence, and sort order.

If more than one key is specified, the order (left to right) in which they appear in the KEY directives or macros determines their precedence for sorting. The key fields on the first KEY directive or macro, from left to right, are compared first; then those on subsequent KEY directives or macros are compared until a field is found in which the two records have unequal values. Then, the records are sequenced appropriately. If all the key fields for the two records have the same value, they are sequenced arbitrarily, unless the RETAIN option has been selected causing records with identical keys to be written in the order they were read.

## ALTERNATE SPECIFICATION OF KEY TYPES

Because Sort/Merge processes integer and logical keys more quickly than floating point or character coded keys, it is advantageous for the user to specify key type as INTEGER or LOGICAL whenever such specification will result in a correct sorting order.

For example, floating point keys will sort properly as integers if either of the following two conditions is true:

1. All keys in the file are normalized. FORTRAN Extended numbers of REAL type, ALGOL numbers of ≠REAL≠ or ≠INTEGER≠ type, and COBOL COMPUTATIONAL-2 numbers are all normalized floating point numbers. In addition, all keys in the file must lie within one of the following ranges:

   $-10^{14} \leqslant X \leqslant -10^{-293}$ or $X=0$ or $10^{-293} \leqslant X \leqslant 10^{14}$

   $X=0$ or $10^{-293} \leqslant X \leqslant 10^{322}$ (non-negative values)

   $-10^{322} \leqslant X \leqslant -10^{-293}$ or $X=0$ (non-positive values)

2. All keys in the file have the same exponent. This condition is true for COBOL COMPUTATIONAL-1 numbers, when they contain 14 or fewer decimal digits.

Certain kinds of character keys can be sorted more efficiently as integers. The most common example is a key consisting of a COBOL item defined by the clause

PICTURE 9(n) SYNCHRONIZED RIGHT USAGE IS DISPLAY.

where $1 \leqslant n \leqslant 9$. Since an item of this description always occupies a full word, and the leftmost digit is always a display code 0 (33 octal), sorting keys of this description as integers results in a proper sort by magnitude.

For character keys, when the character set and the collating sequence have the same name (INTBCD character set and INTBCD collating sequence, or DISPLAY character set and DISPLAY collating sequence), Sort/Merge sorts them as logical keys, since logical keys sort faster than character keys.

Alternate key type specification also can be used to sort fields that do not conform to the characteristics of any data types recognized by Sort/Merge. For example, if a field whose length is 64 bits contains positive and negative integer values in ones complement form, it cannot be sorted as INTEGER type since fields of INTEGER type must be 60 bits long; nor does it sort properly as LOGICAL type, since fields of LOGICAL type are assumed to be unsigned. If the field is divided into two subfields, however, one consisting of the sign bit and the other of the magnitude of the integers, the field can be sorted by the following specifications:

```
FIELD,SIGN(.1,.1,LOGICAL)
FIELD,MAGNITUDE(.2,.63,LOGICAL)
KEY,SIGN(D),MAGNITUDE(A)
```

In this case, unlike INTEGER keys, keys equal to -0 are less than keys equal to +0.

# OWNCODE ROUTINES 3

Owncode routines represent closed subroutines written by the user as COMPASS language programs. Although owncode routines are not required for Sort/Merge execution, they provide the capability for the user to insert, substitute, modify, or delete input and output records. Refer to appendix H for recommendations on the use of COMPASS owncode routines.

All owncode routines specified by the OWNCODE directive must be assembled previously in relocatable binary form and placed in the file "INPUT" or on an alternative source as specified on the SORTMRG control statement (see section 4).

Owncode routines can be specified with the OWNCODE macro call. The routines are assembled in the program calling Sort/Merge or assembled and referenced in a program occupying memory at the same time as the program calling Sort/Merge.

The following program exits can occur during Sort/Merge processing.

Exit 1 After reading each record from any sort input file while the record is still in the intermediate input area, or during a search for a new record when no input file is specified. (This exit is not allowed in a merge-only run.)

Exit 2 After reading the last record from each sort input file, but before file disposal action is initiated. (This exit is not allowed in a merge-only run.)

Exit 3 After each record is selected for output but before the record is moved into the final output area, or when no output file has been specified.

Exit 4 After the last record has been moved into the final output area, but before file disposal action is initiated.

Exit 5 When two records with identical sort keys are encountered.

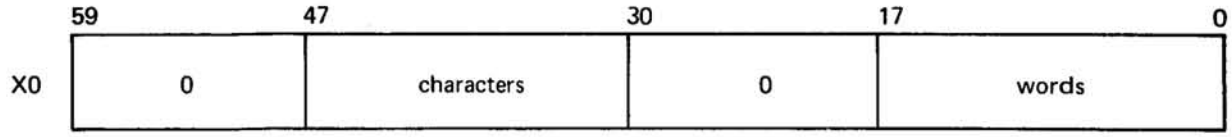
Exit 6 Each time a file with a user specified nonstandard label is to be checked, while the label is being read during input or written during output.

Upon entry to all owncode exits, register A2 contains the address of the current data record and register X0 contains the record length. In addition, during entry into owncode exit 5, registers A3 and X4 are used for the address and length of the second record of a comparison involving identical sort key data.

Exits 1 and 2 owncode routines are not allowed in a merge-only run. Their purpose can be fulfilled by using Exits 3 and 4 owncode routines in a merge-only run, or by using Exits 1 and 2 owncode routines in a sort run with supplementary merge files. An Exit 1 or 2 owncode routine specified in a merge-only run is ignored except that a non-fatal diagnostic message is issued.

When Sort/Merge transfers control to the user routine, the upper 30 bits of the X0 register contain the record length in characters of the current record, and the lower 30 bits contain the length in words of this record.

**Example**




Transfer to owncode routines is accomplished with a return jump (RJ) instruction which fills the entry point of the owncode routine with a return to the Sort/Merge program. To re-establish Sort/Merge control, the user must jump to the entry point of the owncode routine. The user can request specific processing action by altering the return address in the entry point of the owncode routine.

## EXIT 1: PROCESSING OF INPUT RECORDS

Exit 1 is taken after reading each record from the input file while the record is still in the intermediate input area, or during a search for a new record when no input file is specified. The user must specify processing action with a jump to one of the following addresses. Exit 1 is not allowed in a merge-only run.

Normal return address — Sort/Merge accepts the record whose address is in register A2 and whose length is in register X0. Before returning to this address the user can :

- Retain current record without modification; A2 and X0 are unchanged.
- Modify record in central memory without changing its address in A2. X0 should contain the correct length of the modified record; the record length must not exceed the maximum for the run.
- Replace the current record by changing the contents of A2 and X0 to reflect the address and length of a replacement record.
- Provide an input record when no input files have been specified. The address and length of the record are put in A2 and X0.

Normal return address+1 — Sort/Merge deletes current input record.

Normal return address +2 — Sort/Merge inserts a user specified record after current record is read. Address and length of record to be inserted are returned by the user in the A2 and X0 registers. Sort/Merge continues to return control to the Exit 2 routine until a transfer is made by the user to a different return address. The address and length of the original input record are put in registers A2 and X0 each time Sort/Merge returns control.

Normal return address+3 — Sort/Merge terminates record input from current input file and proceeds to next input file. If current file is the last, Sort/Merge proceeds to Exit 2, if specified, or to the merge phase. Current record is not processed.

## EXIT 2: PROCESSING OF INPUT FILES

Exit 2 is taken after the last record is read from each file but before final file handling action is initiated. The user can specify a jump to either of the following addresses. Exit 2 is not allowed in a merge-only run.

| | |
|---|---|
| Normal return address | Sort/Merge continues normal record processing by proceeding to the next input file or to the next phase of Sort/Merge. |
| Normal return address+1 | Sort/Merge inserts a user specified record after the last record read. Address and length of record are put by the user in registers A2 and X0. Control returns to Exit 2 until a normal return is executed. |

## EXIT 3: PROCESSING OF OUTPUT RECORDS

Exit 3 is taken before each record is moved into the final output area or when no output file has been specified. The user can specify one of the following addresses:

| | |
|---|---|
| Normal return address | Sort/Merge writes the record whose address is in register A2 and whose length is in register X0. Before returning to this address the user may:<br><br>Retain current output record without modification; A2 and X0 are unchanged.<br><br>Modify record in central memory without changing address in A2. X0 should contain the correct length of the modified record; the record length must not exceed the maximum for the run.<br><br>Replace the current record by changing the contents of A2 and X0 to reflect the address and length of a replacement record. |
| Normal return address+1 | Sort/Merge deletes the output record. If no output file has been specified, all output records must be deleted. |
| Normal return address+2 | Sort/Merge inserts a user specified record before current record is written. Address and length of record to be inserted are in A2 and X0 registers. Sort/Merge continues to return control to the Exit 3 routine until a return is made to a different address. The address and length of the original output record are put in registers A2 and X0 each time Sort/Merge returns control. |
| Normal return address+3 | Sort/Merge terminates record output to the current output file and proceeds with Exit 4 execution if specified. Current record is not processed. |

## EXIT 4: PROCESSING OF OUTPUT FILES

Exit 4 is taken after the last record is moved into the final output area but before final file handling action is initiated. The user can specify either of the following addresses:

| | |
|---|---|
| Normal return address | Sort/Merge continues normal record processing by proceeding to the end-of-file procedures for the output file. |
| Normal return address+1 | Sort/Merge inserts a user specified record after the last record is written. Address and length of record are in registers A2 and X0. Control returns to Exit 4 until a normal return is executed. |

## EXIT 5: PROCESSING OF DUPLICATE KEYS

Exit 5 is taken when two records with identical sort keys are encountered. One of the following addresses must be specified if this exit is selected.

| | |
|---|---|
| Normal return address | Sort/Merge accepts two records: one with address in register A2 and length in register X0, and the other with address in register A3 and length in register X4. Before returning to this address the user may:<br><br>Retain both current records without modification; address and length registers of current records are unchanged.<br><br>Modify either record, or both records, in central memory without changing the addresses in A2 and A3. The corresponding X registers (X0 and X4) should contain the correct lengths of the modified records; the record length must not exceed the maximum for the run.<br><br>Replace either record, or both records, by changing the contents of the address and length registers to reflect addresses and lengths of replacement records. |
| Normal return address+1 | Sort/Merge deletes one of two records with identical sort keys. The user provides address and length of the record to be retained in registers A2 and X0. |

## EXIT 6: PROCESSING OF NONSTANDARD LABELS

Exit 6 is used for checking or verifying user specified nonstandard labels while the labels are being read during input or written during output. This routine is called for all nonstandard labels on files whose FILE control statement specifies LT=NS, ULP≠NO.

### INPUT

During record input, registers A2 and X0 contain the address and length of a nonstandard label record read by Sort/Merge.

| | |
|---|---|
| Normal return address | User returns to this address when nonstandard label checking is complete. |

| | |
|---|---|
| Normal return address+1 | User returns to this address if an additional nonstandard label is to be read and program control is to be returned to the user. Control returns to Exit 6 until a normal return is executed. |

## OUTPUT

During record output registers A2 and X0 contain the address and length of the label record submitted by the user.

| | |
|---|---|
| Normal return address | User returns to this address when the last, or only, nonstandard label is to be written to the output file. The address and length of this label are in registers A2 and X0. |
| Normal return address+1 | User returns to this address if an additional nonstandard label is to be passed to Sort/Merge for writing on the output file. Sort/Merge enters the user routine again through Exit 6 after the current label is written on the output file. |

## OWNCODE SUMMARY

| Processing Action | Exit 1 | Exit 2 | Exit 3 | Exit 4 | Exit 5 | Exit 6 |
|---|---|---|---|---|---|---|
| Substitute record | NR | | NR | | NR† | |
| Insert record | NR+2 | NR+1 | NR+2 | NR+1 | | |
| Delete record | NR+1 | | NR+1 | | NR+1†† | |
| Verify label record during read | | | | | | NR+1 |
| Verify last label record during read | | | | | | NR |
| Supply a label record during write | | | | | | NR+1 |
| Supply last label record during write | | | | | | NR |
| Terminate file††† | NR+3 | | NR+3 | | | |
| Normal record processing | NR | NR | NR | NR | NR | |

†Both records, designated by the A2 and X0 and the A3 and X4 registers, can be substituted with new address and length specifications.

††Record designated by A3 and X4 registers is deleted.

†††Current record designated by A2 and X0 registers is not included in Sort or Merge processing.

## JOB EXAMPLE

The following Sort/Merge job example illustrates the use of an exit 1 owncode routine to supply input records from a multifile set to a directive sort. The example consists of a COMPASS subroutine, Sort/Merge directives, and job control statements.

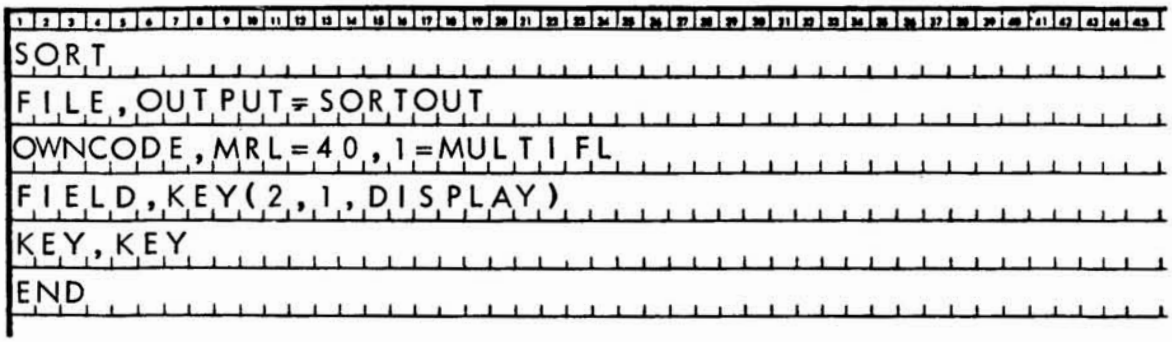
### OWNCODE ROUTINE

Through the COMPASS subroutine shown below, records are retrieved from the input file set SORTIN. When all records have been obtained, the file set is closed and a branch is taken to normal return address + 3. This branch indicates that Sort/Merge is to terminate record input and proceed to the merge phase.

```
LOCATION  OPERATION  VARIABLE SUBFIELDS      COMMENTS
          IDENT      MULTIFL
          SST
          ENTRY      MULTIFL
MULTIFL   BSS        1                       DEFINE ENTRY/EXIT WORD
          SA1        OPENED                  CHECK AND SET OPEN FLAG
          NZ         X1,MULTIFL1             .
          MX6        1                       .
          SA6        A1                      .
          OPENM      SORTIN,INPUT,R          OPEN FILE FOR INPUT WITH
                                               REWIND
MULTIFL1  GET        SORTIN                  READ RECORD
          CHECK      SORTIN                  .
          FETCH      SORTIN,RL,X0            SET X0 TO RECORD LENGTH
          LX0        30                      SHIFT REC LENGTH TO UPPER
                                               30 BITS
          SA2        REC                     SET A2 TO WSA
          EQ         MULTIFL                 EXIT, SUPPLYING RECORD
MULTIFL2  BSS        1                       DEFINE ENTRY/EXIT WORD
          FETCH      SORTIN,FP,X1            CHECK FOR END-OF-INFORMATION
          SX1        X1-≡EOI≡                .
          NZ         X1,MULTIFL1             .
          CLOSEM     SORTIN                  CLOSE FILE
          CHECK      SORTIN                  .
          SA1        MULTIFL                 BRANCH TO NRA + 3
          LX1        30                      .
          SB7        X1                      .
          JP         B7+3                    .
OPENED    DATA       0                       OPEN FLAG
L         EQU        1002B                   LENGTH OF BUFFER
SORTIN    FILE       BT=C,RT=Z,FL=40,BFS=L,  SORTIN FILE DESCRIPTION
                      FWB=BUF,WSA=REC,        .
                      DX=MULTIFL2             .
BUF       BSS        L                       DEFINE FWB
REC       BSS        4                       DEFINE WSA
          END
```

### SORT/MERGE DIRECTIVES

The Sort/Merge directives used for this example include an OWNCODE that specifies the exit number 1 and the entry point name MULTIFL for the owncode routine. The maximum record length of the input records is defined in this directive as 40 characters. Because input records are being supplied by the exit 1 owncode routine, no input file is specified in the FILE directive.




### JOB CONTROL STATEMENTS

The COMPASS control statement causes the owncode routine to be assembled in relocatable binary form. A FILE statement is included to describe the output file SORTOUT. A FILE statement for the input file set SORTIN is not needed because a FILE macro is specified within the owncode routine. The COPYBR statements copy three records from the system INPUT file onto three separate files. The files are then copied onto one multifile set by the COPYBF statements. Sort/Merge processing begins with the SORTMRG control statement. The OWN parameter indicates that the owncode binaries are located on the file LGO.

```
COMPASS(S=IOTEXT,A)
FILE(SORTOUT,BT=C,RT=Z,FL=40)
COPYBR(INPUT,FILE1)
COPYBR(INPUT,FILE2)
COPYBR(INPUT,FILE3)
REWIND(FILE1,FILE2,FILE3)
COPYBF(FILE1,SORTIN)
COPYBF(FILE2,SORTIN)
COPYBF(FILE3,SORTIN)
SORTMRG(OWN)
```

# DIRECTIVE SORT/MERGE PROCESSING 4

## CONTROL STATEMENT REQUIREMENTS

Processing of Sort/Merge directives is accomplished through the SORTMRG control statement. The directives are assumed to be the next unexecuted section on the INPUT file unless an alternative source is specified on the SORTMRG statement. Sample deck structures for various types of Sort/Merge runs are illustrated at the end of this section.

Because Sort/Merge performs all input and output through Record Manager, a Record Manager FILE control statement must be provided for every input or output file to be processed by a directive sort or merge. This control statement specifies the record and block structure of the file as well as file handling options. The FILE control statement is described in appendix D.

In Sort/Merge Version 1, if the file INPUT is used as a sort input file, its FILE statement should specify OF=N and CF=N (no rewind on open or close).

Internally, Sort/Merge requires a value for maximum record length, even for Record Manager record types that do not require this specification. This value can be specified by the MRL or FL parameters on the FILE control statement, or by the MRL parameter on the OWNCODE directive. The largest value provided is used by Sort/Merge as the value for all files.

## SORTMRG STATEMENT

The SORTMRG statement calls for execution of a sort and/or merge based on specifications provided by Sort/Merge directives. This statement can take one of two forms:

```
SORTMRG.
```

or

```
SORTMRG (parameter list)
```

Parameters can be specified in any order with default values supplied for omitted parameters. The following parameters can be included on the SORTMRG control statement.

## PARAMETERS

### DIRECTIVE FORMAT PARAMETERS 6C AND 7C

The format of the Sort/Merge directives is indicated by specification of 6C or 7C:

6C Indicates Sort/Merge directives are in the formats that apply for Sort/Merge Version 3.

7C Default; indicates Sort/Merge directives are in Sort/Merge Version 4 format.

### SOURCE INPUT PARAMETER I

This parameter specifies the file on which the Sort/Merge directives are located. The directives must be a separate section on this file, terminated by an end-of-section delimiter (7/8/9 card equivalent). The format of this parameter is:

I=lfn/R or I=lfn/NR or I=lfn or I or omitted

I=lfn Sort/Merge directives are on the file whose logical file name is lfn.

R File is rewound before opening.

NR Default; file is not rewound before opening.

I Sort/Merge directives are on the file COMPILE.

omitted Sort/Merge directives are on the file INPUT.

If the system INPUT file is indicated, R should not be specified.

### LIST FILE PARAMETER O

This parameter identifies the file to which output by Sort/Merge is written. The output includes directives and any diagnostics. The format of this parameter is:

O=lfn/R or O=lfn/NR or O=lfn or O or omitted

O=lfn Listings are written to the file whose logical file name is lfn.

The format of the file is BT=C, RT=Z, and FL=140. Specification of different parameters can result in I/O errors.

The format of the file is RT=W, unblocked, and FL=137, unless a FILE control statement overrides this format. Specification of different parameters can result in I/O errors.

R File is rewound before opening.

NR Default; file is not rewound before opening.

O or omitted Listings are written to the file OUTPUT.

### OWNCODE FILE PARAMETER OWN

This parameter indicates the name of the file on which the owncode binaries are located. It has the following format:

OWN=lfn/R or OWN=lfn/NR or OWN=lfn or OWN or omitted

OWN=lfn Owncode binaries are located on file whose logical file name is lfn.

R File is rewound before opening.

NR Default; file is not rewound before opening.

OWN Owncode binaries are on file LGO.

omitted Owncode binaries are on the file INPUT.

If the system INPUT file is indicated, R should not be specified.

### MERGE ORDER PARAMETER MO

The MO parameter specifies the intermediate merge order, which is an internal parameter that governs the number of buffers used by Sort/Merge in the intermediate merge phase. Higher merge orders can produce faster sorts at the expense of greater field length requirements; however, an inefficient merge order might degrade Sort/Merge performance. Appendix F, which explains merge order, should be consulted before merge order is changed. Specification of merge order on the SORTMRG statement overrides specification on the OPTIONS directive. The format of the parameter is:

MO=n or omitted

MO=n Merge order; $2 \leq n \leq 64$. If insufficient core is available to merge at the requested order, a fatal error occurs, and a diagnostic indicates how much additional core would be required.

omitted Sort/Merge computes a merge order based on the amount of memory available.

### EXAMPLES

SORTMRG(OWN=MYFILE,6C)

Calls for execution of Sort/Merge based on directives located on the file INPUT, and owncode binaries on the file MYFILE. Listings are written to the file OUTPUT. The directives are in Sort/Merge Version 3 format.

SORTMRG(7C,O=LISFL/R,I=INPUT/NR)

Calls for execution of Sort/Merge based on directives located on the INPUT file, which is not rewound. These directives are in Sort/Merge Version 4 format. Any owncode binaries are on the INPUT file. Listings are written on LISFL, which is rewound before it is opened.

SORTMRG.

This call is equivalent to:

SORTMRG(7C,I=INPUT/NR,O=OUTPUT/NR,OWN=INPUT/NR)

## SORT/MERGE DIRECTIVE CONVENTIONS

Sort/Merge directives can begin in column 1 but must not continue beyond column 72 on any one source line. Directives can be continued but must have a comma in column 1. The number of continuations for any one directive is limited only by the parameter specifications for the particular directive. As an alternative to continuation, the same directive can be respecified with additional parameters. Blank source lines are allowed.

An asterisk in column 1 indicates the statement is used strictly for comments. The comment statement is ignored by the system during processing and can be placed anywhere in the Sort/Merge input deck. Contents of comment statements are printed out where they occur in the deck.

The following characters are reserved by the program as field or parameter separators for directives. Restrictions placed on separators allowed for each directive type depend on the syntax rules for each directive.

To simplify presentation, only the comma is used as a separator in the directive description. Blanks occurring before and after separators are ignored by the system, except in the case of the SEQUENCE directive. The comma is recommended, but any of the following separators can be used in directives.

| | | | |
|---|---|---|---|
| ( | left parenthesis | - | minus sign |
| ) | right parenthesis | / | slash |
| | blank | * | asterisk |
| . | period | = | equals sign |
| + | plus sign | | |

Unique user parameters can consist of any number of letters and digits, the first a letter, with no embedded blanks, for such specifications as field names, names assigned to user defined collating sequences, filename, etc. The first seven characters, however, must be unique for each specification within a single Sort/Merge run input deck.

## SORT/MERGE DIRECTIVES

The Sort/Merge directives for execution are presented below. The order in which they are listed does not imply a standard input order. With the exception of the END directive, which must appear as the last statement of each section of Sort/Merge directives, the directives specified for any single Sort/Merge run can appear in any sequence.

**SORT**

**MERGE**

**BYTESIZE**

**FILE**

**FIELD**

**KEY**

**SEQUENCE**

**EQUATE**

**OPTIONS**

**OWNCODE**

**TAPE (Sort/Merge Version 4 only)**

**END**

## SORT

The SORT directive is required to specify the kind of Sort Only or Sort/Merge processing.

### OPERATING SYSTEM INCOMPATIBILITIES

The TAPE, POLYPHASE and BALANCED specifications for the VAR parameter cannot be specified for Sort/Merge Version 1, as SCOPE 2.1 does not support a tape variant of Sort/Merge.

The LCMSB parameter is specified for Sort/Merge Version 1 only.

**FORMAT**

SORT,VAR=type,LCMSB=ba

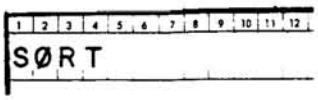
type Processing indicators as listed below. If this parameter is omitted, the system assumes mass storage.

| | |
|---|---|
| DISK | Mass storage Sort/Merge processing. |
| TAPE | Polyphase tape |
| POLYPHASE or POLY | Polyphase tape |
| BALANCED or BAL | Balanced tape |

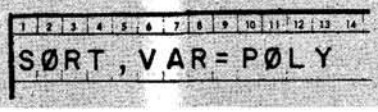
ba Optional parameter specifies, in decimal, the total large core memory (LCM) buffer area for SCOPE 2 Record Manager for all intermediate scratch files developed internally by Sort/Merge. Default value is an installation parameter.

**EXAMPLES**

Using a default value as follows, selects a mass storage sort:




Selects polyphase tape sort processing:




## MERGE

The MERGE directive specifies merge-only processing. The SORT directive is omitted when merge only is selected. See example 5, under JOB EXAMPLES in this section.

**FORMAT**

MERGE

## BYTESIZE

The BYTESIZE directive specifies the number of bits per byte. Defining the bytesize with this directive establishes a standard bytesize for subsequent parameter references to bytes in the FIELD directive. If this directive is omitted, a default value of 6 bits per byte is assumed by the system.

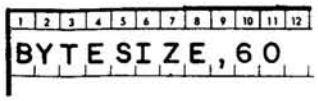
### FORMAT

BYTESIZE,nn

nn Decimal number of bits per byte.

### EXAMPLE

To specify 60 bits per byte:




## FILE

A FILE directive is required to specify all the input and output files to be used during a Sort/Merge run. For a merge run, only MERGE and OUTPUT file types can be specified; for a sort run, SORT, MERGE, INPUT, and OUTPUT file types can be specified.

Each input file is opened with rewind unless the system INPUT file is specified; INPUT is opened with no rewind.

### FORMAT

FILE,type=name(action),name(action),...,type=...

Either FILE or FILES can be specified as the first word of the FILE directive.

The = and ( ) symbols are required as shown in this directive; therefore they cannot be used as separators in other positions.

type File type identifier; all files of a particular type must be specified in one group

INPUT or SORT Sort input file. If no input file is specified, the user must specify Exit 1 on the OWNCODE directive to read in the input records.

MERGE Merge input file.

OUTPUT Sort or merge output file. If no output file is specified, the user must indicate owncode Exit 3 to write the output records. Only one output file is allowed for each sort or merge.

name Logical file name of a file to be processed by Sort/Merge. The name must have been specified previously on a FILE control statement.

action Specifies system action to be performed after file processing is complete.

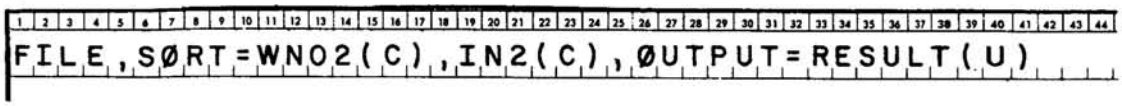
| | |
|---|---|
| C | Close the file |
| R | Rewind the file |
| U | Close and unload the file |
| N | No action is to be taken; however, if type=OUTPUT, an end-of-file is written |
| CR | Close and rewind the file |
| RC | Close and rewind the file; default |
| CU | Close and unload the file |

### EXAMPLE

The FILE directive example has the following parameter specifications.

Two input files, WN02 and IN2, and one output file named RESULT.

The system is requested to close the two input files and unload the output file.




## FIELD

The FIELD directive is required to specify the starting position, length, and data type of a sort or merge key field. These attributes of a key field are described in section 2. At least one FIELD directive must be included in a Sort/Merge run; no more than 100 fields can be specified.

Any sort key can be defined or referenced more than once during a single Sort/Merge run providing a new keyname is specified each time the sort key is defined.

The size of the bytes referenced in the FIELD directive for each job is predefined in the BYTESIZE directive. If BYTESIZE is omitted, the default value of 6 bits per byte is assumed.

### OPERATING SYSTEM INCOMPATIBILITIES

The separate sign feature for signed numeric data and the sign overpunch in the leading character position are supported under Sort/Merge Version 4; therefore, the location and SEPARATE options are applicable to Sort/Merge Version 4.

**FORMAT**

FIELD,keyname(start,length,type,SIGN,location,SEPARATE),...,keyname(...)

The left and right parentheses and the periods are required as shown for this directive; therefore, they cannot be used in other positions.

keyname — Name assigned by user to sort key; keyname can be any number of letters and digits. The first seven characters must be unique and at least one must be alphabetic.

start — Starting position of the sort key as follows:

- byte — Byte number in the record in which the sort key first appears. Bytes are numbered from 1.
- .bit — Bit number, numbered from 1, within the first byte in which the sort key first appears. The system assumes a value of one for an unspecified byte position. The bit number may exceed the number of bits per byte.
- byte.bit — Combination of the byte number and the number of the first bit within that byte in which the sort key first appears.

length — Length of the sort key in one of the following formats:

- nbytes — Number of bytes in the key. A default value of 6 bits per byte is assumed unless specified differently by the BYTESIZE directive.
- .nbits — Number of bits in key. The number of bits may exceed the number of bits per byte.
- nbytes.nbits — Number of bits and bytes in key. A default of 6 bits per byte is assumed unless specified differently by the BYTESIZE directive. The number of bits may exceed the number of bits per byte.

type — Sort key type identifier as follows:

- DISPLAY — Internal display code
- FLOAT — Floating point data
- INTBCD — Internal BCD code
- INTEGER — Signed integer data
- LOGICAL — Unsigned integer data (assumed by system if parameter is omitted)

SIGN — Optional parameter, valid only for sort keys containing numeric data in display code. It indicates the sign is represented by an overpunch on the low order digit of the sort key.

Optional parameter, valid only for sort keys containing numeric data in display code. It indicates the sign is represented either by an overpunch digit or a separate + or - sign, as described by the two following parameters. If SEPARATE is not used, the sign character is an overpunch.

location — Sign position indicator, valid only if SIGN is used, as follows:

- LEADING — Sign character or overpunch digit is at the beginning of the data field.
- TRAILING — Sign character or overpunch digit is at the end of the data field (assumed if parameter is omitted).

The sign character is an overpunch character unless SEPARATE appears.

SEPARATE — Indicator signifying the sign character is a separate + or - character appearing at the beginning or end of the data field. Valid only if SIGN is used.

### EXAMPLE

The FIELD directive example has the following specifications.

Names assigned to the two sort key fields are NAME and JOB.

Starting positions for each sort key field are specified in terms of bytes and bits; the NAME sort key begins in the first byte and the first bit; JOB sort key field begins in byte eleven, bit one. (A BYTESIZE directive has defined bytes to be 12 bits long.)

Length of both sort key fields is specified in terms of bytes; length of the NAME key is ten bytes; length of the JOB key is one byte.

Both keys are written in DISPLAY code.


## KEY

The KEY directive is required to specify the order and collating sequence of 1 to 100 sort keys for a Sort/ Merge run.

### FORMAT

KEY,keyname(order,colseq),...,keyname(...)

The left and right parentheses are required as shown for this directive; therefore, they cannot be used in other positions.

keyname Name assigned to sort key; it must be specified in the FIELD directive.

order Specifies the order in which keys are to be sorted and merged.

A = ascending order (assumed if parameter is omitted)
D = descending order

colseq Name of user specified collating sequence defined in the SEQUENCE directive or one of the following standard collating sequence identifiers. These standard collating sequences are presented in appendix A. A collating sequence is not needed unless the sort key type has been defined to be INTBCD or DISPLAY on the FIELD directive.

ASCII6 6-bit ASCII collating sequence; default for installations using the ASCII character set.

COBOL6 6-bit COBOL collating sequence; default for installation using the CDC character set.

| | |
|---|---|
| DISPLAY | Internal display collating sequence |
| INTBCD | Internal BCD collating sequence |

The default collating sequences can be replaced and respecified with an alternate collating sequence using the SEQUENCE directive.

### EXAMPLE

This KEY directive example has the following parameter specifications:

Two sort keys are named ACCOUNTS and INVEST; both keys are to be sorted in ascending order.

The ACCOUNTS key is to be collated according to the 6-bit ASCII collating sequence. Internal BCD is the collating sequence assigned to the INVEST sort key.

```
KEY,ACCØUNTS(A,ASCII6),INVEST(A,INTBCD)
```

## SEQUENCE

The SEQUENCE directive provides the following capabilities:

Specification of user's own unique collating sequence.

Redefining a standard collating sequence or a user collating sequence to be the default collating sequence.

The collating sequence may be specified by means of characters, octal values, or both. Characters occurring in a user collating sequence are interpreted according to the character set specified as the type parameter of the FIELD directive. Octal values are interpreted as the contents of character positions in memory, regardless of the character set specified.

### FORMAT

SEQUENCE,colseq(c,c,c,c, . . . )

The left and right parentheses and comma are required in the positions shown for this directive; therefore, they may not be used in other positions.

colseq Specifies collating sequence to be used:

Name of user's unique collating sequence.

One of the standard collating sequence identifiers for respecifying a default collating sequence (appendix A):

| | |
|---|---|
| ASCII6 | 6-bit ASCII collating sequence; default for installations using the ASCII character set |
| COBOL6 | 6-bit COBOL collating sequence; default for installation using the CDC character set |
| DISPLAY | Internal display collating sequence |
| INTBCD | Internal BCD collating sequence |

Omitted; indicates the collating sequence in parentheses is to be the default collating sequence.

c Each c is a character or octal value specified in the sequence in which it is to appear in the user collating sequence. Characters or values not specified are considered equal and collated after the specified characters or values.

The system assumes a two-digit number is an octal value (the number must not be followed by a B). When this value occurs as the contents of a character position in a sort key, it is sorted according to the position the value occupies in the user collating sequence, regardless of what character it represents in the character code specified in the FIELD directive.

When a character is specified, the system translates it to a value according to the character set (DISPLAY or INTBCD) specified in the FIELD directive. When this value occurs in a character position in a sort key, it is sorted according to the position the character c occupies in the user collating sequence. A single digit is assumed to be a character, not an octal value.

If colseq identifies a standard collating sequence, no list is provided by the user.
If colseq is omitted, the list in parentheses redefines the default collating sequence.

If the following symbols are included in the user's collating sequence, they must be specified in the c parameter according to their octal equivalent.

| Character | Display Code Octal Equivalent | Internal BCD Octal Equivalent |
|---|---|---|
| ( | 51 | 74 |
| ) | 52 | 34 |
| , | 56 | 73 |
| → | 65 | 75 |

The collating sequence may be continued if necessary. In this case, specification of the collating sequence should extend through column 72 of the first statement and recommence immediately with column 2 of the second statement. Extraneous blanks should not appear because they are not ignored by Sort/Merge. If consecutive blanks appear in the parameter list of the SEQUENCE directive, the system interprets the first blank as a separator, the second blank as the occurrence of the blank character in the collating sequence, the third blank as a separator, and so on. To avoid ambiguity, blanks should not occur within the SEQUENCE directive except to indicate the collating position of the blank character.

### EXAMPLES

In this example the nonstandard collating sequence is named NEWSEQ and A, B, C, D, E, F, G, H, I, J, K comprise the collating sequence.

```
SEQUENCE,NEWSEQ(A,B,C,D,E,F,G,H,I,J,K)
```

The default collating sequence is replaced in the following example:

```
SEQUENCE,(B,D,F,H,J,L,N,P,R,T,V)
```

A continuation statement is used:

```
SEQUENCE,NEWSQ(  ,A,
,1,2,3,4,5,6,7,8,9)
```

```
Y,Z,0,
```

## EQUATE

The EQUATE directive specifies two or more characters already in the collating sequence as equal for comparison purposes.

### FORMAT

EQUATE,colseq(c,c,c,c)(c,c, . . . )

Parentheses and commas are required for this directive in the positions shown; therefore, they may not be used in other positions.

colseq Collating sequence:

Name of a user collating sequence specified in the SEQUENCE directive.

One of the standard collating sequence identifiers:

| | |
|---|---|
| ASCII6 | 6-bit ASCII collating sequence; default for installations using the ASCII character set |
| COBOL6 | 6-bit COBOL collating sequence; default for installations using the CDC character set |
| DISPLAY | Internal display collating sequence |
| INTBCD | Internal BCD collating sequence |

c Characters or values to be equated. The collating position of the list specified within the parentheses is equal to the position of the last character or value specified in the list. Meaning of specification of characters or values is explained under the SEQUENCE directive.

The following symbols, if included in the user's collating sequence, must be specified in the c parameter according to their octal equivalent.

| Character | Display Code Octal Equivalent | Internal BCD Octal Equivalent |
|---|---|---|
| ( | 51 | 74 |
| ) | 52 | 34 |
| , | 56 | 73 |
| → | 65 | 75 |

**EXAMPLE**

In this example, the collating sequence is named LIST.

The characters L, R, T, and 5 are to be equated or assigned a collating value equal to B in the first character string.

In the second character string, the collating values of the characters M and Q are equated to the collating value of the character 3.

```
EQUATE,LIST(L,R,T,5,B)(M,Q,3)
```

## OPTIONS

The OPTIONS directive allows the user to specify special record handling options or operations for sort or merge processing.

### OPERATING SYSTEM INCOMPATIBILITIES

The checkpoint dump features (VOLDUMP, DUMP, and NODUMP) of Sort/Merge are supported only under Sort/Merge Version 4.

The ORDER option is applicable only to Sort/Merge Version 4.

The COMPARE and EXTRACT options are applicable only to Sort/Merge Version 4.

### FORMAT

OPTIONS,option,option, . . .

option Record handling option identifiers:

| | |
|---|---|
| VERIFY | Output file is checked for correct sequencing. If the order of records on the output file is incorrect, the job terminates and the output file is lost. This option verifies that records from merge input files, or inserted through owncode Exits 3, 4, and 5, are in order; it need never be specified for a sort with no owncodes.<br><br>If this parameter is omitted, a sequencing error during a merge run with no inserted records produces a non-fatal error message; all records are written to the output file, but they will not be in order. When records are inserted, no checking takes place; thus omission of this parameter allows the user to to deliberately insert records (such as page headers) out of sequence. |
| RETAIN | Records with identical sort keys that are read from sort input files are written in the order they are read. That is, all the records from each file are grouped together, and the groups occur in the order that the files are specified in the FILE directive. Records from merge files are sequenced arbitrarily. When this parameter is omitted, records with identical sort keys are sequenced arbitrarily. |
| VOLDUMP | Checkpoint dump is to be taken when end-of-volume condition on input file or new-volume condition on output file exists. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files. |
| DUMP(nn) | Parentheses are required for this parameter in the position shown; they cannot be used in other positions in this directive. Checkpoint dump is to be taken when the specified number of records is read from the input file and when the same number of records is written on the output file. nn is decimal. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files.<br><br>If DUMP is specified without nn, a checkpoint dump is taken after each group of 50,000 records on input and output files. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files. |
| NODUMP | No checkpoint dumps are taken; default. |

| | |
|---|---|
| ORDER(n) | Parentheses are required for this parameter; they cannot be used in other positions in this directive. n specifies the intermediate merge order; $2 \leq n \leq 64$. Merge order is explained in appendix F. Specification of merge order on the SORTMRG control statement overrides specification here. If core is not sufficient to merge at the order specified, a fatal error occurs and a diagnostic indicates how much additional core would be required. When no merge order is specified, Sort/Merge computes one based on the amount of memory available. |
| COMPARE | The key comparison technique is to be used. This technique requires less elapsed time and more central processing (CP) time than key extraction. |
| EXTRACT | The key extraction technique is to be used. This technique requires less CP time and more elapsed time than key comparison. |

The COMPARE and EXTRACT options are mutually exclusive within a single sort. If neither option is specified, Sort/Merge attempts to choose the best technique.

If the user requests the OPTIONS directive more than once, only the last OPTIONS directive applies.

#### EXAMPLE

This example requests the VERIFY option. It also requests the DUMP option after each 10,000 records are read from the input file and written on the output file.

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
ØPTIØNS,VERIFY,DUMP(10000)
```

## OWNCODE

The OWNCODE directive is required to specify legal entry point names to a user's relocatable owncode exit routines.

#### OPERATING SYSTEM INCOMPATIBILITIES

Owncode exit 6 for checking nonstandard labels on input files is provided only under Sort/Merge Version 4. Therefore, exit 6 cannot be specified for the exitno parameter under Sort/Merge Version 1.

#### FORMAT

OWNCODE,MRL=mrl,exitno=entry,exitno= . . .

The equals sign is required for this directive in the positions shown; therefore, it cannot be used otherwise.

| | |
|---|---|
| mrl | Maximum decimal record length in 6-bit characters. This parameter is required if an input file has not been specified, and not required otherwise. |
| exitno | Number of the owncode exit desired. |
| entry | Corresponding entry point name of exit. |

When using OWNCODE binaries, Sort/Merge momentarily uses 10000B more words than when first called. The additional 10000B word requirement must be included in the maximum field length. The maximum field length is determined by the CM parameter on the JOB control statement. If CM is not specified, the maximum field length is the machine capacity.

If the extra 10000B words required for loading OWNCODE binaries are not available, a diagnostic message is issued.

During the actual sort or merge, no more than the initial field length is used.

**EXAMPLE**

In this example, the maximum record length is forty 6-bit characters; exits 2 and 4 are specified with the entry point names INTRO and OUTFROM.

```
ØWNC ØDE , MRL = 40 , 2 = I NTRØ , 4 = ØUT FRØM
```

## TAPE (SORT/MERGE VERSION 4 ONLY)

The TAPE directive is required for the tape version of Sort/Merge to specify all magnetic tape intermediate merge files. If the tape files have not been defined in a previous job step (REQUEST, REWIND, etc.), Sort/Merge issues requests for scratch tapes to be assigned to the intermediate files as needed.

Balanced Merge:

A minimum of four tapes is required; the maximum is limited only by the number of tape drives available. A balanced merge is more efficient if an even number of tapes is specified.

Polyphase Merge:

A minimum of three tapes is required; the maximum is limited only by the number of tape drives available.

**FORMAT**

TAPE,filename,filename, . . .

filename Name assigned to the intermediate merge file.

**EXAMPLE**

This example assigns names to four intermediate merge files.

```
TAPE , INTERMI , INTERM2 , I NTERM3 , INTERM4
```

## END

The END directive is required to signify the end of each Sort/Merge run deck.

**FORMAT**

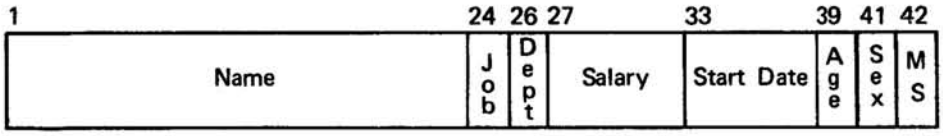
END

## JOB EXAMPLES

The following Sort/Merge job examples provide direct applications of the Sort/Merge directives.

The structure of the records used as input for these examples is defined below. The record format dimensions are represented in 6-bit characters.

### RECORD FORMAT




Name — 23-character internal BCD coded field

Job — 2-character internal BCD coded field containing a job grade identifier

| | |
|---|---|
| 2 | employee |
| 4 | foreman |
| 6 | supervisor |
| 8 | manager |
| 10 | general manager |
| 12 | director |

Dept — 1-character display coded field containing a department identifier

| | |
|---|---|
| A | production department |
| B | shipping department |
| C | personnel department |
| D | accounting department |
| E | sales department |

Salary — 6-character display coded field

Start Date — 6-character display coded field in the format mmddyy:

| | |
|---|---|
| mm | month |
| dd | day |
| yy | year |

Age — 2-character internal BCD coded field

| | |
|---|---|
| Sex | 1-character display coded field |
| MS | 1-character display coded field containing marital status |

| | |
|---|---|
| M | married |
| S | single |
| D | divorced |

**EXAMPLE 1**

Job Requirements:

1. Sort the records located on the input file IN1 on the basis of:

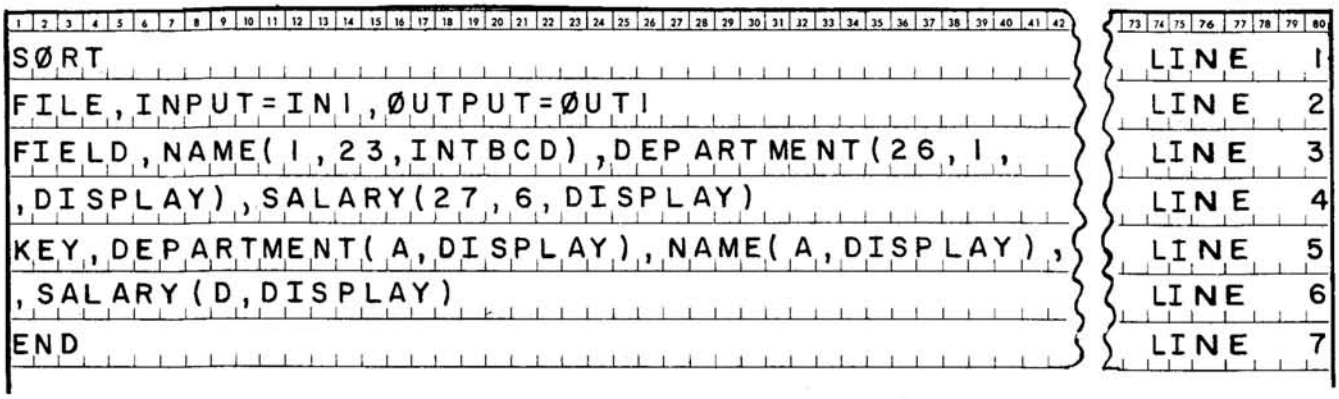
   Department in alphabetical order

   Name in alphabetical order

   Salary beginning with the highest paid

2. Return the output to file OUT1.

Job Code:



```
SØRT                                                        LINE 1
FILE,INPUT=IN1,ØUTPUT=ØUT1                                  LINE 2
FIELD,NAME(1,23,INTBCD),DEPARTMENT(26,1,                    LINE 3
,DISPLAY),SALARY(27,6,DISPLAY)                              LINE 4
KEY,DEPARTMENT(A,DISPLAY),NAME(A,DISPLAY),                  LINE 5
,SALARY(D,DISPLAY)                                          LINE 6
END                                                         LINE 7
```

Code Interpretation:

| Line | Significance |
|---|---|
| 1 | SORT directive specifies a mass storage sort. |
| 2 | FILE directive specifies the input file as IN1 and the output file as OUT1. |

| Line | Significance |
|---|---|
| 3 & 4 | FIELD directive defines:<br>NAME key, 23 bytes in length, beginning in byte 1, and coded in internal BCD.<br>DEPARTMENT key, 1 byte in length, beginning in byte 26, and coded in display.<br>SALARY key, 6 bytes in length, beginning in byte 27, and coded in display. |
| 5 & 6 | KEY directive specifies:<br>DEPARTMENT key to be sorted in ascending order according to the display collating sequence.<br>NAME key to be sorted in ascending order according to the display collating sequence.<br>SALARY key to be sorted in descending order according to the display collating sequence. |

**EXAMPLE 2**

Job Requirements:

1. Sort the records on the input file IN1 on the basis of:
   Department, in the order: sales, personnel, accounting, shipping, and production
   Salaries within these departments beginning with the highest salary
   Ages, beginning with the oldest, of employees in each department and salary level
2. Return the output to file OUT1.

Job Code

```
SØRT                                                                    LINE 1
FILE,INPUT=IN1,ØUTPUT=ØUT1                                              LINE 2
FIELD,DEPARTMENT(26,1,DISPLAY),SALARY(27,6,                             LINE 3
,DISPLAY),AGE(39,2,INTBCD)                                              LINE 4
KEY,DEPARTMENT(A,ØWN),SALARY(D,DISPLAY),                                LINE 5
,AGE(D,DISPLAY)                                                         LINE 6
SEQUENCE,ØWN(E,C,D,B,A)                                                 LINE 7
END                                                                     LINE 8
```

Code Interpretation

| Line | Significance |
|---|---|
| 1 | SORT directive specifies a mass storage sort. |
| 2 | FILE directive specifies the input file as IN1 and the output file as OUT1. |
| 3 & 4 | FIELD directive defines:<br>DEPARTMENT key, 1 byte in length, beginning in byte 26, and coded in display.<br>SALARY key, 6 bytes in length, beginning in byte 27, and coded in display.<br>AGE key, 2 bytes in length, beginning in byte 39, and coded in internal BCD. |
| 5 & 6 | KEY directive specifies:<br>DEPARTMENT key to be sorted in ascending order according to a unique collating sequence assigned the name OWN.<br>SALARY key to be sorted in descending order according to the display collating sequence.<br>AGE key to be sorted in descending order according to the display collating sequence. |
| 7 | SEQUENCE directive specifies the unique collating sequence, OWN, used for sorting the DEPARTMENT key in the order requested. |

**EXAMPLE 3**

Job Requirements:

1. Sort records on the input file IN1 on the basis of:
   - Employee start dates beginning with the most recently hired
   - Ages of employees starting with the oldest
   - Marital status; divorced and single employees are to be considered on an equal basis
   - Names of employees in alphabetical order
2. Return the output to file OUT1.

Job Code:

```
SØRT                                                          LINE  1
FILE,INPUT=IN1,ØUTPUT=ØUT1                                    LINE  2
FIELD,YEAR(37,2,DISPLAY),MØNTH(33,2,DISPLAY),                 LINE  3
,DAY(35,2,DISPLAY),AGE(39,2,INTBCD),MS(42,                    LINE  4
,1,DISPLAY),NAME(1,23,INTBCD)                                 LINE  5
KEY,YEAR(D,DISPLAY),MONTH(D,DISPLAY),DAY(D,                   LINE  6
,DISPLAY),AGE(D,DISPLAY),MS(D,INTBCD),NAME                    LINE  7
,(A,DISPLAY)                                                  LINE  8
EQUATE,INTBCD(S,D)                                            LINE  9
END                                                           LINE 10
```

Code Interpretation:

| Line | Significance |
|---|---|
| 1 | SORT directive specifies a mass storage sort. |
| 2 | FILE directive specifies the input file as IN1 and the output file as OUT1. |
| 3, 4, 5 | FIELD directive defines:<br><br>YEAR key extracted from the last two bytes of the START DATE field. The key begins in byte 37, is 2 bytes in length, and is coded in display.<br><br>MONTH key extracted from the START DATE field. The key begins in byte 33, is 2 bytes in length, and is coded in display.<br><br>DAY key taken from the START DATE field. The key begins in byte 35, is 2 bytes in length, and is coded in display.<br><br>AGE key, begins in byte 39, is 2 bytes in length, and is coded in INTBCD.<br><br>MS key begins in byte 42, is one byte in length, and is coded in display.<br><br>NAME key, begins in byte 1, is 23 bytes in length, and is coded in INTBCD. |
| 6, 7, 8 | KEY directive specifies:<br><br>YEAR key is to be sorted in descending order according to the display collating sequence.<br><br>MONTH key is to be sorted in descending order according to the display collating sequence.<br><br>DAY key is to be sorted in descending order according to the display collating sequence. |

AGE key is to be sorted in descending order according to the display collating sequence.

MS key is to be sorted in descending order according to the internal BCD collating sequence.

NAME key is to be sorted in ascending order according to the display collating sequence.

9 EQUATE directive assigns equal processing value to marital status indicators, for divorced (D) and single (S).

**EXAMPLE 4**

Job Requirements:

1. Sort the records on input tape IN1 on the basis of:

   Job grades beginning with the director down through employees, with foreman and supervisors considered as equal in status

   Salary beginning with the highest paid

   Names of employees in alphabetical order

2. Dump every 1,000 records for processing backup

3. Return the output to file OUT1.

Job Code:

```
SØRT,VAR=BAL                                                    LINE 1
FILE,INPUT=INI(C),ØUTPUT=ØUTI(CR)                               LINE 2
TAPE,INTERMI,INTERM2,INTERM3,INTERM4                            LINE 3
FIELD,NAME(1,23,INTBCD),JØB(24,2,INTBCD),                       LINE 4
,SALARY(27,6,DISPLAY)                                           LINE 5
KEY,JØB(D,INTBCD),SALARY(D,DISPLAY),NAME                        LINE 6
,(A,DISPLAY)                                                    LINE 7
EQUATE,INTBCD(4,6)                                              LINE 8
ØPTIØNS,DUMP(1000)                                              LINE 9
END                                                             LINE 10
```

Code Interpretation:

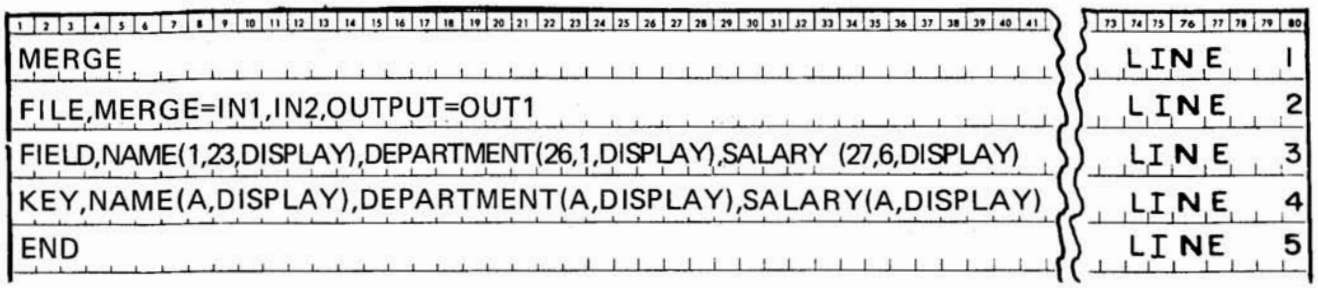
| Line | Significance |
|---|---|
| 1 | SORT directive specifies a balanced tape sort. |
| 2 | FILE directive specifies:<br>Input file as IN1; after processing, the file is to be closed.<br>Output file as OUT1; after processing the file is to be closed and rewound. |
| 3 | TAPE directive specifies four intermediate merge files for the balanced Sort/Merge. |
| 4 & 5 | FIELD directive specifies:<br>NAME key begins in byte 1, is 23 bytes in length, and is coded in internal BCD.<br>JOB key begins in byte 24, is 2 bytes in length, and is coded in internal BCD.<br>SALARY key begins in byte 27, is 6 bytes in length, and is coded in display. |
| 6 & 7 | KEY directive specifies:<br>JOB key is to be sorted in descending order according to the internal BCD collating sequence.<br>SALARY key is to be sorted in descending order according to the display collating sequence.<br>NAME key is to be sorted in ascending order using the display collating sequence. |
| 8 | EQUATE directive assigns equal processing value to the job grade indicators for foreman (4) and supervisor (6). |
| 9 | OPTIONS directive specifies a restart dump to be taken after every 1000 records are read and written. |

**EXAMPLE 5**

Job Requirements:

1. Merge the records located on the pre-sorted files, IN1 and IN2.
2. Return the output to file OUT1.

Job Code:




Code Interpretation:

| Line | Significance |
|---|---|
| 1 | MERGE directive specifies merge-only processing. |
| 2 | FILE directive specifies the merge files as IN1 and IN2 and the output file as OUT1. |
| 3 | FIELD directive defines:<br><br>NAME key, 23 bytes in length, beginning in byte 1, and coded in display.<br><br>DEPARTMENT key, 1 byte in length, beginning in byte 26, and coded in display.<br><br>SALARY key, 6 bytes in length, beginning in byte 27, and coded in display. |
| 3 | KEY directive specifies:<br><br>NAME key to be sorted in ascending order according to the display collating sequence.<br><br>DEPARTMENT key to be sorted in ascending order according to the display collating sequence.<br><br>SALARY key to be sorted in ascending order according to the display collating sequence. |

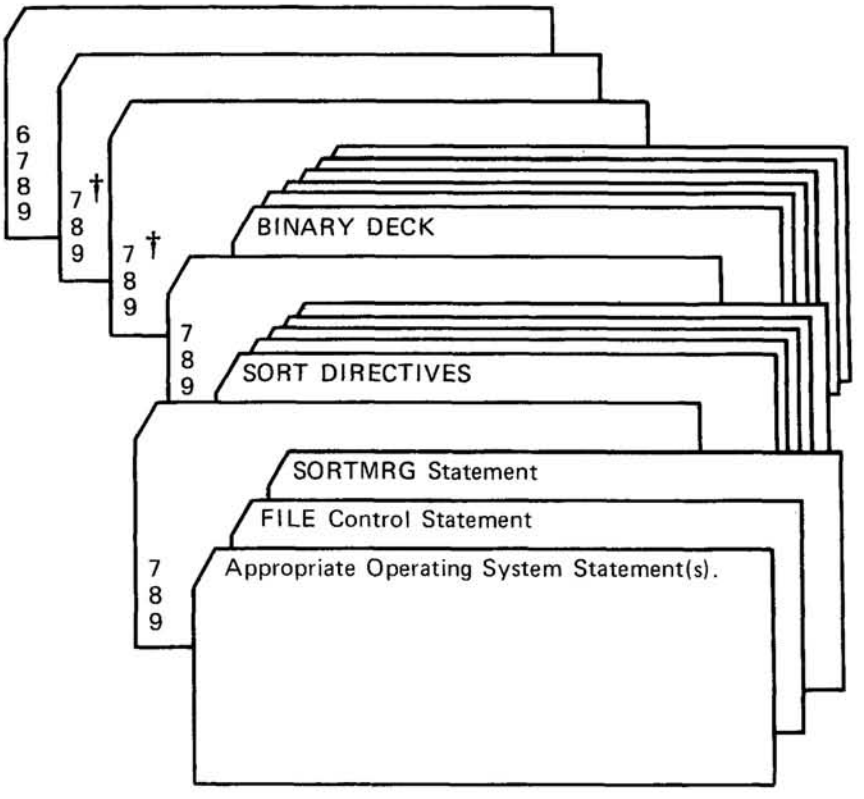
Processing of a MERGE directive is accomplished through the SORTMRG control statement, which must immediately precede the MERGE directive. A Record Manager FILE control statement must be provided for every file processed by a directive sort or merge.

## SAMPLE DECK STRUCTURES

The NOS operating system requires a job name statement and a USER statement. Depending on installation option, a CHARGE statement might also be required.

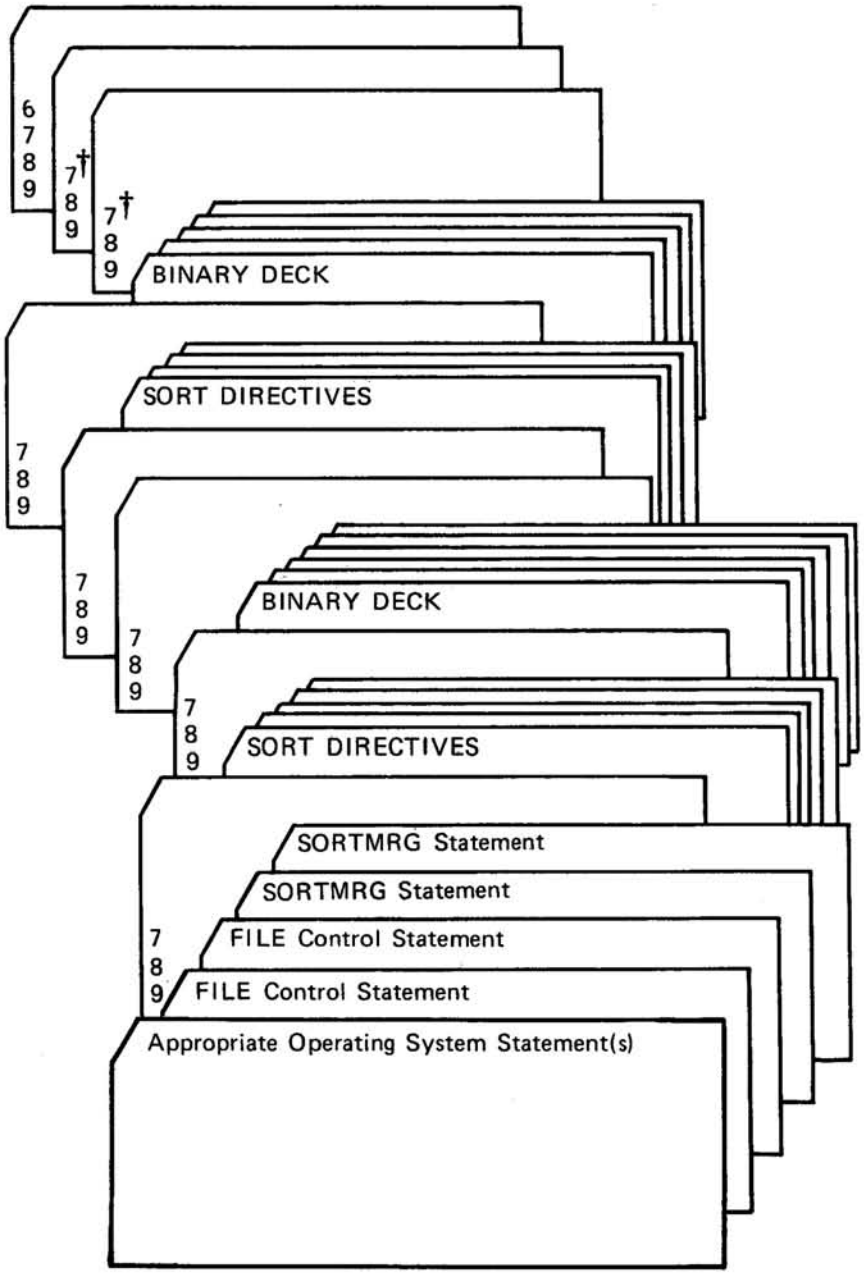
Both the NOS/BE and SCOPE operating systems require a job name statement. An ACCOUNT statement might be required as an installation option.

### JOB DECK CONTAINING A SINGLE SORT/MERGE RUN WITH OWNCODE




†A binary deck run on NOS/BE must be followed by two 7/8/9 cards.

JOB DECK CONTAINING TWO SORT/MERGE RUNS WITH OWNCODE




†A binary deck run on NOS/BE must be followed by two 7/8/9 cards.

# MACRO CALLS 5

When Sort/Merge functions as a COMPASS subroutine for a COMPASS program or as a relocatable program generated for the COBOL SORT verb, the calling sequence is written as a series of macro instructions. In specifying macro instructions, the user must satisfy the following requirements:

1. A SORT or MERGE macro call must precede all other Sort/Merge macro calls specified.

2. All macro calls for any one Sort/Merge run must be specified in a continuous sequence, except when the POINTER and SMLIST macros are used.

3. The system control statement LIBRARY(SRTLIB) is required preceding the relocatable load of the macro binaries.

4. The Sort/Merge macros should not be assembled with the COMPASS LIST M or F options turned on because of the large number of source statements they generate. LIST G can be used to check the macro formats generated.

5. S=SMTEXT is required on the call for COMPASS assembly to ensure proper expansion of Sort/Merge macros under Sort/Merge Version 4.

6. To ensure proper expansion of Sort/Merge macros with Sort/Merge Version 1, the control statement

   ```
   ATTACH(SRTMACS, SRTMACS, ID=PRDLIB)
   ```

   must precede the COMPASS control statement, and the COMPASS program must include the statement

   ```
   SRTMACS    XTEXT
   ```

   with SRTMACS in the location field and XTEXT in the operation field.

After processing each series of macro calls, Sort/Merge returns control to the location immediately following the last macro specified in that series.

Sort/Merge Version 4 uses the Common Memory Manager (CMM) to allocate its space; it does not use the area specified by RA+65B through field length. At least 22000B words of space should be available for Sort/Merge. Sorts with more than 2000 character records, with many user-supplied merge input files, or with other exceptional conditions might require more words. If an insufficient field length for the sort is specified in the CM parameter on the job statement, or if the memory limit is reached during the sort, Sort/Merge issues a diagnostic and aborts.

## SYSTEM FILE MACRO

Every input or output file to be processed by Sort/Merge macros must have a valid file information table when macros are executed. Usually, this table is created through the Record Manager FILE macro. Record Manager requirements for macro sorts and merges are described in appendix D.

Internally, Sort/Merge requires a value for maximum record length in characters, even for Record Manager file structures which do not require this specification (for example, BT=I, RT=W). This value can be specified by the MRL or FL parameters on the FILE control statement, FILE macro, or STORE macro for input or output files, or by the MRL parameter in the OWNCODE macro. Sort/Merge will use the greatest of the values provided.

For Sort/Merge Version 1, if the input file is the standard system INPUT file (filename on the FILE macro is INPUT), the FILE macro should be specified with no rewind for the open file and close file parameters.

## SORT/MERGE MACRO CALLS

Each macro call presented in this section is described for all operating systems. Incompatibilities among the systems in terms of macro call specification are identified within the presentation of each macro call.

The following macro calls are provided for Sort/Merge execution under the operating systems:

| | |
|---|---|
| SORT | EQUATE |
| MERGE | OPTIONS |
| BYTESIZE | OWNCODE |
| FILES | TAPE (Sort/Merge Version 4) |
| KEY | POINTER (Sort/Merge Version 4) |
| SEQUENCE | SMLIST (Sort/Merge Version 4) |

### SORT

The SORT macro call is required to initiate Sort/Merge functions as a subprogram within a job requesting Sort/Merge processing.

#### OPERATING SYSTEM INCOMPATIBILITIES

The tape variant of Sort/Merge is supported only under Sort/Merge Version 4. Therefore SORTB and SORTP macro call variants of the SORT macro are available only under Version 4.

The MAXCM and CM parameters can be specified only under Sort/Merge Version 4.

The ba parameter can be specified only under Sort/Merge Version 1.

#### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SORT | ba,MAXCM=n, CM=BELOWHHA |

ba — This optional parameter specifies in decimal the total large core memory (LCM) buffer area for SCOPE 2 Record Manager for all intermediate scratch files constructed internally by Sort/Merge. The default value is an installation parameter.

Two alternate formats of the SORT macro call for the tape variant of Sort/Merge are as follows.

For a balanced merge (appendix E):

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SORTB | MAXCM=n, CM=BELOWHHA |

For a polyphase merge (appendix E):

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SORTP | MAXCM=n, CM=BELOWHHA |

MAXCM=n — This required parameter specifies the maximum number of central memory (CM) words that Sort/Merge is to use for its working storage area, including space needed for Sort/Merge subroutines. At least 22000B words should be provided. If zero is specified, Sort/Merge uses a default size. n can specify a register.

CM=BELOWHHA — This optional parameter is meaningful only for overlay programs. If specified, the Common Memory Manager preferentially allocates the working storage area for Sort/Merge between the last word address (LWA) of the last loaded overlay and the highest LWA of all overlays, which is the highest high address (HHA). If the parameter is omitted, the working storage area for Sort/Merge begins higher than the highest LWA of all overlays. If an overlay containing a sort does not load other overlays during its execution, this parameter should be specified in order to reduce the amount of memory required for the program. The parameter should not be specified if overlays are loaded during the sort, since the contents of the working storage area for Sort/Merge would be overlayed and lost.

## MERGE

The MERGE macro call specifies merge-only processing.

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | MERGE | MAXCM=n, CM=BELOWHHA |

MAXCM=n — This required parameter specifies the maximum number of central memory (CM) words that Sort/Merge is to use for its working storage area, including space needed for Sort/Merge subroutines. At least 22000B words should be provided. If zero is specified, Sort/Merge uses a default size. n can specify a register.

CM=BELOWHHA This optional parameter is meaningful only for overlay programs. If specified, the Common Memory Manager preferentially allocates the working storage area for Sort/Merge between the last word address (LWA) of the last loaded overlay and the highest LWA of all overlays, which is the highest high address (HHA). If the parameter is omitted, the working storage area for Sort/Merge begins higher than the highest LWA of all overlays. If an overlay containing a sort does not load other overlays during its execution, this parameter should be specified in order to reduce the amount of memory required for the program. The parameter should not be specified if overlays are loaded during the sort, since the contents of the working storage area for Sort/Merge would be overlayed and lost.

## BYTESIZE

The BYTESIZE macro call specifies the number of bits per byte. Defining the bytesize with this macro call establishes a standard bytesize for subsequent parameter references to bytes in the KEY macro call:

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | BYTESIZE | nn |

nn Specifies the number of bits per byte.

### EXAMPLE

Specifies 60 bits per byte with the BYTESIZE macro call:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | BYTESIZE | 60 |

## FILES

The FILES macro call defines the names of all input and output files to be used during Sort/Merge processing.

The files to be defined can be specified in any order. With the exception of the output file which is limited to one specification, any of the remaining file types listed below can have up to 62 individual files specified with a maximum of 63 files per macro call. A FILES macro is required to specify all the input and output files to be used during a Sort/Merge run. For a merge run, only MERGE and OUTPUT file types can be specified; for a sort run, SORT, MERGE, INPUT, and OUTPUT file types can be specified.

If an input file is not specified, the user must define owncode Exit 1 for record input. Omission of an output file specification requires the user to specify owncode Exit 3 to provide for record output.

Each input file is opened with rewind unless the file is already open or unless the system INPUT file is specified; INPUT is opened with no rewind.

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | FILES | (type,name,name),(type,name, . . . |

type Type identifiers:

| | |
|---|---|
| SORT | Sort input file |
| INPUT | Sort input file |
| MERGE | Merge input file |
| OUTPUT | Output file |

name Location of file information table for file to be processed by Sort/Merge.

### EXAMPLE

This example has two input files named ACCOUNTS and ASSETS. One output file is specified, named RESULT.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | FILES | (INPUT,ACCOUNTS,ASSETS),(OUTPUT,RESULT) |

## KEY

The KEY macro call specifies each sort key used in sort and merge processing. A single KEY macro call is required for each sort key specified. A maximum of 100 sort keys can be specified in any one run.

The placement of the individual KEY macro calls within the program determines the processing priority of each sort key. Each sort key specified within a macro call is processed before subsequent sort keys specified in additional macro calls.

Sort keys are discussed more fully in section 2.

### OPERATING SYSTEM INCOMPATIBILITIES

The separate sign feature for signed numeric data and the sign overpunch in the leading character position are supported under Sort/Merge Version 4; therefore, the location and SEPARATE options are applicable to Sort/Merge Version 4.

**FORMAT**

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | KEY | bytepos,bitpos,nbytes,nbits,type,colseq,order,SIGN,location,SEPARATE |

bytepos Position of first byte of sort key in relation to first byte of record, counting from 1.

bitpos Position of first bit of sort key in the byte indicated by bytepos, counting from 1.

nbytes Specifies number of complete bytes in sort key.

nbits Specifies number of bits in sort key in addition to number of complete bytes specified in the previous parameter. The number of bits per byte is 6 unless a BYTESIZE macro is provided.

The following parameters are optional.

type Coding identifier; this parameter always must precede any colseq parameter specified.

DISPLAY Internal display code

FLOAT Floating point data

INTBCD Internal BCD code

INTEGER Signed integer data

LOGICAL Unsigned integer data

colseq Name of a user specified collating sequence defined in the SEQUENCE macro call or one of the following collating sequence identifiers (appendix A). A collating sequence need be specified only if the sort key type is identified as INTBCD or DISPLAY.

ASCII6 6-bit ASCII collating sequence; default for installations using the ASCII character set

COBOL6 6-bit COBOL collating sequence; default for installations using the CDC character set

DISPLAY Internal display collating sequence

INTBCD Internal BCD collating sequence

The default collating sequences can be replaced and respecified with an alternate collating sequence using the SEQUENCE macro call.

order Sequencing order of sort and merge processing:

A Ascending (assumed if parameter is omitted)
D Descending

SIGN — This optional parameter is valid only for sort keys containing numeric data in display code. It indicates the sign is represented by an overpunch on the low order digit of the sort key.

This optional parameter is valid only for sort keys containing numeric data in display code. It indicates the sign is represented either by an overpunch digit or a separate + or - sign, as described by the two following parameters. If SEPARATE is not used, the sign character is an overpunch.

location — Sign position indicator, valid only if SIGN is used. The sign character is an overpunch character unless SEPARATE appears.

- LEADING — Sign character or overpunch digit is at the beginning of the data field
- TRAILING — Sign character or overpunch digit is at the end of the data field (assumed if parameter is omitted)

SEPARATE — Indicator signifying that the sign character is a separate + or - character appearing at the beginning or end of the data field. Valid only if SIGN is used.

#### EXAMPLE

The KEY macro call example has the following specifications.

Sort key begins in byte 1, bit 7.

Key size consists of 1 byte and 12 bits.

Sort key is coded in DISPLAY.

Sort key is to be sequenced according to the internal BCD collating sequence.

Sequencing is ascending.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | KEY | 1,7,1,12,DISPLAY,INTBCD,A |

## SEQUENCE

The SEQUENCE macro call provides the following capabilities:

Specification of user's own unique collating sequence.

Redefining as default a user collating sequence or a standard collating sequence other than the system default.

A collating sequence need be specified only if the sort key type is specified as INTBCD or DISPLAY.

#### FORMATS

To specify a complete sequence with a single statement:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SEQUENCE | colseq,(c,c,...),END |

To define a continuing sequence with successive macro calls:

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SEQUENCE | colseq,(c,c,c,c,c,c,c,c,c,c,c) |
| | SEQUENCE | (c,c,c,c,c,c,c),END |

colseq Collating sequence:

Name of user's unique collating sequence; it cannot be A, D, SIGN, or END.

Standard collating sequence identifier for respecifying the default collating sequence (appendix A).

| | |
|---|---|
| ASCII6 | ASCII collating sequence; default for installations using the ASCII character set |
| COBOL6 | COBOL collating sequence; default for installations using the CDC character set |
| DISPLAY | Internal display collating sequence |
| INTBCD | Internal BCD collating sequence |

Omitted; indicates the collating sequence in parentheses is to be the default collating sequence.

c Each c is a character or octal value specified in the sequence in which it is to appear in the user collating sequence. Characters or values not specified are considered equal and collated after the specified characters or values.

The system assumes a two-digit number is an octal value (the number must not be followed by a B). When this value occurs as the contents of a character position in a sort key, it is sorted according to the position the value occupies in the user collating sequence, regardless of what character it represents in the character code specified in the KEY macro.

When a character is specified, the system translates it to a value according to the character set (DISPLAY or INTBCD) specified in the KEY macro. When this value occurs in a character position in a sort key, it is sorted according to the position the character c occupies in the collating sequence. A single digit is assumed to be a character, not an octal value.

If colseq identifies a standard collating sequence, no list is provided by the user.

If colseq is omitted, the list in parentheses redefines the default collating sequence.

The following symbols, if included in the user's collating sequence, must be specified in the c parameter according to their octal equivalent or by the corresponding descriptive identifiers assigned to the symbol. The descriptive identifiers are considered to be characters, as defined above.

| Character | Display code octal equivalent | Internal BCD octal equivalent | Descriptive Identifier |
|---|---|---|---|
| ( | 51 | 74 | LEFT |
| ) | 52 | 34 | RIGHT |
| blank | 55 | 60 | BLANK or SPACE |
| , | 56 | 73 | COMMA |
| → | 65 | 75 | ARROW |

### EXAMPLES

This example names the nonstandard collating sequence, NEWSEQ, and specifies the characters comprising the sequence.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SEQUENCE | NEWSEQ,(A,B,C,D,F,G,H,I,J,K),END |

The default collating sequence is replaced by the following collating sequence to be used by the system as a new default value for the KEY macro.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | SEQUENCE | ,(B,D,F,J,L,N,P,R,T,V),END |

## EQUATE

The EQUATE macro call equates or specifies two or more characters already specified in a collating sequence as equal when comparisons are made between these characters for sort and/or merge processing.

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | EQUATE | colseq,(c,c,c),(c,c),END |

colseq Collating sequence

Name of a user specified collating sequence specified on the SEQUENCE macro call

Collating sequence identifier:

| | |
|---|---|
| ASCII6 | 6-bit ASCII collating sequence; default for installations using the ASCII character set |
| COBOL6 | 6-bit COBOL collating sequence; default for installations using the CDC character set |
| DISPLAY | Internal display collating sequence |
| INTBCD | Internal BCD collating sequence |

c Characters or values to be equated. The collating position of the list specified within the parentheses is equal to the position of the last character or value specified in the list. Meaning of specification of characters or values is explained under the SEQUENCE macro.

The following symbols, if included in the user's collating sequence, must be specified in the c parameter according to their octal equivalent or by their descriptive identifiers assigned to the symbol.

| Character | Display code octal equivalent | Internal BCD octal equivalent | Descriptive Identifier |
|---|---|---|---|
| ( | 51 | 74 | LEFT |
| ) | 52 | 34 | RIGHT |
| blank | 55 | 60 | BLANK or SPACE |
| , | 56 | 73 | COMMA |
| → | 65 | 75 | ARROW |

Descriptive identifiers are considered the same as characters, as defined under the SEQUENCE macro.

**EXAMPLE**

The name assigned to the collating sequence is LIST; the characters L, R, T, and 5 are to be equated and assigned a collating value equal to B in the first character string.

In the second grouping of characters the collating values of the characters M and Q are equated to the collating value of the character 3.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | EQUATE | LIST,(L,R,T,5,B),(M,Q,3),END |

## OPTIONS

The OPTIONS macro call specifies special record handling options or operations for sort and/or merge processing.

### OPERATING SYSTEM INCOMPATIBILITIES

The checkpoint dump features of Sort/Merge are supported only under Sort/Merge Version 4; therefore, the VOLDUMP, DUMP, NODUMP options are applicable to Sort/Merge Version 4.

The ORDER option is applicable only to Sort/Merge Version 4.

The COMPARE and EXTRACT options are applicable only to Sort/Merge Version 4.

The OPTIONS macro must immediately follow the SORT or MERGE macro under Sort/Merge Version 1.

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | OPTIONS | option,option,... |

option One or a combination of the following record handling identifiers:

VERIFY Output file is checked for correct sequencing. If the order of records on the output file is incorrect, the job terminates and the output file is lost. This option verifies that records from merge input files, or inserted through owncode exits 3, 4, and 5, are in order; it need never be specified for a sort with no owncodes.

If this parameter is omitted, a sequencing error during a merge run with no inserted records produces a non-fatal error message; all records are written to the output file, but they will not be in order. When records are inserted, no checking takes place; thus omission of this parameter allows the user to deliberately insert records (such as page headers) out of sequence.

RETAIN Records with identical sort keys that are read from sort input files are written in the order they are read. That is, all the records from each file are grouped together, and the groups occur in the order that the files are specified in the FILE directive. Records from merge files are sequenced arbitrarily. If this parameter is omitted, records with identical sort keys are sequenced arbitrarily.

VOLDUMP Checkpoint dump is taken when any end-of-volume condition is encountered on the input file or when a new volume condition is detected on the output file. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files.

(DUMP,nn) Checkpoint dump is taken when nn records (decimal) are read and written. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files.

If nn is not specified, a checkpoint dump is taken after each group of 50,000 records is read and after each group of 50,000 records is written. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files.

NODUMP No checkpoint dumps are taken; default.

(ORDER,n) n is the intermediate merge order; $2 \leq n \leq 64$. Merge order is explained in appendix F. If core is not sufficient to merge at the order specified, a fatal error occurs and a diagnostic indicates how much additional core would be required. When merge order is not specified, it is calculated by Sort/Merge based on the amount of memory available.

COMPARE The key comparison technique is to be used. This technique usually requires less elapsed time and more central processing (CP) time than key extraction.

EXTRACT The key extraction technique is to be used. This technique usually requires less CP time and more elapsed time than key comparison.

The COMPARE and EXTRACT options are mutually exclusive within a single sort. If neither option is specified, Sort/Merge attempts to choose the best technique.

**EXAMPLE**

This example requests the VERIFY option. It also requests that the DUMP option be taken after each 10,000 records are read from the input file and written on the output file.

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | OPTIONS | VERIFY,(DUMP,10000) |

## OWNCODE

The OWNCODE macro call is required to specify the entry point names to a user's relocatable owncode exit routines.

### OPERATING SYSTEM INCOMPATIBILITIES

An owncode Exit 6 for the checking of labels on input files is provided only under Sort/Merge Version 4; therefore, Exit 6 is an allowable specification for the exitno parameter of the OWNCODE macro only under Version 4.

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | OWNCODE | (MRL,mrl),(exitno,entry),(exitno, ... |

mrl Maximum record length in 6-bit characters. This parameter is required if an input file has not been specified, and not required otherwise.

exitno Number of owncode exit.

entry Entry point name for exit specified.

### EXAMPLE

In this example, the maximum record length is forty 6-bit characters.

Exits 2 and 4 are specified with the entry point names INTRO and OUTFROM

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | OWNCODE | (MRL,40),(2,INTRO),(4,OUTFROM) |

## TAPE

The TAPE macro call is required to specify all magnetic tape intermediate merge files used for the tape variant of Sort/Merge. Sort/Merge generates the necessary file information tables and fields. If the file names have not been defined previously (such as by the REQUEST or REWIND functions), Sort/Merge issues requests for scratch tapes to be assigned to the files as needed.

### BALANCED MERGE

A minimum of four tapes is required; the maximum is limited only by the number of tape drives available (appendix E). Balanced merging is more efficient if an even number of tapes is specified.

### POLYPHASE MERGE

A minimum of three tapes is required; the maximum is limited only by the number of tape drives available (appendix E).

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELDS |
|---|---|---|
| | TAPE | filename,filename,filename,... |

filename Name assigned to the intermediate merge file

### EXAMPLE

The names assigned to four intermediate merge files are INTERM1, INTERM2, INTERM3, INTERM4.

| LOCATION | OPERATION | VARIABLE SUBFIELD |
|---|---|---|
| | TAPE | INTERM1,INTERM2,INTERM3,INTERM4 |

## POINTER

The POINTER macro call is required when the SMLIST macro is used. POINTER must be included within the sequence containing the SORT macro. SMLIST instructs Sort/Merge to prepare to accept macros outside the macro sequence. One POINTER macro is required for each SMLIST macro. POINTER can not be used within an SMLIST parameter list.

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELD |
|---|---|---|
| | POINTER | addr |

addr Address of an SMLIST macro

### EXAMPLE

In this example, an SMLIST macro appears at location RMTLIST.

| LOCATION | OPERATION | VARIABLE SUBFIELD |
|---|---|---|
| | POINTER | RMTLIST |

## SMLIST

The SMLIST macro is processed only when a POINTER macro is encountered within a sequence following a SORT macro. SMLIST specifies the addresses of Sort/Merge macros that need not exist in the sequence following the SORT macro. For each macro address specified, Sort/Merge processes all contiguous Sort/Merge macros until it encounters an operation other than one of the macros listed below.

### FORMAT

| LOCATION | OPERATION | VARIABLE SUBFIELD |
|---|---|---|
| | SMLIST | macaddr,macaddr, . . . |

macaddr Address of Sort/Merge macro: BYTESIZE, FILES, KEY, SEQUENCE, EQUATE, OPTIONS, OWNCODE, or TAPE.

**EXAMPLE**

The OPTIONS, SEQUENCE, and LOGICAL KEY macros are used for two sorts, yet they are specified only once in the program. The LOGICAL key is the primary key for the first sort and the secondary key for the second sort. The second sort is not affected by the embedded SMLIST macro, since SMLIST macros are processed only when a POINTER macro is encountered.

```
LOCATION    OPERATION   VARIABLE SUBFIELDS                  COMMENTS
            SORT        MAXCM=0                             FIRST SORT
            POINTER     LISTEM
            KEY         1,1,5,0,DISPLAY,MYSEQ
              .
              .
              .
            SORT        MAXCM=0                             SECOND SORT
LISTEM      SMLIST      OPTM,SEQM
            KEY         10,1,4,0,DISPLAY,MYSEQ
            POINTER     LISTEM
              .
              .
              .
OPTM        OPTIONS     RETAIN
            KEY         73,,8,,LOGICAL
            CON         0                   TERMINATE MACRO SEQUENCE
SEQM        SEQUENCE    MYSEQ,(A,B,C,3,2,1),END
```

# FORTRAN EXTENDED CALLS 6

A set of library routines is provided for calling Sort/Merge from a FORTRAN Extended program. All are called by standard FORTRAN Extended CALL statements; all conventions for FORTRAN Extended statements must be observed.

The FORTRAN Extended calls and corresponding Sort/Merge directives are listed below:

| | |
|---|---|
| CALL SMSORT | SORT directive |
| CALL SMSORTB | SORT directive (Sort/Merge Version 4) |
| CALL SMSORTP | SORT directive (Sort/Merge Version 4) |
| CALL SMMERGE | MERGE directive |
| CALL SMFILE | FILE directive |
| CALL SMKEY | KEY directive |
| CALL SMSEQ | SEQUENCE directive |
| CALL SMEQU | EQUATE directive |
| CALL SMOPT | OPTIONS directive |
| CALL SMTAPE | TAPE directive (Sort/Merge Version 4) |
| CALL SMEND | END directive |
| CALL SMOWN | OWNCODE directive |
| CALL SMRTN | No corresponding directive |
| CALL SMABT | No corresponding directive |

The first call must be to SMSORT, SMSORTB, SMSORTP, or SMMERGE. The last call for any one sort must be to SMEND, which initiates processing using the information collected by the other calls.

Any Hollerith constants shown below as parameters delimited by the paired symbols ≠ ≠ can be indicated by the nHf (left justified with blank fill) representation of a Hollerith constant or a variable containing such a value. nRf and nLf representations are required in some instances, as noted below.

## FORTRAN EXTENDED CALLS TO SORT/MERGE

### SMSORT, SMSORTB, SMSORTP, AND SMMERGE

One of these calls must be the first call for any sort.

SMSORTB is used for a balanced tape sort, SMSORTP for a polyphased tape sort, and SMMERGE for merge only processing. All other sorts use the call to SMSORT.

### OPERATING SYSTEM INCOMPATIBILITIES

The tape variant of Sort/Merge is not supported under Sort/Merge Version 1. Therefore, SMSORTB and SMSORTP calls are allowed only under Version 4.

### FORMAT

CALL SMSORT(mrl,ba)
CALL SMSORTB(mrl,ba)
CALL SMSORTP(mrl,ba)
CALL SMMERGE(mrl,ba)

mrl Maximum length, in characters, of record to be sorted.

ba Optional total in decimal of large core memory (LCM) buffer area for SCOPE 2 Record Manager for all intermediate scratch files constructed internally by Sort/Merge. ba can be zero. Default is an installation parameter.

Optional number of words of central memory to be used by Sort/Merge for working storage. Default is 22000B.

## SMFILE

This call defines the names of all input and output files to be used during Sort/Merge processing. One call is needed for each unless the output file is handled by SMOWN, which requires no call to SMFILE.

A file should be positioned before any Sort/Merge processing.

### FORMAT

CALL SMFILE(smo,type,lfn,action)

smo Specifies file processing:

≠SORT≠
≠MERGE≠
≠OUTPUT≠

type Indicates type of input/output used for file access:

| | |
|---|---|
| ≠FORMATTED≠ or ≠CODED≠ | Formatted |
| ≠BINARY≠ | Unformatted |
| 0 (zero) | Record Manager interface routines |

lfn File name can be a tape number or the name left justified with zero fill (nLlfn). If type is zero, lfn is an array containing a FIT to be used in conjunction with Basic Access Methods (BAM) interface routines, and an OPENM must have been done on the file previously. Further, the file names must not appear on a PROGRAM statement. If type is not zero, the files specified by SMFILE must appear on the PROGRAM statement.

action Indicates action to be taken with the file upon Sort/Merge completion:

≠REWIND≠
≠UNLOAD≠
≠NONE≠ (Default)

#### EXAMPLE

The file TAPE1 is an input file for a sort. As no action is specified, the default (≠NONE≠) is used.

```
CALL SMFILE(≠SORT≠,≠FORMATTED≠,1)
```

## SMKEY

One SMKEY call is required to describe each sort key to be used. Sort keys are discussed more fully in section 2.

#### OPERATING SYSTEM INCOMPATIBILITIES

The separate sign feature for signed numeric data and the sign overpunch in the leading character position are supported under Sort/Merge Version 4; therefore, the LEADING, TRAILING, and SEPARATE options are applicable to Sort/Merge Version 4.

#### FORMAT

CALL SMKEY(bytepos,bitpos,nbyte,nbits,type,colseq,order)

bytepos, bitpos — Starting position of the sort key in relation to the first 6-bit byte of the record, counting from 1. bytepos gives the byte, bitpos gives the bit within the byte.

nbyte, nbits — Length of sort key in 6-bit bytes, or characters (nbyte), plus bits (nbits).

The remaining three parameters are optional:

type Specifies the type of code used to interpret keys. Type is a character expression having the following values:

| | |
|---|---|
| ≠DISPLAY≠ | Internal display code |
| ≠FLOAT≠ | Floating point data |
| ≠INTEGER≠ | Signed integer data |
| ≠LOGICAL≠ | Unsigned integer data (default) |

The identifiers ≠SIGN≠, ≠SEPARATE≠, ≠TRAILING≠, and ≠LEADING≠ must be preceded by type ≠DISPLAY≠; the identifiers must be separated by commas, as indicated:

| | |
|---|---|
| ≠DISPLAY≠, ≠SIGN≠ | Numeric data in display code; represented by an overpunch on the low order character of the key. |
| | Numeric data in display code; required if data has a sign overpunch. |
| ≠DISPLAY≠, ≠SIGN≠, ≠LEADING≠ | Numeric data in display code; + or - sign present as overpunch at beginning of the field. |
| ≠DISPLAY≠, ≠SIGN≠, ≠TRAILING≠ | Numeric data in display code; + or - sign present as an overpunch at the end of the field. |
| ≠DISPLAY≠, ≠SEPARATE≠, ≠LEADING≠ | Numeric data in display code; sign is a separate character at the beginning of the field. |
| ≠DISPLAY≠, ≠SEPARATE≠, ≠TRAILING≠ | Numeric data in display code; sign is a separate character at the end of the field. |

colseq Name of user supplied collating sequence defined by SMSEQ call, or one of the following collating sequences:

| | |
|---|---|
| ≠ASCII6≠ | 6-bit ASCII collating sequence (default for installations using ASCII character set) |
| ≠COBOL6≠ | 6-bit COBOL collating sequence (default for installations using CDC character set) |
| ≠DISPLAY≠ | Internal display code collating sequence |
| ≠INTBCD≠ | Internal BCD collating sequence |
| name | Name of a collating sequence specified by a call to SMSEQ |

A colseq parameter cannot be used unless the type parameter specifies ≠DISPLAY≠.

When a type parameter other than ≠DISPLAY≠ is used, the colseq parameter must be omitted. No indication of the omitted parameter is necessary.

order Sequencing order for sort processing. It can be either of the following:

| | |
|---|---|
| ≠A≠ | Ascending (default) |
| ≠D≠ | Descending |

### EXAMPLE

The SMKEY call example has the following specifications:

Sort key begins in byte 1, bit 1.

Key size is 20 bytes.

Sort key is coded in ≠DISPLAY≠.

Sort key is to be sequenced according to the installation default.

Sequencing is ascending.

CALL SMKEY(1,1,20,0,≠DISPLAY≠,≠A≠).

The order parameter follows the sort key parameter when colseq is not needed.

The following SMKEY example is invalid:

CALL SMKEY(13,6,10,0,≠INTEGER≠,0,≠A≠)

Sort/Merge interprets the second 0 (zero) as the name of a collating sequence because it is not a key type or ≠A≠ or ≠D≠. Inclusion of a collating sequence parameter with a non-character key type is invalid.

The correct calling sequence is:

CALL SMKEY(13,6,10,0,≠INTEGER≠,≠A≠)

## SMSEQ

This call specifies and names a user supplied collating sequence.

### FORMAT

CALL SMSEQ(colseq,array)

colseq — Names the collating sequence being defined. The collating sequence cannot be ≠ASCII6≠, ≠COBOL6≠, ≠DISPLAY≠ or ≠INTBCD≠.

array — Name of array containing characters in the order they are to be collated. Each character should be in nRx format (right justified with zero fill) or ijB format (octal). Unspecified characters collate high and equal. The collating sequence is terminated by a negative number.

### EXAMPLE

A new collating sequence is specified:

```
INTEGER COL(8)
DATA COL / 1RA, 1R, 1R1, 1R1, 57B, 1R$, 51B, -1/
. . .
CALL SMSEQ(≠MINE≠,COL)
```

## SMEQU

This call specifies two or more characters (already specified in a collating sequence) as equal when comparisons are made between these characters for Sort/Merge processing.

### FORMAT

```
CALL SMEQU(colseq,array)
```

colseq — Specifies the collating sequence containing the characters to be equated. The collating sequence cannot be ≠ASCII6≠, ≠COBOL6≠, ≠DISPLAY≠ or ≠INTBCD≠.

array — Name of array specifying the characters to be equated. Each character should be in nRx format (right justified with zero fill) or ijB format (octal). The end of the list of characters to be equated is indicated by a negative number.

### EXAMPLE

The characters A, B, and 1 in the collating sequence ≠MINE≠ are equated:

```
INTEGER EQ(4)
DATA EQ / 1RA, 1RB, 1R1, -1/
. . .
CALL SMEQU(≠MINE≠,EQ)
```

## SMOPT

This call specifies special record handling options or operations for Sort/Merge processing.

### OPERATING SYSTEM INCOMPATIBILITIES

The checkpoint dump features of Sort/Merge are supported only by Sort/Merge Version 4; therefore the ≠VOLDUMP≠, ≠DUMP≠, and ≠NODUMP≠ options are applicable only for the option parameter of the SMOPT call under Sort/Merge Version 4.

The ≠ORDER≠, ≠NODAY≠, ≠COMPARE≠, and ≠EXTRACT≠ options are also applicable only under Sort/Merge Version 4.

If SMOPT is called under Sort/Merge Version 1, the call must be made immediately after the call to SMSORT or SMMERGE.

**FORMAT**

CALL SMOPT(option$_1$, . . . , option$_n$)

option$_i$ Any of the following record handling options may be specified:

| | |
|---|---|
| ≠VERIFY≠ | Output file is checked for correct sequencing. If the records are not in correct order, the job terminates and the output file is lost. This option verifies that records from the merge input file or inserted through owncode exits 3, 4, and 5 are in order; it need never be specified for a sort with no owncodes.<br><br>If this parameter is omitted, a sequencing error during a merge run with no inserted records produces a nonfatal error message. All records are written to the output file, but they will not be in order. When records are inserted, no checking takes place; thus, omission of this parameter allows the user to insert records (such as page headers) out of sequence. |
| ≠RETAIN≠ | Records with identical sort keys that are read from sort input files are written in the order they are read. That is, all the records from each file are grouped together. Records from merge files are sequenced arbitrarily. When this parameter is omitted, records with identical sort keys are sequenced arbitrarily. |
| ≠VOLDUMP≠ | A checkpoint dump is taken at end-of-volume condition on the input file or new-volume condition on the output file. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files. |
| ≠DUMP≠,n | A checkpoint dump is taken when n records have been read from the input file or written to the output file. n is decimal. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files.<br><br>If n is not specified, a checkpoint dump is taken after 50,000 records are read or written. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual for information about requesting checkpoint files. |
| ≠NODUMP≠ | No checkpoint dumps are taken; default. |
| ≠NODAY≠ | Dayfile messages are suppressed. |

| | |
|---|---|
| ≠ORDER≠,mo | mo specifies the intermediate merge order; $2 \leq mo \leq 64$. Merge order is explained in appendix F. If core is not sufficient to merge at the order specified, a fatal error occurs and a diagnostic indicates how much additional core is required. When merge order is not specified, it is calculated by Sort/Merge based on the amount of memory available. |
| ≠COMPARE≠ | The key comparison technique is to be used. This technique usually requires less elapsed time and more central processing (CP) time than key extraction. |
| ≠EXTRACT≠ | The key extraction technique is to be used. This technique usually requires less CP time and more elapsed time than key comparison. |

The COMPARE and EXTRACT options are mutually exclusive within a single sort. If neither option is specified, Sort/Merge attempts to choose the best technique.

If SMOPT is called more than once, only the last call applies.

#### EXAMPLE

This example requests the ≠VERIFY≠ option and a checkpoint dump after each 10,000 records are read from the input file or written to the output file.

```
CALL SMOPT(≠VERIFY≠,≠DUMP≠,10000)
```

## SMTAPE

This call is required for the tape variant of Sort/Merge Version 4 to specify all magnetic tape intermediate merge files. If the tape files have not been defined in a previous job step (REQUEST, REWIND, etc.), Sort/Merge issues requests for scratch tape assignment as needed for intermediate files.

A minimum of four tapes is required for a balanced merge; this merge is more efficient if an even number of tapes is specified.

A minimum of three tapes is required for a polyphased merge. The maximum number of tapes for either type of merge is limited only by the number of tape drives available.

#### OPERATING SYSTEM INCOMPATIBILITIES

The tape variant of Sort/Merge is not supported under Version 1; therefore, the SMTAPE call is allowed only under Sort/Merge Version 4.

#### FORMAT

CALL SMTAPE($lfn_1$, . . . , $lfn_n$)

| | |
|---|---|
| $lfn_i$ | Name assigned to intermediate merge file. Each file name must be in nLlfn format (left justified with zero fill) and must not be defined by the FORTRAN Extended program. A maximum of 63 file names may be specified. |

**EXAMPLE**

Names are assigned to four intermediate merge files:

```
CALL SMTAPE(5LPOPPY,5LPANSY,5LDAISY,5LVIOLA)
```

## SMEND

This call initiates Sort/Merge processing. It must be the last call for any one sort or merge.

### FORMAT

```
CALL SMEND
```

This call has no parameters.

## SMOWN AND SMRTN

The call to SMOWN allows owncode routines to be set up. This call provides the capability to insert, substitute, modify, or delete input and output records.

If the owncode routine is a FORTRAN Extended subroutine, the call to SMRTN is used to return from the owncode subroutine and resume Sort/Merge processing. Specific processing action can be requested from Sort/Merge by altering the return address with a parameter on the SMRTN call.

An owncode routine must be supplied for each owncode exit specified in the call to SMOWN. Exits that may be specified and the use of COMPASS owncode routines are discussed in section 3.

### FORMAT

CALL SMOWN($exitnum_1$,$subname_1$, . . . ,$exitnum_n$,$subname_n$)

$exitnum_i$ Number of owncode exit to be taken.

$subname_i$ Name of the user-supplied owncode exit subroutine to be called when $exitnum_i$ is taken.

Each name specified in a call to SMOWN must appear in an EXTERNAL statement in the calling program. For each subname specified, the user must supply a subroutine that exits through a call to system subroutine SMRTN, in accordance with the owncode exit number and return address as follows:

| Exitnum | Entry | Exit |
|---|---|---|
| 1 or 3 | SUBROUTINE subname(a,rl) | CALL SMRTN(retaddr), for retaddr=1 or 3 |
| | | CALL SMRTN(retaddr,b,rl), for retaddr=0 or 2 |

| Exitnum | Entry | Exit |
|---|---|---|
| 2 or 4 | SUBROUTINE subname | CALL SMRTN(retaddr), for retaddr=0 |
| | | CALL SMRTN(retaddr,b,rl), retaddr=1 |
| 5 | SUBROUTINE subname($a_1$,$rl_1$,$a_2$,$rl_2$) | CALL SMRTN($b_1$,$rl_1$,$b_2$,$rl_2$), for retaddr=0 |
| | | CALL SMRTN($b_1$,$rl_1$), for retaddr=1 |

a — Integer array of length rl+9/10 in which Sort/Merge stores a record when subname is called. Storing into array a causes indeterminate results.

b — Integer array of length rl+9/10 in which the user stores a record when subname is called. Array b should not be the same as a.

rl — Record length in characters.

retaddr — Alters the normal return address used to resume Sort/Merge processing as follows:

| retaddr | Return address: |
|---|---|
| 0 | Normal return address |
| 1 | Normal return address +1 |
| 2 | Normal return address +2 |
| 3 | Normal return address +3 |

The retaddr parameter of the SMRTN call determines the processing action requested. Actions that can be requested are discussed in section 3. These actions are summarized as follows:

| Processing Action | Exit 1 | Exit 2 | Exit 3 | Exit 4 | Exit 5 |
|---|---|---|---|---|---|
| Substitute a record | retaddr | | retaddr | | retaddr |
| Insert a record | retaddr+2 | retaddr+1 | retaddr+2 | retaddr+1 | |
| Delete a record | retaddr+1 | | retaddr+1 | | retaddr+1 |
| Terminate a file | retaddr+3 | | retaddr+3 | | |
| Normal processing | retaddr | retaddr | retaddr | retaddr | retaddr |

When the processing action requested is file termination, the current record in array a is not included in Sort/Merge processing.

Special factors to consider when using Exit 5 owncode routines are:

If SMRTN is called from an Exit 5 owncode routine, the number of parameters on the SMRTN call determines the processing action requested.

If an Exit 5 owncode routine is used for record subsitution, the record contained in both area a and array b can be substituted with both new address and new field length specifications.

If an Exit 5 owncode routine is used to delete a record, the record contained in array b is deleted.

## SMABT

This call terminates a sequence of Sort/Merge interface calls without initiating an execution of Sort/Merge. The state of the interface is the same as if no calls had been made.

The format of the SMABT subroutine is as follows:

```
CALL SMABT
```

This call has no parameters.

### EXAMPLE

```
CALL SMOWN(3,SUB3)
. . .
SUBROUTINE SUB3(X,N)
DIMENSION X(20)
. . .
CALL SMRTN(1)
. . .
```

## SAMPLE PROGRAM

The following sample program adds new records to a master file, merges two files containing updates to the master file, updates the master file, and produces a report.

The master file is a list of students by name and student number, along with the grade to date. The updates are the results of an exam. The updated grade is used to produce the report.

A main program and two subprograms, one of them an owncode routine, were used:

```
      PROGRAM GRADES(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,
     *               CLASS1,TAPE11=CLASS1,CLASS2,TAPE21=CLASS2,
     *               MASTER,TAPE31=MASTER,TAPE22,TAPE32,TAPE33,TAPE34)
      IMPLICIT INTEGER (A-Z)
      COMMON /OWN/ NCT, RECRL
      DIMENSION NAME(2), CURVE(5,2)
      EXTERNAL UPDATE
      DATA (CURVE(J,2), J = 1, 5) /1HA, 1HB, 1HC, 1HD, 1HF/
      DATA NCT /0/

      REWIND 11
      REWIND 21
      REWIND 31
      READ(5,1000) CSIZE, NCT, RECRL
      CSIZE = CSIZE + NCT

      CALL SMSORT(20)
      CALL SMFILE("SORT","CODED",11,"REWIND")
      CALL SMFILE("SORT","CODED",21,"REWIND")
      CALL SMFILE("OUTPUT","CODED",22,"REWIND")
      CALL SMKEY(1,1,6,0,"DISPLAY")
      CALL SMEND

      CALL SMSORT(RECRL)
      CALL SMFILE("SORT","CODED",31,"REWIND")
      CALL SMFILE("OUTPUT","CODED",32,"REWIND")
      CALL SMKEY(21,1,6,0,"DISPLAY")
      CALL SMOWN(2,UPDATE)
      CALL SMEND
```

```
      DO 100 I = 1, CSIZE
      READ(22, 1001) STU1, GRD1
      READ(32, 1002) NAME(1), NAME(2), STU2, GRD2
      IF (STU1 .NE. STU2) CALL ERRORS(I, STU1, STU2)
      GRD3 = GRD2 + GRD1
      WRITE(33, 1002) NAME(1), NAME(2), STU2, GRD3
  100 CONTINUE
      REWIND 32

      CALL SMSORT(RECRL)
      CALL SMFILE("SORT","CODED",33,"REWIND")
      CALL SMFILE("OUTPUT","CODED",34,"REWIND")
      CALL SMKEY(31,1,4,0,"DISPLAY","D")
      CALL SMKEY(1,1,10,0,"DISPLAY")
      CALL SMKEY(21,1,6,0,"DISPLAY")
      CALL SMEND

      CURVE(1,1) = CURVE(5,1) = .15 * CSIZE + .5
      CURVE(2,1) = CURVE(4,1) = .20 * CSIZE + .5
      CURVE(3,1) =              .30 * CSIZE + .5
      WRITE(6, 2002)

      DO 102 J = 1, 5
      LIMIT = CURVE(J, 1)
      LETTER = CURVE(J, 2)
      DO 101 K = 1, LIMIT
      READ(34, 1002) NAME(1), NAME(2), STU2, GRD2

      WRITE(6,2001) NAME(1), NAME(2), STU2, GRD2, LETTER
  101 CONTINUE
      IF (J .EQ. 3) WRITE(6,2003)
  102 CONTINUE

      STOP

 1000 FORMAT ( 3I3 )
 1001 FORMAT ( A6, 4X, I3 )
 1002 FORMAT ( 2A10, A6, 4X, I4 )

 2001 FORMAT ( 7X, 2A10, A6, 11X, I4, 5X, A1 )
 2002 FORMAT ( 1H1, 6X, "NAME", 12X, "STUDENT NUMBER", 7X, "SCORE", 3X,
     *         "GRADE" // 5X, "PASSING" / )
 2003 FORMAT ( / 5X, "FAILING" / )

      END
```

```
      SUBROUTINE UPDATE(ARRAY, ARL)
      IMPLICIT INTEGER (A-Z)
      COMMON /OWN/ NCT,RECRL
      DIMENSION ARRAY(4), RECORD(4)

      IF (NCT .EQ. 0) CALL SMRTN(0)
      NCT = NCT - 1
      READ (5, 1001) (RECORD(K), K = 1, 4)
      CALL SMRTN(1, RECORD, RECRL)

      STOP 77

1001  FORMAT ( 4A10 )

      END


      SUBROUTINE ERRORS(M, NO1, NO2)
      CALL REMARK(35H      STUDENT NUMBERS DO NOT MATCH  )
      CALL DISPLA(29H      FIRST STUDENT NUMBER IS , NO1)
      CALL DISPLA(29H      SECOND STUDENT NUMBER IS, NO2)
      CALL DISPLA(21H      ITERATION EQUALS, M)
      CALL REMARK(20H      JOB TERMINATED  )
      CALL EXIT
      END
```

The following files were used as input to the program:

```
MASTER  SMITH,JOHN              105432    271
        BROWN,WILLIAM           113751    193
        DOE,CHERYL              100222    221
        JONES,CHRISTOPHER       115741    254
        NORTH,PATRICIA          107043    242
        GORDON,ROBERT           102778    178
        NOE,JANE                113542    156
        WATSON,JOSEPH           102782    267
        HARRIS,JEAN             114154    169
        CARTER,BARBARA          116315    234
        SHARP,DANIEL            109467    217
        SMITH,CATHERINE         111696    187
        ROLLINS,SARAH           107461    177
        ANDERSON,PETER          103167    184
        LYNCH,DENNIS            112373    192
        HALL,MARY               104725    223
        HENDERS,GERALD          118538    249
        THOMPSON,LOUISE         101886    211
```

```
INPUT      18  5 40
           PETERS,SANDRA              103759      182
           GRAY,HENRY                 115254      168
           MARTIN,PATRICIA            111111      300
           ANDERSON,WILLIAM           111112      299
           ANDERSON,JAMES             111113      299

CLASS1     105432      97
           102778      91
           113751      89
           108467      87
           113542      82
           102782      77
           100222      74
           111696      69
           114154      68
           115741      63
           107043      59
           111113      96
           116315      55

CLASS2     103759      81
           107461      62
           103167      74
           112373      83
           104725      91
           118538      87
           101886      68
           115254      79
           111111      98
           111112      96
```

The program was run using the following control statements:

```
FTN.

LGO.
```

The following report was produced:

| NAME | STUDENT NUMBER | SCORE | GRADE |
|---|---|---|---|
| PASSING | | | |
| MARTIN,PATRICIA | 111111 | 398 | A |
| ANDERSON,JAMES | 111113 | 395 | A |
| ANDERSON,WILLIAM | 111112 | 395 | A |
| SMITH,JOHN | 105432 | 368 | B |
| WATSON,JOSEPH | 102782 | 344 | B |
| HENDERS,GERALD | 118538 | 336 | B |
| JONES,CHRISTOPHER | 115741 | 317 | B |
| HALL,MARY | 104725 | 314 | B |
| SHARP,DANIEL | 108467 | 304 | C |
| NORTH,PATRICIA | 107043 | 301 | C |
| DOE,CHERYL | 100222 | 295 | C |
| CARTER,BARBARA | 116315 | 289 | C |
| BROWN,WILLIAM | 113751 | 282 | C |
| THOMPSON,LOUISE | 101886 | 279 | C |
| LYNCH,DENNIS | 112373 | 275 | C |
| FAILING | | | |
| GORDON,ROBERT | 102778 | 269 | D |
| PETERS,SANDRA | 103759 | 263 | D |
| ANDERSON,PETER | 103167 | 258 | D |
| SMITH,CATHERINE | 111696 | 256 | D |
| GRAY,HENRY | 115254 | 247 | D |
| ROLLINS,SARAH | 107461 | 239 | F |
| NOE,JANE | 113542 | 238 | F |
| HARRIS,JEAN | 114154 | 237 | F |

Four scratch files were used; when the job finished execution, they were as follows:

TAPE 32

| | | |
|---|---|---|
| DOE,CHERYL | 100222 | 221 |
| THOMPSON,LOUISE | 101886 | 211 |
| GORDON,ROBERT | 102778 | 178 |
| WATSON,JOSEPH | 102782 | 267 |
| ANDERSON,PETER | 103167 | 184 |
| PETERS,SANDRA | 103759 | 182 |
| HALL,MARY | 104725 | 223 |
| SMITH,JOHN | 105432 | 271 |
| NORTH,PATRICIA | 107043 | 242 |
| ROLLINS,SARAH | 107461 | 177 |
| SHARP,DANIEL | 108467 | 217 |
| MARTIN,PATRICIA | 111111 | 300 |
| ANDERSON,WILLIAM | 111112 | 299 |
| ANDERSON,JAMES | 111113 | 299 |
| SMITH,CATHERINE | 111696 | 187 |
| LYNCH,DENNIS | 112373 | 192 |
| NOE,JANE | 113542 | 156 |
| BROWN,WILLIAM | 113751 | 193 |
| HARRIS,JEAN | 114154 | 169 |
| GRAY,HENRY | 115254 | 168 |
| JONES,CHRISTOPHER | 115741 | 254 |
| CARTER,BARBARA | 116315 | 234 |
| HENDERS,GERALD | 118538 | 249 |

TAPE 22

| | |
|---|---|
| 100222 | 74 |
| 101886 | 68 |
| 102778 | 91 |
| 102782 | 77 |
| 103167 | 74 |
| 103759 | 81 |
| 104725 | 91 |
| 105432 | 97 |
| 107043 | 59 |
| 107461 | 62 |
| 108467 | 87 |
| 111111 | 98 |
| 111112 | 96 |
| 111113 | 96 |
| 111696 | 69 |
| 112373 | 83 |
| 113542 | 82 |
| 113751 | 89 |
| 114154 | 68 |
| 115254 | 79 |
| 115741 | 63 |
| 116315 | 55 |
| 118538 | 87 |

TAPE33

| | | |
|---|---|---|
| DOE,CHERYL | 100222 | 295 |
| THOMPSON,LOUISE | 101886 | 279 |
| GORDON,ROBERT | 102778 | 269 |
| WATSON,JOSEPH | 102782 | 344 |
| ANDERSON,PETER | 103167 | 258 |
| PETERS,SANDRA | 103759 | 263 |
| HALL,MARY | 104725 | 314 |
| SMITH,JOHN | 105432 | 368 |
| NORTH,PATRICIA | 107043 | 301 |
| ROLLINS,SARAH | 107461 | 239 |
| SHARP,DANIEL | 108467 | 304 |
| MARTIN,PATRICIA | 111111 | 398 |
| ANDERSON,WILLIAM | 111112 | 345 |
| ANDERSON,JAMES | 111113 | 345 |
| SMITH,CATHERINE | 111696 | 256 |
| LYNCH,DENNIS | 112373 | 275 |
| NOE,JANE | 113542 | 238 |
| BROWN,WILLIAM | 113751 | 282 |
| HARRIS,JEAN | 114154 | 237 |
| GRAY,HENRY | 115254 | 247 |
| JONES,CHRISTOPHER | 115741 | 317 |
| CARTER,BARBARA | 116315 | 289 |
| HENDERS,GERALD | 118538 | 336 |

TAPE34

| | | |
|---|---|---|
| MARTIN,PATRICIA | 111111 | 398 |
| ANDERSON,JAMES | 111113 | 345 |
| ANDERSON,WILLIAM | 111112 | 345 |
| SMITH,JOHN | 105432 | 368 |
| WATSON,JOSEPH | 102782 | 344 |
| HENDERS,GERALD | 118538 | 336 |
| JONES,CHRISTOPHER | 115741 | 317 |
| HALL,MARY | 104725 | 314 |
| SHARP,DANIEL | 108467 | 304 |
| NORTH,PATRICIA | 107043 | 301 |
| DOE,CHERYL | 100222 | 295 |
| CARTER,BARBARA | 116315 | 289 |
| BROWN,WILLIAM | 113751 | 282 |
| THOMPSON,LOUISE | 101886 | 279 |
| LYNCH,DENNIS | 112373 | 275 |
| GORDON,ROBERT | 102778 | 269 |
| PETERS,SANDRA | 103759 | 263 |
| ANDERSON,PETER | 103167 | 258 |
| SMITH,CATHERINE | 111696 | 256 |
| GRAY,HENRY | 115254 | 247 |
| ROLLINS,SARAH | 107461 | 239 |
| NOE,JANE | 113542 | 238 |
| HARRIS,JEAN | 114154 | 237 |

# FORTRAN 5 CALLS 7

Fortran 5 provides the capability for processing data records under Sort/Merge. Fortran 5 interfaces with Sort/Merge through the subroutines described in this section. Because Sort/Merge uses the unused part of the field length as a scratch area, the ARG=FIXED control statement option is not permitted for programs using Sort/Merge. All conventions for FORTRAN 5 statements must be observed.

The Fortran 5 calls and corresponding Sort/Merge directives are as follows:

| | |
|---|---|
| CALL SMSORT | SORT directive |
| CALL SMSORTB | SORT directive (Sort/Merge Version 4) |
| CALL SMSORTP | SORT directive (Sort/Merge Version 4) |
| CALL SMMERGE | MERGE directive |
| CALL SMFILE | FILE directive |
| CALL SMKEY | KEY directive |
| CALL SMSEQ | SEQUENCE directive |
| CALL SMEQU | EQUATE directive |
| CALL SMOPT | OPTIONS directive |
| CALL SMTAPE | TAPE directive (Sort/Merge Version 4) |
| CALL SMEND | END directive |
| CALL SMOWN | OWNCODE directive |
| CALL SMRTN | No corresponding directive |
| CALL SMABT | No corresponding directive |

## FORTRAN 5 CALLS TO SORT/MERGE

The series of calls to Sort/Merge subroutines must begin with a call to SMSORT, SMSORTB, SMSORTP or SMMERGE. If a file is processed by CYBER Record Manager subroutines, OPENM should be called before any of these routines. The last call for any one sort must be to SMEND, which intiates processing using the information collected by the other calls.

In an overlay structured program using blank common, the Sort/Merge interface routines must not be called from the 0,0 overlay.

### SMSORT, SMSORTB, SMSORTP, AND SMMERGE

One of these calls must be the first call for any sort.

SMSORTB is used for a balanced tape sort, SMSORTP for a polyphased sort, and SMMERGE for merge only processing. All other sorts use the call to SMSORT.

## OPERATING SYSTEM INCOMPATIBILITIES

The tape variant of Sort/Merge is not supported under Sort/Merge Version 1. Therefore, SMSORTB and SMSORTP are allowed only under Version 4.

## FORMAT

CALL SMSORT (mrl, ba)

CALL SMSORTB (mrl, ba)

CALL SMSORTP (mrl, ba)

CALL SMMERGE (mrl, ba)

mrl Maximum length in characters of record to be sorted.

ba Optional total in decimal of large core memory (LCM) buffer area for SCOPE 2 Record manager for all intermediate scratch files constructed internally by Sort/Merge. ba can be zero. Default is the installation parameter.

Optional number of words of central memory to be used by Sort/Merge for working storage. Default is 22000B.

## SMFILE

This call defines the names of all input and output files to be used during Sort/Merge processing. SMFILE must be called for each file to be sorted or merged and must be called once for the file to receive the output (unless SMOWN is called).

Files should be properly positioned before any Sort/Merge processing.

## FORMAT

CALL SMFILE (dis, i/o, lfn, action)

dis Character expression indicating file processing:

| | |
|---|---|
| 'SORT' | File is to be sorted. |
| 'MERGE' | File is to be merged. |
| 'OUTPUT' | File is to receive output. |

i/o Character expression indicating mode of file input/output:

| | |
|---|---|
| 'FORMATTED' | File accessed with formatted input/output. |
| 'CODED' | File accessed with formatted input/output. |
| 'BINARY' | File accessed with unformatted input/output. |
| 0 (zero) | File accessed with interfacing Record Manager Subroutines. |

lfn Character or boolean file name indicator:

| | |
|---|---|
| u | Logical unit number, 0 through 99 |
| L "filename" | File name left justified with zero fill |
| fit | Array containing the file information table when i/o is specified as 0 (zero) |

action Character expression indicating the action to be taken for the file upon Sort/Merge completion:

'REWIND'

'UNLOAD'

'NONE' (default)

## EXAMPLE

The file TAPE1 is an input file for a sort. The file is to be rewound upon Sort/Merge completion.

```
CALL SMFILE ('SORT','FORMATTED',1,'REWIND')
```

## SMKEY

This call describes the sort key to be used. One SMKEY is required for each key. The first call for each file indicates the major key; subsequent calls indicate addtional or minor keys in the order encountered. Sort keys are discussed more fully in section 2.

## OPERATING SYSTEM INCOMPATIBILITIES

The separate sign feature for signed numeric data and the sign overpunch in the leading character position are supported under Sort/Merge Version 4; therefore, the LEADING, TRAILING, and SEPARATE options are applicable to Sort/Merge Version 4 only.

## FORMAT

```
CALL SMKEY (charpos,bitpos,nchar,nbits,type,colseq,order)
```

charpos Integer specifying the relation of the first character of the sort key to the first character of the record. The first character of the record is in position 1.

bitpos Integer specifying the position of the first bit of the sort key of the character (or 6-bit byte) specified by charpos. The first bit of the record is considered bit number 1.

nchar Integer specifying the number of characters or complete 6-bit bytes in the sort key.

nbits Integer specifying the number of bits in the sort key in excess of those indicated by nchar.

The remaining three parameters are optional:

type Specifies the type of code used to interpret keys. Type is a character expression having the following values:

| | |
|---|---|
| 'DISPLAY' | Internal display code. |

| | |
|---|---|
| 'FLOAT' | Floating point data |
| 'INTEGER' | Signed integer data |
| 'LOGICAL' | Unsigned integer data (default) |

The identifiers 'SIGN', 'SEPARATE', 'TRAILING' and 'LEADING' must be preceded by type 'DISPLAY'; the identifiers must be separated by commas, as indicated:

| | |
|---|---|
| 'DISPLAY','SIGN' | Numeric data in display code; represented by overpunch on low order character of the key. |
| | Numeric data is display code; required if data has a sign overpunch. |
| 'DISPLAY','SIGN', 'LEADING' | Numeric data in display code; + or - sign present as overpunch at beginning of the field. |
| 'DISPLAY','SIGN', 'TRAILING' | Numeric data in display code; + or - sign present as an overpunch at the end of the field. |
| 'DISPLAY', 'SEPARATE', 'LEADING' | Numeric data in display code, sign is a separate character at beginning of the field. |
| 'DISPLAY', 'SEPARATE', 'TRAILING' | Numeric data in display code; sign is a separate character at the end of the field. |

colseq Name of user-supplied collating sequence defined by SMSEQ call, or one of the following collating sequences:

| | |
|---|---|
| 'ASCII6' | 6-bit ASCII collating sequence (default for installations using ASCII collating character set). |
| 'COBOL6' | 6-bit COBOL collating sequence (default for installations using CDC character set). |
| 'DISPLAY' | Internal display collating sequence. |
| 'INTBCD' | Internal BCD collating sequence. |
| segname | Name of a user supplied collating sequence specified in a call to SMSEQ. |

A colseq parameter cannot be used unless the type parameter specifies 'DISPLAY'. When a type parameter other than 'DISPLAY' is used, the colseq parameter must be omitted. No indication of the missing parameter is necessary.

order Character expression specifying the order of sort processing. It can be either of the following:

| | |
|---|---|
| 'A' | Ascending (default) |
| 'D' | Descending |

### EXAMPLE

In the following SMKEY example, the first two parameters describe a sort key beginning in position 1, bit 1:

```
CALL SMKEY(1,1,20,0,'DISPLAY','A')
```

Other parameters specify the number of characters in the sort key (exactly 20), the sort key type (display), and the order of processing (ascending). The colseq parameter is omitted; therefore, the sort key is collated according to installation default. The order parameter directly follows the sort key parameter when colseq is omitted.

The following SMKEY example is invalid:

```
CALL SMKEY (13,6,10,0,'INTEGER',0,'A')
```

The second 0 will be intepreted as the name of a collating sequence because it is not a key type or 'A' or 'D'. Inclusion of a collating sequence parameter with a non-character key type is invalid. The correct calling sequence for the example is:

```
CALL SMKEY (13,6,10,0,'INTEGER','A')
```

## SMSEQ

This call specifies and names a user-supplied collating sequence.

### FORMAT

```
CALL SMSEQ (segname,segspec)
```

segname Names the user-supplied collating sequence being defined. The collating sequence cannot be 'ASCII6', 'COBOL6', 'DISPLAY' or 'INTBCD'.

segspec Names the integer array containing characters in the order they are to be collated. Each character should be in nR"s" format (right justified with zero fill) or O"o" format (octal). Unspecified characters collate high and equal. The collating sequence is terminated by a negative number.

### EXAMPLE

A new collating sequence is specified:

```
INTEGER COL(8)
DATA COL/R"A",R"1",R"I",O"57",R"$",O"51",-1/
    .
    .
    .
CALL SMSEQ('MINE',COL)
```

## SMEQU

This call specifies that two or more characters already specified in a user's collating sequence are equal for comparison.

### FORMAT

CALL SMEQU(colseg,equspec)

colseg — Specifies the user collating sequence determined by a previous call to SMKEY or SMSEQ. The collating sequence cannot be 'ASCII6', 'COBOL6', 'DISPLAY', or 'INTBCD'.

equspec — Defines the name of an integer array that specifies the characters to be equated. Each character should be in nR"s" format (right justified with zero fill) or O"o" format (octal). The end of the list of characters to be equated is indicated by a negative number.

### EXAMPLE

The characters A, B, and 1 in the collating sequence 'MINE' are equated:

```
INTEGER EQ(4)
DATA EQ/R"A",R"B",R"1",-1/
    .
    .
    .
CALL SMEQU('MINE',EQ)
```

## SMOPT

This call specifies special record handling options or operations for Sort/Merge processing.

### OPERATING SYSTEM INCOMPATIBILITIES

The checkpoint dump features of Sort/Merge are supported only by Sort/Merge Version 4; therefore the 'VOLDUMP', 'DUMP', and 'NODUMP' options are applicable only for the option parameter of the SMOPT call under Sort/Merge Version 4.

The 'ORDER','NODAY', 'COMPARE', and 'EXTRACT' options are also applicable only under Sort/Merge Version 4.

If SMOPT is called under Sort/Merge Version 1, the call must be made immediately after the call to SMSORT or SMMERGE.

### FORMAT

CALL SMOPT(opt ,opt ...)

opt — Any of the following nonordered options can be specified:

'VERIFY' — Output file is checked for correct sequencing. If the records are not in correct order, the job terminates and the output file is lost. This option verifies that records from the merge input file or inserted through owncode exits 3, 4, and 5 are in order. 'Verify' need never be specified for a sort with no owncodes.

'RETAIN' — Records with identical sort keys that are read from sort input files are written in the order in which they are read. All the records from each file are grouped together. Records from merge files are sequenced arbitrarily. When this parameter is omitted, records with identical sort keys are sequenced arbitrarily.

| | |
|---|---|
| 'VOLDUMP' | A checkpoint dump is taken at end-of-volume condition on the input file or new-volume condition on the output file. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual. |
| 'DUMP' | A checkpoint dump is taken after 50,000 records are read or written. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual. |
| 'DUMP',n | A checkpoint dump is taken when n records have been read from the input file or written to the output file. n is decimal. A checkpoint file must have been previously requested. See the NOS or NOS/BE reference manual. |
| 'NODUMP' | No checkpoint dumps are taken; default. |
| 'NODAY' | Dayfile messages are suppressed. |
| 'ORDER',mo | mo specifies the intermediate merge order; $2 \leq mo \leq 64$. Merge order is explained in appendix F. If core is not sufficient to merge at the order specified, a fatal error occurs and a diagnostic indicates how much additional core is required. When merge order is not specified, it is calculated by Sort/Merge based on the amount of memory available. |
| 'COMPARE' | The key comparison sorting technique is to be used. This technique usually requires less elapsed time and more central processing (CP) time than key extraction. |
| 'EXTRACT' | The key extraction technique is to be used. This technique usually requires less CP time and more elapsed time than key comparison. |

The 'COMPARE' and 'EXTRACT' options are mutually exclusive. If neither option is specified, Sort/Merge attempts to choose the best technique.

### EXAMPLE

This example requests the 'VERIFY' option and a checkpoint dump after each 10,000 records are read from the input file or written to the output file.

```
CALL SMOPT('VERIFY', 'DUMP',10000)
```

If SMOPT is called more than once, only the last call is processed.

## SMTAPE

This call is required for the tape variant of Sort/Merge Version 4 to specify all magnetic tape intermediate merge files. If the tape files have not been defined in a previous job step (LABEL, REWIND, etc.), Sort/Merge issues requests for scratch tape assignment as needed for intermediate files.

A minimum of four tapes is required for a balanced merge; this merge is more efficient if an even number of tapes is specified.

A minimum of three tapes is required for a polyphased merge. The maximum number of tapes for either type of merge is limited only by the number of tape drives available.

### OPERATING SYSTEM INCOMPATIBILITIES

The tape variant of Sort/Merge is not supported under Version 1; therefore, the SMTAPE call is allowed only under Sort/Merge Version 4.

### FORMAT

CALL SMTAPE(taplist)

taplist List of logical file names, each in the form L"filename", to be used in balanced or polyphase tape merge. The file names in taplist must not be declared in the PROGRAM statement. A maximum of 63 file names can be specified.

### EXAMPLE

Names are assigned to four intermediate merge files.

```
CALL SMTAPE(5LPOPPY,5LPANSY,5LDAISY,5LVIOLA)
```

## SMEND

This call initiates Sort/Merge processing. It must be the last call for any one sort or merge. The format for the call SMEND subroutine is as follows:

CALL SMEND

This call has no parameters.

## SMOWN

The call to SMOWN allows owncode routines to be set up. This call provides the capability to insert, substitute, modify, or delete input and output records.

If the owncode routine is a FORTRAN 5 subroutine, a call to SMRTN is used to return from the owncode subroutine and resume Sort/Merge processing. Specific processing action can be requested from Sort/Merge by altering the return address with a parameter on the SMRTN call.

An owncode routine must be supplied for each owncode exit specified in the call to SMOWN. Exits that can be specified and the use of COMPASS owncodes are discussed in section 3.

### FORMAT

CALL SMOWN($exitnum_1$,$subname_1$,$exitnum_n$,$subname_n$ ...)

exitnum Number of the owncode exit.

subname Name of the user-supplied owncode exit routine.

Each subname specified in a call to SMOWN must appear in an EXTERNAL statement in the calling program. For each subroutine specified, the user must specify a subroutine that exits through a call to system subroutine SMRTN, in accordance with the owncode exit number and return address as follows:

| Exitnum | Entry | Exit |
|---|---|---|
| 1 or 3 | SUBROUTINE subname (a,rl) | CALL SMRTN (retaddr), for retaddr=1 or 3 |
| | | CALL SMRTN (retaddr,b,rl), for retaddr=0 or 2 |
| 2 or 4 | SUBROUTINE subname | CALL SMRTN (retaddr), for retaddr=0 |
| | | CALL SMRTN(retaddr,b,rl), for retaddr=1 |
| 5 | SUBROUTINE subname($a_1$,$rl_1$,$a_2$,$rl_2$) | CALL SMRTN($b_1$,$rl_1$,$b_2$,$rl_2$), for retaddr=1 |
| | | CALL SMRTN($b_1$,$rl_1$), for retaddr=1 |

a — Integer array of length (rl +9)/10 in which Sort/Merge stores a record when subname is called. Storing into array a causes indeterminate results.

b — Integer array of length (rl +9)/10 in which the user stores a record when subname is called. Array b should not be the same as array a.

rl — Record length in characters.

retaddr — Alters the normal return address used to resume Sort/Merge processing as follows:

| retaddr | Return address: |
|---|---|
| 0 | Normal return address |
| 1 | Normal return address +1 |
| 2 | Normal return address +2 |
| 3 | Normal return address +3 |

The retaddr parameter of the SMRTN call determines the processing action requested. Actions that can be requested are discussed in section 3. These actions are summarized as follows:

| Processing Action | Exit 1 | Exit 2 | Exit 3 | Exit 4 | Exit 5 |
|---|---|---|---|---|---|
| Substitute a record | retaddr | | retaddr | | retaddr |
| Insert a record | retaddr+2 | retaddr+1 | retaddr+2 | retaddr+1 | |
| Delete a record | retaddr+1 | | retaddr+1 | | retaddr+1 |
| Terminate a file | retaddr+3 | | retaddr+3 | | |
| Normal processing | retaddr | retaddr | retaddr | retaddr | retaddr |

When the processing action requested is file termination, the current record in array a is not included in Sort/Merge processing.

Special factors to consider when using Exit 5 owncode routines are:

If SMRTN is called from an Exit 5 owncode routine, the number of parameters on the SMRTN call determines the processing action requested.

If an Exit 5 owncode routine is used for record substitution, the record contained in both area a and array b can be substituted with both new address and new field length specifications.

If an Exit 5 owncode routine is used to delete a record, the record contained in array b is deleted.

### EXAMPLE

```
CALL SMOWN(3,SUB3)
      .
      .
      .
SUBROUTINE SUB3(L,N)
DIMENSION L(20),M(20)
      .
      .
CALL SMRTN(2,M,N)
```

## SMABT

This call terminates a sequence of Sort/Merge interface calls without initiating execution of Sort/Merge. The state of the interface is the same as if no calls had been made.

The format for the SMABT subroutine is as follows:

```
CALL SMABT
```

This call has no parameters.

## SAMPLE PROGRAM

The following program merges STDFILE, a master file containing student records, and NEWFILE, a file containing new student records.

New records are added to NEWGRAD, a file containing student grades for this semester.

The program computes new grade point averages, and the total units and grade points to date for each student.

A new master file is created, MFILE. Records of students on probation are deleted from MFILE and placed on REPORT.

A main program and three subprograms are used. Two of the subprograms are owncode routines.

```
      PROGRAM GPA
      IMPLICIT INTEGER(A-Z)
      DIMENSION STUDNAM(2)
      COMMON/OWN/RECRL,NRECRL,REPRTRL
      EXTERNAL UPDATE
      EXTERNAL LIST
      REAL GP1,UNITS1,GPA1,UNITS2,GPA2,TOTGP,TOTUNIT,GP
C     ***
C     ***
C     ***  OPEN THE FOLLOWING FILES TO BE USED IN PROGRAM GPA-
C     ***
C     ***  INPUT CONTAINS THE STUDENT NUMBER, GRADE POINT AVERAGE AND TOTAL
C     ***  UNITS OF NEW STUDENTS.
C     ***
C     ***  REPORT CONTAINS THE FINAL OUTPUT FILE FOR THOSE STUDETNS ON PROBATION.
C     ***
C     ***  STDFILE CONTAINS STUDENT RECORDS OF THOSE ENROLLED AT THE SCHOOL PRIOR
C     ***  TO THE PRESENT SEMESTER.
C     ***
C     ***  NEWFILE CONTAINS THE STUDENT RECORDS OF NEW STUDENTS.
C     ***
C     ***  NEWGRAD CONTAINS THE STUDENT NUMBER, GRADE POINT AVERAGE AND UNITS TAKEN
C     ***  THIS SEMESTER FOR STUDENTS WITH RECORDS IN STDFILE.
C     ***
C     ***  TEMP1 IS A FILE THAT CONTAINS SORTED RECORDS FROM BOTH STDFILE AND
C     ***  NEWFILE.
C     ***
C     ***  TEMP2 IS A SORTED FILE COMPOSED OF THE FILES INPUT AND NEWGRAD.
C     ***
C     ***  TEMP3 IS A FILE CONTAINING THE RECORDS OF THOSE STUDENTS PLACED ON
C     ***  PROBATION THIS SEMESTER.
C     ***
C     ***  MFILE CONTAINS THE OUTPUT FILE FOR THOSE STUDENTS WITH NORMAL STANDING.
C     **
C     **
      OPEN(5,FILE=↑INPUT↑)
      OPEN(6,FILE=↑REPORT↑)
      OPEN(7,FILE=↑STDFILE↑)
      OPEN(8,FILE=↑NEWGRAD↑)
      OPEN(9,FILE=↑NEWFILE↑)
      OPEN(10,FILE=↑TEMP1↑)
      OPEN(11,FILE=↑TEMP2↑)
      OPEN(12,FILE=↑TEMP3↑)
      OPEN(14,FILE=↑MFILE↑)
C     **
      REWIND 7
      REWIND 8
      REWIND 9
C     **
C     **
      READ(5,1)RECRL,NRECRL,REPRTRL,CSIZE
C     **
C     **
C     SORT STDFILE AND NEWFILE ACCORDING TO STUDENT NUMBERS, PLACE
C     RESULTS IN TEMP1.
C     ***
C     ***
      CALL SMSORT(NRECRL)
      CALL SMFILE(↑SORT↑,↑CODED↑,9,↑REWIND↑)
      CALL SMFILE(↑SORT↑,↑CODED↑,7,↑REWIND↑)
```

```
      CALL SMFILE(↑OUTPUT↑,↑CODED↑,10,↑REWIND↑)
      CALL SMKEY(21,1,6,0,↑LOGICAL↑)
      CALL SMEND
C  **
C  **
C  SORT NEWGRAD ACCORDING TO STUDENT NUMBER.  PLACE IN TEMP2.
C  **
C  **
      CALL SMSORT(RECRL)
      CALL SMFILE(↑SORT↑,↑CODED↑,8,↑REWIND↑)
      CALL SMFILE(↑OUTPUT↑,↑CODED↑,11,↑REWIND↑)
      CALL SMKEY(1,1,6,0,↑LOGICAL↑)
      CALL SMOWN(2,UPDATE)
      CALL SMEND
C  **
C  **
C  COMPUTE TOTGP(TOTAL GRADE POINTS),TOTUNIT(TOTAL UNITS TAKEN TO DATE)
C  AND GP(NEW GRADE POINT AVERAGE).
C  WRITE NEW RECORDS ON TEMP3.
C  **
C  **
      DO 100 I=1,CSIZE
         READ(10,5,END=150)STUDNAM(1),STUDNAM(2),STUDNO1,GPA1,UNITS1
         READ(11,6,END=150)STUDNO2,GPA2,UNITS2
         IF(STUDNO1 .NE. STUDNO2)THEN
            CALL ERROR(1,STUDNO1,STUDNO2)
         ELSE
            GP1=UNITS1*GPA1
            GP2=UNITS2*GPA2
            TOTGP=GP1+GP2
            TOTUNIT=UNITS1+UNITS2
            GP=TOTGP/TOTUNIT
         ENDIF
            WRITE(12,10)STUDNAM(1),STUDNAM(2),STUDNO2,GP,TOTUNIT,TOTGP
  100 CONTINUE
  150 REWIND 12
C  **
C  **
C  WRITE HEADERS FOR REPORT(TAPE6) AND MFILE(TAPE14).
C  **
C  **
      WRITE(6,11)
      WRITE(14,12)
      WRITE(6,13)
      WRITE(14,13)
C  **
C  **
C  SORT TEMP3.
C  BEFORE THE FINAL OUTPUT FILE IS WRITTEN, SEND THE RECORDS TO OWNCODE
C  ROUTINE (SUBROUTINE LIST).  SUBROUTINE LIST DETERMINES WHETHER THE
C  VALUE OF THE FIELD CONTAINING THE GRADE POINT AVERAGE IS LESS THAN 2.0.
C  IF THE GRADE POINT AVERAGE IS BELOW 2.0, SUBROUTINE LIST WRITES
C  THE RECORD TO REPORT, AND DELETES THE RECORD FROM THE FINAL OUTPUT FILE,
C  MFILE.
C  FIRST SORT KEY FOR MFILE-GRADE POINT AVERAGE(GP) IN DESCENDING ORDER.
C  SECOND SORT KEY FOR MFILE-STUDENT NAME(STUDNAM) IN ALPHABETICAL ORDER.
C  THIRD SORT KEY FOR MFILE-STUDENT NUMBER(STUDNO)
C  **
C  **
      CALL SMSORT(REPRTRL)
      CALL SMFILE(↑SORT↑,↑CODED↑,12,↑REWIND↑)
```

```
      CALL SMFILE(↑OUTPUT↑,↑CODED↑,14,↑REWIND↑)
      CALL SMKEY(43,1,4,0,↑LOGICAL↑,↑D↑)
      CALL SMKEY(1,1,23,0,↑LOGICAL↑)
      CALL SMKEY(25,1,6,0,↑LOGICAL↑)
      CALL SMOWN(3,LIST)
      CALL SMEND



    1 FORMAT(I2,I2,I2,I2)
    5 FORMAT(2A10,I6,2X,F3.1,2X,F5.1)
    6 FORMAT(I6,2X,F3.1,2X,F5.1)
   10 FORMAT(2A10,4X,I6,12X,F4.2,15X,F5.1,15X,F5.1)
   11 FORMAT(↑STUDENTS ON PROBATION:↑,/)
   12 FORMAT(↑STUDENTS WITH NORMAL STANDING:↑,/)
   13 FORMAT(8X,↑NAME↑,10X,↑STUDENT NO↑,2X,↑GRADE POINT AVERAGE↑,6X,
     *       ↑TOTAL UNITS↑,5X,↑TOTAL GRADE POINTS↑,/)
      STOP
      END


      SUBROUTINE ERROR(ITERATE,NO1,NO2)
      CALL REMARK(↑STUDENT NUMBERS DO NOT MATCH↑)
      CALL DISPLA(↑FIRST STUDENT NUMBER IS↑,NO1)
      CALL DISPLA(↑SECOND STUDENT NUMBER IS↑,NO2)
      CALL REMARK(↑JOB TERMINATED↑)
      CALL DISPLA(↑AT ITERATION↑,ITERATE)
      CALL EXIT
      END


      SUBROUTINE UPDATE(ARRAY,ARL)
      IMPLICIT INTEGER(A-Z)
      COMMON/OWN/RECRL,NRECRL,REPRTRL
      DIMENSION RECORD(2),ARRAY(2)
      READ(5,100,END=18)(RECORD(K),K=1,2)
      CALL SMRTN(1,RECORD,RECRL)
   18 CALL SMRTN(0)
      STOP 77
  100 FORMAT(2A10)
      END


      SUBROUTINE LIST(N,A)
      IMPLICIT INTEGER(A-Z)
      COMMON/OWN/RECRL,NRECRL,REPRTRL
      REAL TEMP
      DIMENSION N(9),NEWREC(9)
      DECODE(46,1,N)TEMP
      IF(TEMP .LT.2.0)THEN
      WRITE(6,2)(N(K),K=1,9)
         CALL SMRTN(1)
      ENDIF
      ENCODE(90,2,NEWREC)(N(K),K=1,9)
   20 CALL SMRTN(0,NEWREC,REPRTRL)
      STOP 77
    1 FORMAT(42X,F4.2)
    2 FORMAT(9A10)
      END
7/8/9
```

The following files were used as input to the program:

**STDFILE**

```
DAVIS          ,SUSAN     103392   2.0   102.5
ZABROSKI       ,GERRI     294531   4.0    88.0
CASSET         ,PETER     271217   2.8   115.5
BORCHARD       ,JOAN      807337   2.7    70.0
ANDERSON       ,GEORGE    408227   2.5    95.5
DE LA CRUZ     ,LILY      666358   3.3    50.0
DAMIEN         ,SCOTT     805483   3.1   101.5
MICHAELS       ,TERESA    669240   4.0    45.0
MONTE          ,MARK      578910   2.1    87.0
ROBBINS        ,STEVE     225121   1.9    62.0
MARTIN         ,JUDY      145010   2.5    57.5
PHILLIP        ,RALPH     741020   2.4   120.0
STEVENS        ,MARY      593478   3.3    78.5
MASTER         ,RON       943621   2.1   124.5
YAFFEE         ,JOSEPH    606080   3.7    74.0
FILICE         ,DON       694321   3.9    24.5
PHILPS         ,SUE       313559   2.8   115.0
WILSON         ,JOHN      215996   2.9    16.0
```

**INPUT**

```
20409023

857932  3.5    15.0
239410  1.9     6.0
973249  2.4    10.0
423911  3.0    12.5
143976  4.0    13.0
7/8/9
6/7/8/9
```

**NEWGRAD**

```
313559  1.9     7.5
215996  2.9    13.0
807337  1.5     8.0
271217  2.9    15.0
294531  3.9     8.5
103392  1.0    10.0
606080  3.2     6.0
593478  3.5    15.0
694321  3.8    12.0
943621  1.7     9.0
145010  3.0    10.0
225121  1.5    12.0
578910  2.0     9.0
741020  2.5    15.0
669240  3.8    15.0
805483  2.9    11.0
666358  2.5    17.0
408227  3.8    15.0
```

NEWFILE

```
BIANCI      ,NANCY    423911   0.0   000.0
ROBBINS     ,SAM      973249   0.0   000.0
WHELDON     ,GERT     143976   0.0   000.0
NEWMAN      ,AL       239410   0.0   000.0
TORRES      ,MANUAL   857932   0.0   000.0
```

The program was run using the following control statements:

```
FTN5.

LGO.
```

The following reports were produced:

MFILE

STUDENTS WITH NORMAL STANDING:

| NAME | | STUDENT NO | GRADE POINT AVERAGE | TOTAL UNITS | TOTAL GRADE POINTS |
|---|---|---|---|---|---|
| WHELDON | ,GERT | 143976 | 4.00 | 13.0 | 52.0 |
| ZABROSKI | ,GERRI | 294531 | 3.99 | 96.5 | 385.0 |
| MICHAELS | ,TERESA | 669240 | 3.95 | 60.0 | 237.0 |
| FILICE | ,DON | 694321 | 3.85 | 36.5 | 140.6 |
| YAFFEE | ,JOSEPH | 606080 | 3.66 | 80.0 | 292.8 |
| TORRES | ,MANUAL | 857932 | 3.47 | 15.0 | 52.0 |
| STEVENS | ,MARY | 593478 | 3.33 | 93.5 | 311.1 |
| DE LA CRUZ | ,LILY | 666358 | 3.09 | 67.0 | 207.0 |
| DAMIEN | ,SCOTT | 805483 | 3.07 | 112.5 | 345.7 |
| BIANCI | ,NANCY | 423911 | 2.96 | 12.5 | 37.0 |
| WILSON | ,JOHN | 215996 | 2.88 | 29.0 | 83.4 |
| CASSET | ,PETER | 271217 | 2.81 | 130.5 | 366.4 |
| PHILPS | ,SUE | 313559 | 2.74 | 122.5 | 336.0 |
| ANDERSON | ,GEORGE | 408227 | 2.68 | 110.5 | 295.8 |
| BORCHARD | ,JOAN | 807337 | 2.58 | 78.0 | 201.0 |
| MARTIN | ,JUDY | 145010 | 2.57 | 67.5 | 173.8 |
| PHILLIP | ,RALPH | 741020 | 2.41 | 135.0 | 325.0 |
| ROBBINS | ,SAM | 973249 | 2.40 | 10.0 | 24.0 |
| MONTE | ,MARK | 578910 | 2.09 | 96.0 | 200.7 |
| MASTER | ,RON | 943621 | 2.07 | 133.5 | 276.5 |

REPORT

STUDENTS ON PROBATION:

| NAME | | STUDENT NO | GRADE POINT AVERAGE | TOTAL UNITS | TOTAL GRADE POINTS |
|---|---|---|---|---|---|
| DAVIS | ,SUSAN | 103392 | 1.91 | 112.5 | 215.0 |
| ROBBINS | ,STEVE | 225121 | 1.84 | 74.0 | 135.8 |
| NEWMAN | ,AL | 239410 | 1.83 | 6.0 | 11.0 |

# CHARACTER SETS A

CONTROL DATA operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation was specified when the operating system was installed.

Depending on another installation option, the system assumes an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use). Under NOS/BE 1, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card.

Under NOS 1, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card, as described above for a 7/8/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1, and 029 mode can be specified by a card with 5/7/9 multipunched in column 1 and a 9 punched in column 2.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table (table A-1) are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

## STANDARD COLLATING SEQUENCES

If the installation character set is the CDC character set, the collating sequence default is COBOL6. If the installation character set is ASCII, the collating sequence default is ASCII6 (as shown in table A-2).

## COLLATION OF ARBITRARY CHARACTERS

Several graphics are not common for all codes. Where these differences in graphics occur, arbitrary assignment of collation positions and of translations between codes must be made. For example, display code data that is collated in the ASCII6 collating sequence requires assignment of specific graphics. One of these graphics is the identity character ≡ (60) in display code that is interpreted as the number character (#) in ASCII6. The identity is collated in position 03, according to the ASCII6 collation column in table A-2.

## TABLE A-1. STANDARD CHARACTER SETS

| Display Code (octal) | CDC | | | ASCII | | |
|---|---|---|---|---|---|---|
| | Graphic | Hollerith Punch (026) | External BCD Code | Graphic Subset | Punch (029) | Code (octal) |
| 00† | : (colon)†† | 8-2 | 00 | : (colon)†† | 8-2 | 072 |
| 01 | A | 12-1 | 61 | A | 12-1 | 101 |
| 02 | B | 12-2 | 62 | B | 12-2 | 102 |
| 03 | C | 12-3 | 63 | C | 12-3 | 103 |
| 04 | D | 12-4 | 64 | D | 12-4 | 104 |
| 05 | E | 12-5 | 65 | E | 12-5 | 105 |
| 06 | F | 12-6 | 66 | F | 12-6 | 106 |
| 07 | G | 12-7 | 67 | G | 12-7 | 107 |
| 10 | H | 12-8 | 70 | H | 12-8 | 110 |
| 11 | I | 12-9 | 71 | I | 12-9 | 111 |
| 12 | J | 11-1 | 41 | J | 11-1 | 112 |
| 13 | K | 11-2 | 42 | K | 11-2 | 113 |
| 14 | L | 11-3 | 43 | L | 11-3 | 114 |
| 15 | M | 11-4 | 44 | M | 11-4 | 115 |
| 16 | N | 11-5 | 45 | N | 11-5 | 116 |
| 17 | O | 11-6 | 46 | O | 11-6 | 117 |
| 20 | P | 11-7 | 47 | P | 11-7 | 120 |
| 21 | Q | 11-8 | 50 | Q | 11-8 | 121 |
| 22 | R | 11-9 | 51 | R | 11-9 | 122 |
| 23 | S | 0-2 | 22 | S | 0-2 | 123 |
| 24 | T | 0-3 | 23 | T | 0-3 | 124 |
| 25 | U | 0-4 | 24 | U | 0-4 | 125 |
| 26 | V | 0-5 | 25 | V | 0-5 | 126 |
| 27 | W | 0-6 | 26 | W | 0-6 | 127 |
| 30 | X | 0-7 | 27 | X | 0-7 | 130 |
| 31 | Y | 0-8 | 30 | Y | 0-8 | 131 |
| 32 | Z | 0-9 | 31 | Z | 0-9 | 132 |
| 33 | 0 | 0 | 12 | 0 | 0 | 060 |
| 34 | 1 | 1 | 01 | 1 | 1 | 061 |
| 35 | 2 | 2 | 02 | 2 | 2 | 062 |
| 36 | 3 | 3 | 03 | 3 | 3 | 063 |
| 37 | 4 | 4 | 04 | 4 | 4 | 064 |
| 40 | 5 | 5 | 05 | 5 | 5 | 065 |
| 41 | 6 | 6 | 06 | 6 | 6 | 066 |
| 42 | 7 | 7 | 07 | 7 | 7 | 067 |
| 43 | 8 | 8 | 10 | 8 | 8 | 070 |
| 44 | 9 | 9 | 11 | 9 | 9 | 071 |
| 45 | + | 12 | 60 | + | 12-8-6 | 053 |
| 46 | - | 11 | 40 | - | 11 | 055 |
| 47 | * | 11-8-4 | 54 | * | 11-8-4 | 052 |
| 50 | / | 0-1 | 21 | / | 0-1 | 057 |
| 51 | ( | 0-8-4 | 34 | ( | 12-8-5 | 050 |
| 52 | ) | 12-8-4 | 74 | ) | 11-8-5 | 051 |
| 53 | $ | 11-8-3 | 53 | $ | 11-8-3 | 044 |
| 54 | = | 8-3 | 13 | = | 8-6 | 075 |
| 55 | blank | no punch | 20 | blank | no punch | 040 |
| 56 | , (comma) | 0-8-3 | 33 | , (comma) | 0-8-3 | 054 |
| 57 | . (period) | 12-8-3 | 73 | . (period) | 12-8-3 | 056 |
| 60 | ≡ | 0-8-6 | 36 | # | 8-3 | 043 |
| 61 | [ | 8-7 | 17 | [ | 12-8-2 | 133 |
| 62 | ] | 0-8-2 | 32 | ] | 11-8-2 | 135 |
| 63 | %†† | 8-6 | 16 | %†† | 0-8-4 | 045 |
| 64 | ≠ | 8-4 | 14 | " (quote) | 8-7 | 042 |
| 65 | ← | 0-8-5 | 35 | _ (underline) | 0-8-5 | 137 |
| 66 | ∨ | 11-0 | 52 | ! | 12-8-7 | 041 |
| 67 | ∧ | 0-8-7 | 37 | & | 12 | 046 |
| 70 | ↑ | 11-8-5 | 55 | ' (apostrophe) | 8-5 | 047 |
| 71 | ↓ | 11-8-6 | 56 | ? | 0-8-7 | 077 |
| 72 | < | 12-0 | 72 | < | 12-8-4 | 074 |
| 73 | > | 11-8-7 | 57 | > | 0-8-6 | 076 |
| 74 | ≤ | 8-5 | 15 | @ | 8-4 | 100 |
| 75 | ≥ | 12-8-5 | 75 | \ | 0-8-2 | 134 |
| 76 | ¬ | 12-8-6 | 76 | ^ (circumflex) | 11-8-7 | 136 |
| 77 | ; (semicolon) | 12-8-7 | 77 | ; (semicolon) | 11-8-6 | 073 |

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end-of-record mark rather than two colons.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank ($55_8$).

## TABLE A-2. 6-BIT CHARACTER CODE COLLATING SEQUENCES

| COBOL6† | | DISPLAY† | | INTBCD | | ASCII6†† | |
|---|---|---|---|---|---|---|---|
| Graphics | Display Code | Graphics | Display Code | Graphics | CDC INTBCD | Graphics | Sequence |
| blank | 55 | :† | 00† | 0 | 00 | blank | 00 |
| ≤ | 74† | A | 01 | 1 | 01 | ! | 01 |
| %† | 63 | B | 02 | 2 | 02 | " | 02 |
| [ | 61 | C | 03 | 3 | 03 | # | 03 |
| → | 65 | D | 04 | 4 | 04 | $ | 04 |
| ≡ | 60 | E | 05 | 5 | 05 | %†† | 05 |
| ∧ | 67 | F | 06 | 6 | 06 | & | 06 |
| ↑ | 70 | G | 07 | 7 | 07 | ' | 07 |
| | 71 | H | 10 | 8 | 10 | ( | 10 |
| > | 73 | I | 11 | 9 | 11 | ) | 11 |
| ≥ | 75 | J | 12 | : | 12 | * | 12 |
| ¬ | 76 | K | 13 | = | 13 | + | 13 |
| . | 57 | L | 14 | ≠ | 14 | , | 14 |
| ) | 52 | M | 15 | ≤ | 15 | - | 15 |
| ; | 77 | N | 16 | % | 16 | . | 16 |
| + | 45 | O | 17 | [ | 17 | / | 17 |
| $ | 53 | P | 20 | + | 20 | 0 | 20 |
| * | 47 | Q | 21 | A | 21 | 1 | 21 |
| - | 46 | R | 22 | B | 22 | 2 | 22 |
| / | 50 | S | 23 | C | 23 | 3 | 23 |
| , | 56 | T | 24 | D | 24 | 4 | 24 |
| ( | 51 | U | 25 | E | 25 | 5 | 25 |
| = | 54 | V | 26 | F | 26 | 6 | 26 |
| ≠ | 64 | W | 27 | G | 27 | 7 | 27 |
| < | 72 | X | 30 | H | 30 | 8 | 30 |
| A | 01 | Y | 31 | I | 31 | 9 | 31 |
| B | 02 | Z | 32 | < | 32 | : | 32 |
| C | 03 | 0 | 33 | . | 33 | ; | 33 |
| D | 04 | 1 | 34 | ) | 34 | < | 34 |
| E | 05 | 2 | 35 | ≥ | 35 | = | 35 |
| F | 06 | 3 | 36 | ¬ | 36 | > | 36 |
| G | 07 | 4 | 37 | ; | 37 | ? | 37 |

## TABLE A-2. 6-BIT CHARACTER CODE COLLATING SEQUENCES (Contd)

| COBOL6 † | | DISPLAY † | | INTBCD | | ASCII6 †† | |
|---|---|---|---|---|---|---|---|
| Graphics | Display Code | Graphics | Display Code | Graphics | CDC INTBCD Code | Graphics | Sequence |
| H | 10 | 5 | 40 | - | 40 | @ | 40 |
| I | 11 | 6 | 41 | J | 41 | A | 41 |
| ∨ | 66 | 7 | 42 | K | 42 | B | 42 |
| J | 12 | 8 | 43 | L | 43 | C | 43 |
| K | 13 | 9 | 44 | M | 44 | D | 44 |
| L | 14 | + | 45 | N | 45 | E | 45 |
| M | 15 | - | 46 | O | 46 | F | 46 |
| N | 16 | * | 47 | P | 47 | G | 47 |
| O | 17 | / | 50 | Q | 50 | H | 50 |
| P | 20 | ( | 51 | R | 51 | I | 51 |
| Q | 21 | ) | 52 | ∨ | 52 | J | 52 |
| R | 22 | $ | 53 | $ | 53 | K | 53 |
| ] | 62 | = | 54 | * | 54 | L | 54 |
| S | 23 | blank | 55 | ↑ | 55 | M | 55 |
| T | 24 | , | 56 | ↓ | 56 | N | 56 |
| U | 25 | . | 57 | > | 57 | O | 57 |
| V | 26 | ≡ | 60 | blank | 60 | P | 60 |
| W | 27 | [ | 61 | / | 61 | Q | 61 |
| X | 30 | ] | 62 | S | 62 | R | 62 |
| Y | 31 | %† | 63† | T | 63 | S | 63 |
| Z | 32 | ≠ | 64 | U | 64 | T | 64 |
| : | 00 † | → | 65 | V | 65 | U | 65 |
| 0 | 33 | ∨ | 66 | W | 66 | V | 66 |
| 1 | 34 | ∧ | 67 | X | 67 | W | 67 |
| 2 | 35 | ↑ | 70 | Y | 70 | X | 70 |
| 3 | 36 | ↓ | 71 | Z | 71 | Y | 71 |
| 4 | 37 | < | 72 | ] | 72 | Z | 72 |
| 5 | 40 | > | 73 | , | 73 | [ | 73 |
| 6 | 41 | ≤ | 74 | ( | 74 | \ | 74 |
| 7 | 42 | ≥ | 75 | → | 75 | ] | 75 |
| 8 | 43 | ¬ | 76 | ≡ | 76 | ^ | 76 |
| 9 | 44 | ; | 77 | ∧ | 77 | _ | 77 |

†Under the CDC 63-character set, there is no percent graphic; the colon is display code 63. Display Code 00 is not used.

††Under the ASCII 63-character set, there is no percent graphic; the colon collates in position 05, not position 32.

# SORT/MERGE DIAGNOSTICS B

## DIRECTIVE DIAGNOSTICS

During directive processing, Sort/Merge returns the diagnostic messages listed in table B-1. The dayfile message for Sort/Merge directive errors will be one of the following:

n NON-FATAL DIRECTIVE ERROR(S) or n FATAL DIRECTIVE ERROR(S)

A diagnostic message is written to the output file for each error. The messages appear in three formats:

This format is used for error conditions detected during a syntax check.

**type**CARD NBR num message AT param subparam

| | | |
|---|---|---|
| type | F | Fatal error causing abnormal termination of job |
| | N | Nonfatal error; job processing continues |
| num | Number of directive or statement in error | |
| message | One of the diagnostics listed | |
| param | Parameter that contains error | |
| subparam | Specific character causing the error | |

Because Sort/Merge continues syntax checking after an error is detected and a diagnostic is returned to the user, subsequent diagnostics for a given directive may not correctly apply to the intended syntax. Subsequent syntax (which may or may not be correct) could have been made illegal by a preceding error.

This diagnostic format is listed after the END directive is printed. It indicates errors encountered during cross checks made by Sort/Merge on internal tables generated from directive specifications.

**type**statement,message

| | | |
|---|---|---|
| type | F | Fatal error causing an abnormal termination of job |
| | N | Nonfatal error resulting in continued processing of the job |
| statement | Directive in error | |
| message | One of the diagnostic messages listed | |

This format is used for all diagnostics returned to the dayfile:

SORTMRG message

TABLE B-1. DIRECTIVE DIAGNOSTICS

| Message | Significance | Action |
|---|---|---|
| BAD CALL TO SORTL | System or user error; SORTL is no longer supported. | Notify a systems analyst. |
| BOUNDARY OVERLAP | Character in sort key overlaps a word boundary. | Use LOGICAL type key, or reformat records and FIELD directive to align characters on character boundaries. |
| BYTE AND BIT MISSING | Byte and bit specifications have been omitted from FIELD directive. | Respecify FIELD directive with byte and bit parameters included. |
| BYTESIZE MISSING | Byte parameter was specified on BYTESIZE directive, but no bit specification followed the period. | Delete the period. |
| CARD AFTER END CARD | Card other than a blank card or a comment card detected after END directive. | Delete illegal card. |
| CODE SIZE CONFLICT | Sort key length for INTBCD or DISPLAY type key is not an integer multiple of character code size. | Specify length of sort key as integral multiple of 6 bits. |
| DATA OVERFLOW | Too many characters specified in user's collating sequence. | Reduce number of characters to 64 or less. |
| DEFAULT REPLACES | Default value defined by installation has been used for LCMSB parameter in SORT directive. | Specify LCMSB parameter on SORT directive. |
| DUPLICATE BYTESZ CARD | More than one BYTESIZE directive has been specified in a single run. | Remove duplicate BYTESIZE directive. |
| DUPLICATE COLLATION | One character appears more than once in user's collating sequence. This might occur when the SEQUENCE directive is continued on a second card and trailing blanks have been specified on the first card. | Delete duplicate occurrence of character in user collating sequence. |
| EMPTY MERGE FILE...file-name | No records contained in merge file. File-name is name of merge file specified in FILE directive. | None. |

TABLE B-1. DIRECTIVE DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| EOF, NO LEGAL CARD | No SORT or MERGE directive has been specified. | Include SORT or MERGE directive in sort or merge run. |
| EXIT NUMBER LT 1 | Improper exit number specified on OWNCODE directive. | Specify exit number as a value from 1 to 6. |
| EXIT NUMBER GT 6 | Improper exit number specified on OWNCODE directive. | Specify exit number as a value from 1 to 6. |
| EX1, EX2 OWN ILLEGAL | Exit 1 or exit 2 owncode specified for a merge-only run. | Change to sort run. |
| EX-1 OWNCODE ILLEGAL | Exit 1 owncode specified for a merge-only run. | Change to sort run. |
| EXTRA SEPARATOR | Too many separators specified in directive. | Delete illegal separators. |
| FATAL ERRORS LOADING SORT | Sort/Merge system was installed incorrectly. | Notify a systems analyst. |
| FIELD SIZE $\neq$ 60 BITS | Length of integer or floating point sort key is not 60 bits. | Respecify FIELD directive with length of sort key equal to 60 bits. |
| ILLEGAL ATTRIBUTE | Parameter specification is illegal or partially missing. | Respecify parameter correctly. |
| ILLEGAL BIT = 0 | Bit parameter illegally specified as 0 in FIELD directive. | Change bit parameter to value other than zero. |
| ILLEGAL BYTE = 0 | Byte parameter illegally specified as 0 in FIELD directive. | Change byte parameter to value other than zero. |
| ILLEGAL CARD TYPE | Unrecognizable card was submitted. | Correct directive format. |
| ILLEGAL CONTINUATION | Card following an incomplete directive did not have a comma in column 1. | Either complete the directive on the previous line or use a continuation line with a comma in column 1. |
| ILLEGAL DELIMITER | Separator other than the ones legally designated has been used. | Replace illegal separator with legal one. |
| ILLEGAL EQUALS | Equals sign specified in wrong position. | Correct directive format. |
| ILLEGAL FILE TYPE | Illegal file type specified on FILE directive. | Correct type parameter specification. |

TABLE B-1. DIRECTIVE DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| ILLEGAL KEYWORD | Entry parameter specified with illegal exit number on OWNCODE directive; or MRL, VAR, or LCMSB specification omitted or illegal. | Specify legal exit number for entry parameter; or specify MRL, VAR, or LCMSB parameter correctly. |
| ILLEGAL MERGE ORDER | Merge order specified was not an integer between 2 and 64 inclusive. | Specify merge order value as an integer between 2 and 64 on the SORTMRG control statement or through the ORDER option of the OPTIONS directive. |
| ILLEGAL NBYTE = NBIT = 0 | Zero cannot be specified for both nbyte and nbit parameters in the FIELD directive. | Specify value other than zero for nbyte and/or nbit. |
| ILLEGAL OPTION | Illegal option specified. | Correct option specification. |
| ILLEGAL PERIOD | Period cannot be used to separate start and length parameters in the FIELD directive. | Respecify FIELD directive with other legal separator between start and length parameters. |
| ILLEGAL PROG NAME | First character specified in entry point name was numeric, or name exceeded 7 characters. | Correct entry point name in OWNCODE directive. |
| ILLEGAL RIGHT PAREN | Right parenthesis specified incorrectly; sort key definition incomplete. | Correct directive format. |
| ILLEGAL TERMINATION | Incomplete specification of required parameter. | Correct parameter specification. |
| INPUT MRL NOT GIVEN | Maximum record length not specified as input to Sort/Merge. | Specify MRL or FL parameter on FILE control statement, or MRL parameter on OWNCODE directive. |
| INSUFFICIENT CM FOR SPECIFIED MERGE ORDER xxxxx MORE REQUIRED | Insufficient core was provided for Sort/Merge to merge intermediate strings at the order specified. xxxxx is the number of extra words (in octal) required. | Increase field length for control statement sorts. |
| INSUFFICIENT SCM | Not enough small core memory was allocated. | Increase field length for control statement sorts. |

TABLE B-1. DIRECTIVE DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| INVALID DIRECTIVE | SORT, SORTB, SORTP, or MERGE must be the first directive encountered in a SORTMRG call in 6C format. | Insert SORT, SORTB, SORTP, or MERGE directive in correct position. |
| INVALID SEQUENCE | Collating sequence cannot be specified for a sort key not character coded. | Delete colseq parameter from KEY directive or define field as DISPLAY. |
| LFN = filename, BFS = 0, FWB NZ, FWB IGNORED | The FIT for the file named was provided with a value for FWB, but no buffer size was given. The FWB is ignored. | Either clear FWB to zero or set BFS to size of allocated space. |
| MEMORY NEEDS EXCEED MAXIMUM. PROBABLY CAUSED BY USING OWN-CODE WITH RESTRICTED FIELD LENGTH | Extra 10000B words needed for OWNCODE binaries not included in maximum field length (CM parameter or machine capacity). | Remove CM from job card or reduce the RFL by 10000B before using Sort/Merge. |
| MERGE, MERGE ASSUMED | A merge with INPUT= was specified on the FILE directive. MERGE= is assumed for the merge. | Change INPUT= to MERGE=. Or if merge was assumed due to the omission of the SORT card, supply the SORT directive. |
| MERGE ORDER xx USED | Insufficient core was provided to attain the maximum merge order specified; order xx was used instead. | Increase field length for control statement sorts. |
| MINIMUM REPLACES | Value specified in LCMSB parameter was less than installation defined minimum; installation default value was used instead. | Increase value specified for LCMSB parameter. |
| MISSING ATTRIBUTES | Keyword specified with no following parameters. | Correct directive format. |
| MISSING EQUAL SIGN | The equals sign, required after the file type specification in the FILE directive, was omitted. | Insert equals sign. |
| MISSING FILE NAME | File name was omitted from FILE directive. | Include file name on the FILE directive. |
| MISSING FILE TYPE | File type was omitted from FILE directive. | Include file type on the FILE directive. |
| MISSING KEY NAME | Sort key name was omitted from FIELD or KEY directive or did not follow required separator. | Specify key name correctly on FIELD or KEY directive. |

TABLE B-1. DIRECTIVE DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| MISSING LEFT PAREN | Left parenthesis was omitted from a directive. | Respecify directive with left parenthesis included. |
| MISSING OPTION | No option specified on OPTIONS directive. | Delete OPTIONS directive or specify an option on it. |
| MISSING RIGHT PAREN | Right parenthesis was omitted from a directive. | Respecify directive with right parenthesis included. |
| MISSING SORT CARD | The first directive is not a SORT or MERGE directive. MERGE is assumed. | Supply a SORT or MERGE directive. |
| MORE THAN 1 OUT-FILE | More than one output file was specified on the FILE directive. | Delete all but one output file. |
| MRL UNDEFINED | MRL parameter for an exit 1 was omitted from the OWNCODE directive. | Respecify OWNCODE directive with MRL parameter included. |
| MULTIPLE SORT CARD | More than one SORT directive was specified in a single run. | Delete all but one SORT directive. |
| MULTIPLY DEFINED | If fatal error, a parameter was specified in different ways within one run. If nonfatal error, a parameter was specified in the same way more than once within one run. | Delete all but one occurrence of parameter. |
| NBYTE AND NBIT MISSING | Nbyte and nbit specifications were omitted from the FIELD directive. | Respecify FIELD directive with nbyte and/or nbit included. |
| NEED MRL FOR ... filename | MRL parameter was not specified on FILE control statement. File name is name specified on FILE directive. | Specify MRL parameter on FILE control statement. |
| NO LEGAL CARD | End-of-file condition was detected before a legal control statement was processed. | Insert legal control statement. |
| NON-NUMERIC | Alphabetic character was specified in numeric field. | Delete alphabetic character from field or redefine field. |
| ONLY 1 MERGE FILE | Merge-only processing requires more than one presorted input file. | Replace the sort operation with a copy operation. |
| OVER 100 KEYS | More than 100 sort keys were specified. | Reduce number of sort keys to 100 or less. |

TABLE B-1. DIRECTIVE DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| OVER 100 SEQUENCES | More than 100 collating sequences were specified. | Reduce number of collating sequences to 100 or less. |
| OWNCODE NOT FOUND, entry | Entry name specified in OWNCODE directive was not found in the input file. Entry is the entry point name of an exit owncode. | Correct entry name or include owncode routine in input file. |
| RESERVED COL-SEQ | Standard collating sequence name was specified as a user's collating sequence. | Redefine user collating sequence name. |
| SORT/MERGE ABORT | Sort/Merge program aborted because of preceding fatal error. | Fix preceding fatal error. |
| UNDEFINED FIELD | Sort key name was not specified on FIELD directive. | Respecify FIELD directive with key name included. |
| UNDEFINED IN-FILE | An input file was not specified either on a FILE directive or with exit 1 on an OWNCODE directive. | Specify input file on FILE or OWNCODE directive. |
| UNDEFINED I/O FILE | No files or owncode exits were specified for input or output. | Specify input or output file on FILE or OWNCODE directive. |
| UNDEFINED KEY(S) | No sort key was specified. | Specify sort key on FIELD directive. |
| UNDEFINED OUT-FILE | An output file was not specified on a FILE directive and no exit 3 was specified on an OWNCODE directive. | Specify output file on FILE or OWNCODE directive. |
| UNDEFINED SEQUENCE | Collating sequence name was nonstandard, but was not followed by a user's unique collating sequence. | If standard collating sequence is desired, change sequence name accordingly. If user collating sequence is desired, specify sequence after nonstandard name. |
| VALUE OF P5 GT 6 | Value of 6C RECORD directive parameter P5 is restricted to $1 \leqslant p5 \leqslant 6$. P5 is set to 6 and p4 to p4 + p5 - 6. | See Sort/Merge Version 3 Reference Manual. |
| ZERO FIELD | Bytesize specified in BYTESIZE directive or MRL value specified in OWNCODE directive cannot be zero. | Specify bytesize or MRL field as value other than zero. |

## MACRO AND FORTRAN EXTENDED DIAGNOSTICS

The dayfile message for Sort/Merge macro or FORTRAN Extended call error will be one of the following:

n NON-FATAL SORT MACRO ERROR(S) or n FATAL SORT MACRO ERROR(S)

A diagnostic message is written to the output file for each error. The messages appear in three formats:

This format is returned in the user's assembly listing for errors detected during syntax checking.

P ERR message

message One of the diagnostic messages listed.

This format is returned to the dayfile; it involves errors encountered during cross checks made by Sort/Merge on internal tables generated from.macro specification or FORTRAN Extended calls.

SORTMRG **type**macro,message

type F Fatal error causing an abnormal termination of job

N Nonfatal error; job processing continued

macro Macro call in error

message One of the diagnostic messages listed

This format returns informative messages to the dayfile regarding Sort/Merge processing.

SORTMRG message

message One of the messages listed

The diagnostic messages for Sort/Merge macro or FORTRAN Extended call errors are listed in table B-2.

TABLE B-2. MACRO AND FORTRAN EXTENDED DIAGNOSTICS

| Message | Significance | Action |
|---|---|---|
| A LEGAL TYPE DOES NOT PRECEDE SEQ NAME, name | Sequence name indicated appeared before type parameter specification in KEY macro call. | Rearrange order of parameters. |
| AT LEAST THREE SCRATCH TAPES ARE REQUIRED | Less than three scratch tapes were specified in TAPE macro call. | Specify a minimum of four tapes for a balanced merge, three tapes for a polyphase merge. |
| BOTH BYTE AND BIT MAY NOT BE NULL | Byte and bit parameter specifications were omitted from KEY macro call. | Respecify KEY macro call with byte and bit parameters included. |
| BOTH NBYTE AND NBIT MAY NOT BE ZERO | Zero cannot be specified for both nbyte and nbit in KEY macro call. | Specify value other than zero for nbyte and/or nbit. |
| BOUNDARY OVERLAP | Character in character-coded sort key overlapped a word boundary. | Specify starting position of sort key on character boundary. |
| CODE SIZE CONFLICT | Sort key length for INTBCD or DISPLAY type key is not an integer multiple of character code size. | Specify length of sort key as integral multiple of 6 bits. |
| DUPLICATE COLLATION | One character appears more than once in user's collating sequence. | Delete duplicate occurrence of character in user's collating sequence. |
| DUPLICATE PARAMETER, param | The parameter indicated by param was specified more than once. | Delete all but one occurrence of parameter. |
| EMPTY MERGE FILE | No records contained in merge file. | None. |
| FIELD SIZE ≠ 60 BITS | Length of integer or floating point sort key is not 60 bits. | Respecify KEY macro call with length of sort key equal to 60 bits, or use LOGICAL type key. |
| FORMAT: NAME, (SEQUENCE), END | Format error detected in initial SEQUENCE macro call. | Correct macro call format. |
| ILLEGAL FILE TYPE, filetype | Type parameter in FILES macro call had illegal file type or was omitted. | Correct type parameter specification. |
| ILLEGAL NULL PARAMETER | First parameter of EQUATE macro call was specified as null but END did not follow it. | Correct macro call format. |

TABLE B-2. MACRO AND FORTRAN EXTENDED DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| ILLEGAL OWNCODE EXIT | Value other than 1 through 6 was specified for exit number of OWNCODE macro call. | Specify exit number as one of the digits 1 through 6. |
| ILLEGAL OWNCODE PARAM | Illegal parameter was specified in OWNCODE macro call. | Correct macro call parameter specification. |
| ILLEGAL PARAMETER, param | An illegal parameter was specified. | Correct parameter specification. |
| IMPROPER OCTAL PARAMETER, param | Param is illegal octal parameter. | Correct parameter specification. |
| INSUFFICIENT FIELD LENGTH FOR MERGE BUFFERS | Sort/Merge attempted to sort a file too large for it to handle with the memory allocated. | Increase field length of macro called sort with MAXCM parameter of SORT macro call; or increase field length of FORTRAN called sort with ba parameter of SMSORT call. |
| INSUFFICIENT SCM | Small core memory allocated is insufficient for Sort/Merge processing. | Same as above. |
| INVALID SEQUENCE | Collating sequence cannot be specified for a sort key not character coded. | Delete colseq parameter. |
| KEY NOT WITHIN RECRD | A sort key is not completely within the record size. | Specify MRL or FL for all input files. The second parameter within a field specification must be the length of the key; the second parameter is not the ending byte position. |
| MERGE ONLY BUFFERS EXCEED MEMORY LIMITS | Small core memory allocated is insufficient for Sort/Merge processing. | Increase field length of macro called sorts with MAXCM parameter of SORT macro call; or increase field length of FORTRAN called sorts with ba parameter of SMSORT call. |
| MISSING END PARAM | END parameter was omitted from SEQUENCE or EQUATE macro call. | Respecify macro call with END parameter included. |
| MISSING FILE PARAMETER | Type and name parameters were omitted from FILES macro call. | Respecify FILES macro call with type and name parameters included. |
| MISSING OWNCODE PARAMETER | MRL parameter was omitted from OWNCODE macro call. | Respecify OWNCODE macro call with MRL parameter included. |
| MISSING PARAMETER | Option parameters were omitted from OPTIONS macro call. | Delete OPTIONS macro call or respecify it with options included. |

TABLE B-2. MACRO AND FORTRAN EXTENDED DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| MORE THAN ONE OUT-FILE | More than one output file was specified on the FILES macro call. | Delete all but one output file. |
| MULTIPLE A/D PARAMETER, A | More than one A specification was included for order parameter in KEY macro call. | Delete extra occurrences of A. |
| MULTIPLE A/D PARAMETER, D | More than one D specification was included for order parameter in KEY macro call. | Delete extra occurrences of D. |
| MULTIPLE DUMP PARAMETER, DUMP, nn | DUMP parameter was specified more than once on OPTIONS macro call. nn is number of records specified in the DUMP parameter. | Delete extra occurrences of DUMP parameter. |
| MULTIPLE TYPE OR SEQUENCE, name | Type or colseq parameter indicated by name was specified more than once in KEY macro call. | Delete extra occurrences of type or colseq parameter. |
| MULTIPLY DEFINED | Parameter was specified in different ways within one run. | Delete all but one occurrence of parameter. |
| MULTIPLY DEFINED AT OWNCD EXIT num | Same exit number was assigned to more than one entry point name. num is the exit number specified in the OWNCODE macro call. | Assign each exit number to only one entry point name. |
| NEED MRL FOR FILE ... filename | MRL parameter was not specified on system FILE macro for file indicated by filename. | Specify MRL parameter on system FILE macro. |
| NO BYTESIZE PARAMETER | Bytesize parameter was omitted from BYTESIZE macro call. | Respecify BYTESIZE macro call with bytesize parameter included. |
| NO EQUATE STRING | Listing of all characters to be equated was omitted from EQUATE macro call. | Respecify EQUATE macro call with characters listed correctly. |
| NO LEGAL PARAMETER | No legal parameter specification was detected. | Correct parameter specification. |
| NO NAME AND NO EQUATE STRING | Initial EQUATE macro call was submitted without name parameter and equated characters. | Respecify EQUATE macro call with name parameter and equated characters included. |

TABLE B-2. MACRO AND FORTRAN EXTENDED DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| NO NAME AND NO SEQUENCE | Initial SEQUENCE macro call was submitted without sequence name parameter and sequence. | Respecify SEQUENCE macro call with sequence name parameter and sequence included. |
| NO OWNCODE EXIT PARAMS | No legal exit was specified on OWNCODE macro call. | Specify legal exit on OWNCODE macro call. |
| NO SEQUENCE | Initial SEQUENCE macro call was submitted without sequence. | Respecify SEQUENCE macro call with sequence included. |
| NON-STANDARD NAME WITH NO SEQUENCE | Nonstandard sequence name was specified without an associated user collating sequence. | If standard collating sequence is desired, change sequence name accordingly. If user collating sequence is desired, specify sequence after nonstandard name. |
| NULL PARAMETER IN CHAR STRING | No character position in character list of SEQUENCE or EQUATE macro call can be null. | Insert character into null position or delete null position. |
| ONE MRL (MAX RCD LENGTH) PARAM IS REQUIRED | MRL parameter was specified more than once or not at all. | Correct parameter specification. |
| ONE-ORIGIN BIT MAY NOT BE ZERO | Bit parameter was illegally specified as zero. | Change bit parameter to value other than zero. |
| ONE-ORIGIN BYTE MAY NOT BE ZERO | Byte parameter was illegally specified as zero. | Change byte parameter to value other than zero. |
| ONLY END OR NULL AFTER SEQUENCE | Sequence in SEQUENCE or EQUATE macro call was followed by a parameter other than END or null. | Respecify SEQUENCE or EQUATE macro call in correct format. |
| ONLY 1 MERGE FILE | Merge-only processing requires more than one presorted input file. | Include another merge file. |
| OPTION OUT-OF-PLACE | OPTIONS macro call must immediately follow the SORT or MERGE macro call. | Insert OPTIONS macro call directly after SORT or MERGE macro call. |
| OVER 100 SEQUENCES | More than 100 collating sequences were specified. | Reduce number of collating sequences to 100 or less. |
| OWNCODE > 6 PARAMS | More than six parameters were specified on OWNCODE macro call. | Delete all but six parameters. |

TABLE B-2. MACRO AND FORTRAN EXTENDED DIAGNOSTICS (Cont'd)

| Message | Significance | Action |
|---|---|---|
| PARAM AFTER END | Parameter was specified after END in SEQUENCE or EQUATE macro call. | Delete parameter after END or insert it before END. |
| SORT CANNOT USE SPECIFIED/DEFAULT CM – SORT USING xxxxx CM WORDS INSTEAD | Value of MAXCM parameter on SORT or MERGE macro call or ba parameter on SMSORT or SMMERGE was too small. | Specify a larger value. |
| STANDARD NAME AFTER SEQUENCE | Standard collating sequence name was specified with user collating sequence. | Change collating sequence name to nonstandard name. |
| UNDEFINED IN-FILE | An input file was not specified either on a FILES macro call or with exit 1 on an OWNCODE macro call. | Specify input file on FILES or OWNCODE macro call. |
| UNDEFINED I/O FILE | No files or owncode exits were specified for input or output. | Specify input or output file on FILES or OWNCODE macro call. |
| UNDEFINED KEY(S) | No sort key was specified. | Specify sort key on KEY macro call. |
| UNDEFINED OUT-FILE | An output file was not specified either on a FILES macro call or with exit 3 on an OWNCODE macro call. | Specify output file on FILES or OWNCODE macro call. |
| UNDEFINED SEQUENCE | Collating sequence was nonstandard, but was not followed by a user's unique collating sequence. | If standard collating sequence is desired, change sequence name accordingly. If user collating sequence is desired, specify sequence after nonstandard name. |

# INCOMPATIBILITIES C

The following information concerns the incompatibilities existing between Sort/Merge Version 3 and Sort/Merge Version 4. These incompatibilities require conversion of existing Sort/Merge Version 3 programs if they are to be executed under the new version of Sort/Merge.

1. Sort/Merge Version 3 macros are not compatible with Sort/Merge Version 4. Programs using Version 3 macros must have these macro calls manually recorded in the new Sort/Merge macro format, and the programs must be reassembled.

2. The default record-mark (default R type) record format with Sort/Merge Version 4 differs from Sort/Merge Version 3 for binary files in operating system format and S or L tapes. Sort/Merge Version 4 always reads/writes a BT=C, RT=Z file for default R type records. Sort/Merge Version 3 reads/writes logical records on binary files or unblocked physical records on S or L tapes for default R type records. System logical records (and physical records on S or L tapes) can be sorted by Sort/Merge Version 4, although they cannot be specified by the 6C directives alone. A FILE control statement preceding the SORTMRG(6C) control statement must specify RT=S, MRL=n, where n is the maximum record length in characters.

3. Parity errors are processed by Basic Access Methods (BAM) for Version 4 rather than by Sort/Merge for Version 3. Therefore, parity errors with the display option are recorded in an internal error file maintained by BAM rather than in the output file as for Sort/Merge Version 3. The file can be printed by executing the control statement CREMEP,LO.

   The default action for a parity error is job termination for Sort/Merge Version 4 rather than dropping of the bad record as for Sort/Merge Version 3.

4. The Sort/Merge Version 3 padding capability permits padding the last block of an S or L tape that contains only fixed length records and a fixed blocking factor. This padding feature allows the user to make the last block equal in length to all preceding blocks. BAM does not provide this capability and does not include the padding record counts as part of the dayfile tally messages. However, conversion to Sort/Merge Version 4 does provide, through BAM, an increased number of padding options.

5. Sort/Merge Version 4 does not support a tag sort capability; therefore tag sort specifications from Sort/Merge Version 3 control statements in existing jobs must be removed. This conversion results in a full record sort by Sort/Merge Version 4.

6. Sort/Merge Version 4 can accommodate labeled and multi-labeled tapes, but does not perform label processing. When Sort/Merge directives under Version 4 are used with a nonstandard labeled tape, the user must specify owncode exit 6. The label is sent to the user through this exit for checking before the system skips over the label to process the record data.

   Exit 6 is not available under Sort/Merge Version 3. Instead the nonstandard LABEL control statement is specified for Sort/Merge Version 3 when nonstandard labeled tapes are used. Use of this control statement positions the file immediately past the label, without label processing.

7. Owncode routines for Sort/Merge Version 4 have the record address residing in the location specified by the contents of the register A2. Under Sort/Merge Version 3, the record addresses for owncode routines reside in the location specified by the contents of A2+2 (there is a two word header). Therefore, Version 3 owncode routines must be modified to specify the record address in A2.

8. Exit 1 for Sort/Merge Version 4 inserts the user record after the original input record, whereas exit 1 under Sort/Merge Version 3 inserts the record before the original record. This difference in exit 1 execution is significant only when the user record and input record have identical keys, and the user has specified that the original order of records with identical keys is to be preserved. In the rare event that Version 3 ordering is required, the user can either change his record formats and/or keys, or he can revise his application to supply the record through owncode exits or the SORT verb in COBOL.

9. When input files described by Sort/Merge Version 3 control statements meet the following conditions:

   display coded,

   produced by the 6000 FORTRAN,

   each record on the file is terminated by a short PRU, or Level 0 PRU,

   BAM takes the end-of-data exit in sort after each record and Sort/Merge Version 4 interprets this on end-of-file. To overcome this situation, insert a FILE statement for the file ahead of the SORTMRG control statement. The FILE statement should describe the file with BT=K,RB=1,MRL=n.

10. Sort/Merge Version 3 can use blank common; blank common is preserved by Sort/Merge Version 4.

11. Sort/Merge Version 3, by default, preserved the original order of records with equal keys. For efficiency, Sort/Merge Version 4 does not preserve order by default. Use the RETAIN parameter of OPTIONS to force the original order of equal keys.

12. If a nonexistent file is specified for Sort/Merge Version 3, a REQUEST will be issued; Sort/Merge Version 4 assumes that the file resides on disk.

The following incompatibilities do not require conversion.

1. Record blocking and record types specified on Sort/Merge Version 3 control statements are related internally by Sort/Merge Version 4 to corresponding BAM control statement specifications defining record blocking and record types. Therefore, FILE control statements are not required with Sort/Merge Version 3 control statements in specifying blocking and record types.

2. Parity error procedures for Sort/Merge Version 4 must be specified on the FILE control statement. These parity error procedures, if already listed for Sort/Merge Version 3, will correspond to the new Sort/Merge procedures and therefore do not require FILE statement specifications.

3. The following parameters on Sort/Merge Version 3 SORT, SORTB, SORTP, and MERGE control statements are ignored by Sort/Merge Version 4. Processing of the Version 3 Sort/Merge job continues.

   M option in parameter 1 because multireel files are handled automatically by BAM

   Parameter 7

4. The maximum number of input files has increased from 32 to 100.

5. If p2 of the RECORD directive is V and decimal character count (RT=D) records are being described, the number of 6-bit bytes in the character count field (p5) cannot exceed 6. Version 4 Sort/Merge adjusts the values of p4 and p5 accordingly and issues a nonfatal error. If the source program exceeds this limit, the program must be revised.

# SORT/MERGE AND RECORD MANAGER D

This appendix describes the use of Record Manager by Sort/Merge Versions 1 and 4, with the primary emphasis on Version 4. It also describes, for the Sort/Merge user, aspects of Record Manager that he must know to use Sort/Merge, as well as some features that will be helpful in using Sort/Merge efficiently.

Sort/Merge performs all input and output through Record Manager, and it requires that all input and output files conform to Record Manager record and block structures. Files created through COBOL, ALGOL 4, or FORTRAN Extended are compatible with Record Manager file structures. To determine if files created through other products are compatible with Record Manager, and to determine record and block structure, consult the individual product reference manual. Files structured differently can perhaps be reformatted through the FORM utility.

The Sort/Merge user must provide file structure descriptions for each input or output file:

The Sort/Merge directive user must provide a FILE control statement for all files. Under Sort/Merge Version 4, FILE control statements are not necessary for the special system files INPUT, OUTPUT, and PUNCH. The FILE statement provides values for file information table fields needed to process the file; optional parameters may be included, subject to the restrictions outlined below. Sort/Merge will construct the actual table and set some of the fields. The FILE statement is the only part of Record Manager available to the user who restricts his processing to directives.

The Sort/Merge macro user must ensure each input or output file has a valid file information table at the time the macros are executed. In most cases, the Record Manager FILE macro is used for this purpose; the FILE statement also may be used to alter file information table fields when the file is opened. Since the full range of Record Manager processing is available to the macro user, the user should be aware of how Sort/Merge uses Record Manager, so that he or she does not interfere with normal Sort/Merge processing.

The FORTRAN Extended call user normally need not be concerned with Record Manager. FORTRAN Extended provides a file information table for all files specified in the PROGRAM statement. If the FORTRAN Extended Record Manager interface routines are used with the Sort/Merge calls, the user must ensure each input or output file has a valid file information table at the time the calls are executed. The user also should be aware of how Sort/Merge uses Record Manager, so that he does not interfere with normal Sort/Merge processing.

## FILE CONTROL STATEMENT

**Format**

FILE(lfn,field=value, . . . , field=value)

lfn Logical file name; 1 to 7 letters or digits, beginning with a letter.

field File information table field mnemonic, 2 or 3 characters.

value — Value to be placed in corresponding field; integer (assumed to be decimal unless a B suffix indicates octal) or symbolic (character string).

The FILE control statement specifies values for file information table fields to be set when the SETFIT macro is executed or the file is opened during execution. The FILE statement does not establish the file information table; rather, it causes the system to save the values specified. Subsequently, when a call is made to open the file, the saved values are placed in the file information table, overriding any previous contents.

Not all fields can be set by the FILE statement; in particular, fields whose values are relative storage locations cannot be so set (such as WSA, EX, DX, etc.). The descriptions of specific FIT fields given below indicate whether each can be set by the FILE statement.

Under Sort/Merge Version 4, some fields (such as C1 and SB) are set only by the FILE statement. In most cases, Sort/Merge does not change file information table values provided by the user on a FILE statement; but Sort/Merge will reset a field if it requires a specific value that differs from the value provided by the user. The special system files INPUT, OUTPUT, and PUNCH do not require FILE control statements under Sort/Merge Version 4, since BAM provides the following default characteristics for these files:

| File Name | Block Type (BT) | Record Type (RT) | Fixed Length (FL) |
|---|---|---|---|
| INPUT | C | Z | 80 |
| OUTPUT | C | Z | 140 |
| PUNCH | C | Z | 80 |

Under Sort/Merge Version 1, a FILE control statement that specifies CF=N and OF=N must be provided for INPUT, OUTPUT, and PUNCH. The same statement should specify these values:

| File Name | Block Type (BT) | Record Type (RT) |
|---|---|---|
| INPUT, OUTPUT, or PUNCH | Unblocked | W |

## RELEVANT FILE INFORMATION TABLE FIELDS

File information table fields applicable to sequential files that can be set by the user are listed below. For each field, indication is made whether it can be set by the FILE control statement, whether and under what circumstances Sort/Merge will alter a user-provided value, and what default value is provided if the user does not set the field. For a complete description of each field and possible values, consult the appropriate Record Manager reference manual.

ASCII — Character set bits for terminal processing. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is 64-character display code.

BBH — Buffer allocated below highest high address. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is NO.

BFS — Buffer size in words. Can be set by FILE statement.

If BFS for any file is set by the user to a value other than 0, Sort/Merge does not change user setting. If BFS $\neq$ 0 and FWB $\neq$ 0, Sort/Merge uses the specified buffer; if BFS $\neq$ 0 and FWB = 0, Sort/Merge sets FWB and allocates a buffer whose length is given by BFS. For all files whose BFS is 0, Sort/Merge calculates a reasonable buffer size for each such file and then sets BFS of all these files to the largest value calculated. The algorithm used to calculate buffer size is described under the title "Buffer Size Calculation" below.

BT Block type. Can be set by FILE statement. Sort/Merge does not alter user setting; block type chosen may require setting of additional file information table fields.

Default for Sort/Merge Version 4 is BT=I.

Default for Sort/Merge Version 1 is unblocked.

CF File disposal code. For directive sorts or merges, user setting of CF has no effect, because Sort/Merge disposes file in accordance with file disposition code specified on FILE directive. For macro sorts, Sort/Merge takes no file disposal action; files are not closed before control is returned to the user.

Under Sort/Merge Version 1, a FILE control statement must be provided for the special system files INPUT, OUTPUT, and PUNCH in order to specify CF=N and OF=N.

CL Length in characters of count field for T type records. Can be set by FILE statement; Sort/Merge does not alter user setting. No default when RT=T.

CM Conversion mode for tapes. Can be set by FILE statement. Sort does not alter user setting; used in calculation of BFS. Default is NO.

CNF Connected file flag. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is NO.

CP Starting location of count field for T type record. Can be set by FILE statement. Sort/Merge does not alter user setting. Default is 0.

C1 Count field of D or T type record is COMPUTATIONAL-1. Can be set by FILE statement. Sort/Merge does not alter user setting. Default is NO.

DFC Dayfile error message control. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is 0.

DX Data exit for end-of-section, end-of-partition, end-of-information. Sort/Merge sets this field unconditionally; user setting is ignored.

EFC Error file message control. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is 0.

EO Error option for parity errors. Not applicable for NOS 1. Can be set by FILE statement. Sort/Merge does not alter user setting. Default is T (terminate file under NOS/BE 1; terminate job under SCOPE 2.1).

ERL Trivial error limit. Can be set by FILE statement. If ERL=0, Sort/Merge sets to 1. If ERL≠0, Sort/Merge does not alter user setting. Default is 0.

EX Error exit routine. Can be set by FILE macro but not by FILE statement. Sort/Merge does not alter user setting. Default is no exit routine.

FL Fixed length of F and Z type records; same field as MRL. Can be set by FILE statement; Sort/Merge does not alter user setting. FL is required for files with RT=F or Z; either MRL or FL is required for all files processed by Sort/Merge.

FO — File organization. Can be set by FILE statement. Sort/Merge requires FO=SQ, sequential files, which is default.

FWB — Address of I/O buffer. Cannot be set by FILE statement. Sort/Merge does not alter user setting. If not set by user, Sort/Merge allocates a buffer of BFS words and sets FWB to its address. FWB≠0, BFS=0 is a nonfatal error; Sort/Merge overrides user setting of FWB in this case.

HL — Header length in characters for T type records. Can be set by FILE statement; Sort/Merge does not alter user setting. No default when RT=T.

LA — Address of label area. Cannot be set by FILE statement. Sort/Merge sets LA when LT=NS, ULP≠NO, and an Exit 6 owncode routine has been provided, and ignores it otherwise.

LBL — Length of label area in characters. Sort/Merge sets this field when LT=NS, ULP≠NO, and an Exit 6 owncode routine has been provided, and ignores it otherwise.

LCR — Label check/creation. Must be specified on FILE statement for tape sorts. Sort/Merge does not alter user setting. No default.

LFN — Logical file name, one to seven characters beginning with a letter. Can be set by FILE statement; Sort/Merge does not alter user setting. No default.

LL — Length in characters of count field for D type records. Can be set by FILE statement; Sort/Merge does not alter user setting. No default when RT=D.

LP — Starting position of count field for D type records. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is 0 (first character of record).

LT — Label type for tapes. Can be set by FILE statement. Sort/Merge does not alter user setting. Default S (ANSI standard labels).

LX — Label exit for user label processing routine. Sort/Merge sets LX only if LT=NS, ULP≠NO, and an Exit 6 owncode routine is provided, and ignores it otherwise.

MBL — Maximum block length. Can be set by FILE statement; Sort/Merge does not alter user setting. For defaults, see the appropriate Record Manager reference manual.

MNB — Minimum block length for K and E blocks. Can be set by FILE statement; Sort/Merge does not alter user setting. No default.

MNR — Minimum record length in characters. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is 0.

MRL — Maximum record length in characters. Can be set by FILE statement. Sort/Merge does not alter user setting. Either MRL or FL is required for every file processed by Sort/Merge; MRL is required on the OWNCODE directive or macro if no input file is specified.

MUL — Multiple of characters for padding on K or E type blocks. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is MUL=2.

OC Open/close status. Not set by user; referenced internally by Sort/Merge. For a macro sort or merge, a file may be either open (OPE), closed (CLO), or never opened (NOP) when control is transferred to Sort/Merge; if the file is open (OPE), or closed (CLO), Sort/Merge assumes its FIT is valid.

OF Open flags. Sort/Merge rewinds every file whose logical file name is not OUTPUT; user setting is overridden. If a file is already open when control is transferred to Sort/Merge, its current position is not changed. Under SCOPE 2.1, FILE control statements must be provided for the special system files INPUT, OUTPUT, and PUNCH to specify OF=N and CF=N.

PC Padding character for K and E type blocks. Specified in display code. It must not be the same as the record mark character. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is 76 (octal).

PD Processing direction. Can be set by FILE statement. Sort/Merge requires that input files be open for input or for input/output, and that output files be open for output or for input/output. If PD is set incorrectly, Sort/Merge will reset it to INPUT for input files and to OUTPUT for output files.

RB Number of records per block for K type blocks. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is RB=1.

RL Record length in characters. Cannot be set by FILE statement. This field is used internally by Sort/Merge; the user is not involved with its handling. If the field is set when control is transferred to Sort/Merge, the user setting is overridden.

RMK Record mark character for R type records. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is 62 (octal).

RT Record type. Can be set by FILE statement. Sort/Merge does not alter user setting and permits any record type except U. Default is RT=W. SCOPE 2 permits RT=U when BT=K and RB=1.

SB Length field for D and T type records has a COBOL sign overpunch. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is NO.

SBF Suppress allocation of buffers and circular buffering. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is NO.

SPR Suppress read ahead and write behind. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is NO.

TL Trailer length in characters for T type records. Can be set by FILE statement; Sort/Merge does not alter user setting. No default when RT=T.

ULP User label processing. Can be set by FILE statement; Sort/Merge does not alter user setting. An Exit 6 owncode routine is valid only when LT=NS and ULP≠NO. Default is ULP=NO.

VF Volume disposition code. Can be set by FILE statement; Sort/Merge does not alter user setting. Default is VF=U.

WSA Address of current record. Cannot be set by FILE statement. Used internally by Sort/Merge; user setting is overridden.

## SORT/MERGE VERSION 4 USE OF BAM

The remainder of this appendix is intended primarily for macro Sort/Merge users; it is applicable only to Sort/Merge Version 4. Directive Sort/Merge users need not be concerned with internal use of BAM by Sort/Merge, since they can access BAM only through the FILE statement.

When the macro user transfers control to Sort/Merge, files may be open, closed, or never opened. If a file is open, Sort/Merge assumes it has a valid file information table. In particular, processing direction (PD) must be set correctly, as well as MRL or FL, RT, BT, etc., and the file must have a valid buffer (BFS and FWB must be set properly in the FIT). Closed files (files with OC=CLO) and files that have never been opened (files with OC=NOP) will be opened and rewound. File position of open files (files with OC=OPE) is not changed by Sort/Merge.

In most cases, Sort/Merge does not alter file information table fields set by the user. Some fields are set unconditionally, and user settings are overridden (WSA, RL, DX, LX). Other fields are set only if the user did not set them (FWB, BFS, ERL). The processing direction field (PD) is changed only when an inappropriate value was provided. If FWB≠0 and BFS=0, an error message is issued and the user setting for FWB is overridden.

### FILE INFORMATION TABLE MANIPULATION FOR INPUT FILES

The procedures outlined below comprise a summary rather than a precise algorithm for how Sort/Merge handles the file information table. The purpose of this discussion is to clarify, for macro Sort/Merge users, the order and manner in which FIT fields are set, so the user knows what to expect when control is returned to him through an EX error exit, at the completion of Sort/Merge processing, or in an owncode exit routine.

Sort/Merge processes input files as follows:

1. If ERL=0, sets ERL to 1: otherwise not changed.

3. Performs initialization (described below).

4. If FWB≠0, goes to step 7. (The error case FWB≠0, BFS=0 is diagnosed during initialization.)

5. Sets FWB to the address of the allocated buffer.

6. If BFS=0, sets BFS for this file to the largest value of BFS for any sort input file.

7. If RT=Z for this input file and RT=F for the output file, saves the output FL for use as input RL, to preserve trailing blanks.

8. If LFN=INPUT, sets DX to address of routine described in step 22.

9. If LFN≠INPUT, sets DX to address of routine described in step 21.

10. If the file is open (OC=OPE), goes to step 17.

11. If LT≠NS, goes to step 13.

12. This file has nonstandard labels; they should be processed either by an Exit 6 owncode routine or according to the 6C LABEL directive. LX is set to address of routine described in step 23 if an Exit 6 owncode routine is provided and to address of routine described in step 26 otherwise.

14. If LFN=INPUT, performs

    OPENM fit,INPUT

    and goes to step 17.

15. If PD≠I-O, sets PD=INPUT.

16. Performs

    OPENM fit,,R

17. Sets WSA appropriately.

18. Reads a record with the GET macro.

19. Length of record read is determined either by RL field of FIT or as in step 7.

20. Goes to step 17.

21. Data exit for all files except INPUT. If this exit was taken at end-of-section, goes to step 17.

22. Data exit for end of section on INPUT, end of partition or end of information on any other file. Disposes file according to Sort/Merge disposition code:

| | |
|---|---|
| N | ENDFILE fit |
| CR, RC | CLOSEM fit,R |
| C | CLOSEM fit,N |
| U, CU | CLOSEM fit,U |
| R | REWINDM fit |

23. Label exit to interface with the Exit 6 owncode routine. Reads a label with GETL and calls the Exit 6 owncode routine, returning to step 24 or step 25.

24. Exits to OPENM or CLOSEM with a CLOSEL.

25. Verifies label and goes to step 23.

26. Label exit to implement the 6C LABEL directive. Reads a label with GETL, verifies it, and exits to OPENM or CLOSEM with a CLOSEL.

## FILE INFORMATION TABLE MANIPULATION FOR MERGE FILES

Sort/Merge processes merge files as follows:

1. If ERL=0, sets ERL=1.

3. Performs initialization (described below).

4. Sets WSA.

6. If FWB=0, sets FWB.

7. If RT=Z for this file and RT=F for the output file, saves output FL for use as input RL to preserve trailing blanks.

8. If LFN=INPUT, sets DX to address of routine described in step 21.

9. If LFN≠INPUT, sets DX to address of routine described in step 20.

10. If file is open (OC=OPE), goes to step 17.

11. If LT≠NS, goes to step 13.

12. File has nonstandard labels. The labels should be processed either by an Exit 6 owncode routine or according to the 6C LABEL directive. LX is the same routine as described in input step 23 if an Exit 6 routine is provided, and the same routine as described in input step 26 otherwise.

14. If LFN=INPUT, performs

    OPENM fit,INPUT

    and goes to step 17.

15. If PD≠I-O, sets PD=INPUT.

16. Since LFN≠INPUT, performs

    OPENM fit,,R

17. Reads a record.

18. Assumes record length is given by RL field, unless step 7 applies.

19. Goes to step 17.

20. Data exit for all files except INPUT. Goes to step 17 if this exit was taken because of end-of-section (FP=EOS).

21. Data exit for end-of-section on INPUT, and end-of-partition and end-of-information on all files. Dispose the file according to the Sort/Merge disposal code:

| | |
|---|---|
| N | ENDFILE fit |
| CR, RC | CLOSEM fit,R |
| C | CLOSEM fit,N |
| U, CU | CLOSEM fit,U |
| R | REWINDM fit |

## FILE INFORMATION TABLE MANIPULATION FOR OUTPUT FILES

Sort/Merge processes output files as follows:

1. If ERL=0, sets ERL to 1.

3. Performs initialization (described below).

4. If the file is open (OC=OPE), goes to step 12.

5. If FWB=0, sets FWB.

7. If PD≠I-O, sets PD=OUTPUT.

8. If LFN=OUTPUT, performs

```
OPENM  fit,,N
```

and goes to step 12.

9. If LT=NS, goes to step 11.

10. File has nonstandard labels. They should be processed either by an Exit 6 owncode routine or according to the 6C LABEL directive. If an Exit 6 owncode routine is provided, sets LX to the address of the routine described in step 16; otherwise, sets LX to the address of the routine described in step 19.

11. Performs

```
OPENM  fit,,R
```

12. If no more records remain to be output, goes to step 15.

13. Sets WSA and RL.

14. Writes a record and goes to step 12.

15. Disposes the file according to the Sort/Merge file disposition code:

| | |
|---|---|
| N | ENDFILE fit |
| CR, RC | CLOSEM fit,R |
| C | CLOSEM fit,N |
| U, CU | CLOSEM fit,U |
| R | REWINDM fit |

16. Label exit to interface with the Exit 6 owncode routine. Calls the Exit 6 owncode routine, returning to step 17 or 18.

17. Writes user label with PUTL and exits to OPENM or CLOSEM with a CLOSEL.

18. Writes user label with PUTL and goes to step 16.

19. Label exit to implement the 6C LABEL directive. LA and LBL have already been set, so writes the label with PUTL and exits to OPENM or CLOSEM with a CLOSEL.

## INITIALIZATION FOR ALL FILES

1. If this file has ever been open (OC≠NOP), goes to step 11.

2. If BFS≠0, goes to step 7.

3. If FWB=0, goes to step 5.

4. A user setting of FWB≠0,BFS=0 is prohibited as the buffer size calculated by Sort/Merge might be so large as to cause overwriting of valid user data. Therefore displays the error message:

   LFN=axxxxxx,BFS=0,FWB NZ, FWB IGNORED

   and sets FWB to 0.

5. Uses the STATUS macro to determine whether the file is on disk or tape. A nonexistent file is assumed to be on disk (such a file can only be an output file). A tape with CM=YES is a BCD tape, and a tape with CM=NO is a binary tape.

6. Chooses a reasonable buffer size based on file residency and MBL. Sets BFS to the computed value; saves this value for use in step 8.

7. Performs SETFIT. This BAM routine sets defaults for BT, RT, and FL for files INPUT, OUTPUT, and PUNCH, then accesses the latest FILE control statement to update the file information table, and sets OC to NOP.

8. If BFS=0 after FILE statement processing, sets BFS to the value computed in step 6.

9. If file not a sort input file, goes to step 11.

10. If BFS for this file is greater than the maximum BFS for all sort input files, sets the maximum to this BFS. If Sort/Merge is to allocate the buffer (FWB=0), sets BFS to 0. (It will be set to the maximum at input step 5.)

11. Updates the MRL of all files except the output file, so that the MRL of all files is equal to the greatest value of any provided.

## BUFFER SIZE CALCULATION

If the user does not specify a buffer size (BFS) for a particular file, Sort/Merge (rather than BAM) calculates a value. An attempt is made by BAM to minimize buffer size, but Sort/Merge makes use of larger buffers to increase the speed of input/output. The following algorithm is used:

1. Sets BFS to 2*MBL, rounded up to a multiple of PRU size for the device.

2. If step 1 would result in BFS > 4000B, sets BFS to MBL, rounded up to a multiple of PRU size for the device.

3. If BFS < default for device, sets BFS to default for device.

4. Adds 3 to BFS. One word is added so the in pointer of the file information table is not equal to the out pointer when the buffer is exactly full. Two words are added so that 2*MBL will not fill the buffer for an S or L tape.

PRU sizes for devices:

| | |
|---|---|
| Disk | 100B |
| BCD Tape | 200B |
| Binary Tape | 1000B |

Default buffer sizes for devices:

| | |
|---|---|
| Disk | 2000B (16 PRUs) |
| BCD Tape | 1400B (6 PRUs) |
| Binary Tape | 3000B (3 PRUs) |

## SAMPLE PROGRAMS

### USER ERROR EXIT ROUTINE

This sample program demonstrates a macro sort with a user error exit subroutine. The main program begins at the entry point SORT; during a sort, when an error is encountered, the routine with entry point ERREXIT is called. This subroutine determines if error was type 142, excess data. The writer of this program evidently decided excess data would not be a problem; if an input record longer than 20 characters is read, the excess data is truncated. Any other error produces an abort. If excess data was encountered, + is put in column 19 of the record. Then SDS is set to NO, and ERL to 0 so needless error messages are displayed, and the program is not aborted when Record Manager gets control. A count is kept of the type 142 errors encountered.

After Sort/Merge processing is completed, control in the main program advances to the line containing ERRCOUNT and a message indicating the number of errors is sent to the dayfile.

Source code for this COMPASS program:

```
          IDENT   SORT
          ENTRY   SORT
SORT      SORT
          FILES   (INPUT,SORTIN),(OUTPUT,SORTOUT)
          KEY     11,,5,,DISPLAY
          KEY     1,,10,,LOGICAL
          SA1     ERRCOUNT      CONVERT ERROR COUNT
          RJ      =XCONVERT       TO DISPLAY CODE
          SA6     ERRMESS       INSERT INTO MESSAGE
          MESSAGE  ERRMESS,,R   DISPLAY THE ERROR COUNT
          ENDRUN                END THIS JOB STEP
*
ERRCOUNT  DATA    0             COUNT OF RM ERROR 142-S
ERRMESS   BSS     1
          DATA    28L LONG RECORDS WERE TRUNCATED
SORTIN    FILE    LFN=SORTIN,BT=C,RT=Z,FL=20,EX=ERREXIT,ERL=2
SORTOUT   FILE    LFN=SORTOUT,BT=C,RT=F,FL=20
*
*    ERROR ROUTINE -  B1=1, A0=FIT
ERREXIT                         ENTRY/EXIT WORD
          FETCH   A0,IRS,X1     IF NOT A 142 ERROR
          SX1     X1-142B         THEN ABORT
          NZ      X1,400000B+*
          FETCH   A0,WSA,X1     GET FWA RECORD
          SA2     X1+B1         INSERT A PLUS
          MX0     60-6            INTO COL. 19
          SX3     1R+
          BX6     -X0*X2
          IX6     X6+X3
          SA6     A2
          STORE   A0,SDS=NO     DO NOT DISPLAY ERROR MESSAGES
          STORE   A0,ERL=0      ALLOW INFINITELY MANY TRIVIAL ERRORS
          SA1     ERRCOUNT      ADVANCE LOCAL ERROR COUNT
          MX6     59            (X6 = -1)
          IX6     X1-X6
          SA6     A1
          EQ      ERREXIT       EXIT TO *GET*
*
          END     SORT
```

The following message appears when this program is executed:

```
********12 LONG RECORDS WERE TRUNCATED
```

## USE OF BFS TO REDUCE CENTRAL MEMORY REQUIRED

This directive merge illustrates user specification of BFS to reduce the amount of central memory required to run the job. Since 20 files are to be merged, memory requirements under default processing would be prohibitive; if Sort/Merge were allowed to calculate BFS, 2003B words would be required for each file, or a total of 50074B, in addition to other memory required by Sort/Merge. To reduce this requirement, the user takes advantage of the fact that the records in all the files are relatively short (40 words); therefore, he specifies a buffer size that is just as large as 1 record, rounded up to 65 words (1 PRU + 1 word). This way the entire merge is able to complete in only 30000B words. The execution time required, however, will probably be greater than it would be if larger buffers were used, since more mass storage accesses are required.

Directives for this job, as listed on OUTPUT file:

```
1  MERGE
2  FILES      OUTPUT = OUT
3  FILES      MERGE = IN01, IN02, IN03, IN04, IN05, IN06, IN07,
4  ,                  IN08, IN09, IN10, IN11, IN12, IN13, IN14,
5  ,                  IN15, IN16, IN17, IN18, IN19, IN20
6  FIELD      KEY(2, 1, LOGICAL)
7  KEY        KEY
8  END
```

Dayfile:

```
16.48.52.FILE(OUT,BT=C,RT=Z,FL=80)
16.48.56.FILE(IN01,BT=C,RT=Z,FL=40,BFS=65)
16.48.57.FILE(IN02,BT=C,RT=Z,FL=40,BFS=65)
16.48.57.FILE(IN03,BT=C,RT=Z,FL=40,BFS=65)
16.48.58.FILE(IN04,BT=C,RT=Z,FL=40,BFS=65)
16.49.00.FILE(IN05,BT=C,RT=Z,FL=40,BFS=65)
16.49.00.FILE(IN06,BT=C,RT=Z,FL=40,BFS=65)
16.49.02.FILE(IN07,BT=C,RT=Z,FL=40,BFS=65)
16.49.17.FILE(IN08,BT=C,RT=Z,FL=40,BFS=65)
16.49.22.FILE(IN09,BT=C,RT=Z,FL=40,BFS=65)
16.49.25.FILE(IN10,BT=C,RT=Z,FL=40,BFS=65)
16.49.29.FILE(IN11,BT=C,RT=Z,FL=40,BFS=65)
16.49.34.FILE(IN12,BT=C,RT=Z,FL=40,BFS=65)
16.49.36.FILE(IN13,BT=C,RT=Z,FL=40,BFS=65)
16.49.37.FILE(IN14,BT=C,RT=Z,FL=40,BFS=65)
16.49.39.FILE(IN15,BT=C,RT=Z,FL=40,BFS=65)
16.49.42.FILE(IN16,BT=C,RT=Z,FL=40,BFS=65)
16.49.44.FILE(IN17,BT=C,RT=Z,FL=40,BFS=65)
16.49.46.FILE(IN18,BT=C,RT=Z,FL=40,BFS=65)
16.49.47.FILE(IN19,BT=C,RT=Z,FL=40,BFS=65)
16.49.50.FILE(IN20,BT=C,RT=Z,FL=40,BFS=65)
16.49.53.RFL(30000)
16.51.09.SORTMRG.
16.52.35. ** INSERTIONS DURING INPUT    *********0
16.52.35. ** DELETIONS DURING INPUT     *********0
16.52.35. ** TOTAL RECORDS SORTED       *********0
16.52.35. ** INSERTIONS DURING OUTPUT   *********0
16.52.35. ** DELETIONS DURING OUTPUT    *********0
16.52.35. ** TOTAL RECORDS OUTPUT       ********60
16.52.35. **END MERGE RUN
```

# TAPE SORT/MERGE PROCESSING OPTIONS E

The tape variant of Sort/Merge provides two forms of Sort/Merge processing; balanced and polyphase. Either form is specified by the user on the SORT directive or the SORT macro call.

The following sections are provided to give the user a graphic description of both balanced and polyphase sort processing related to the three phases of Sort/Merge execution; the internal sort phase, the intermediate merge phase, and the final merge phase.

## BALANCED SORT

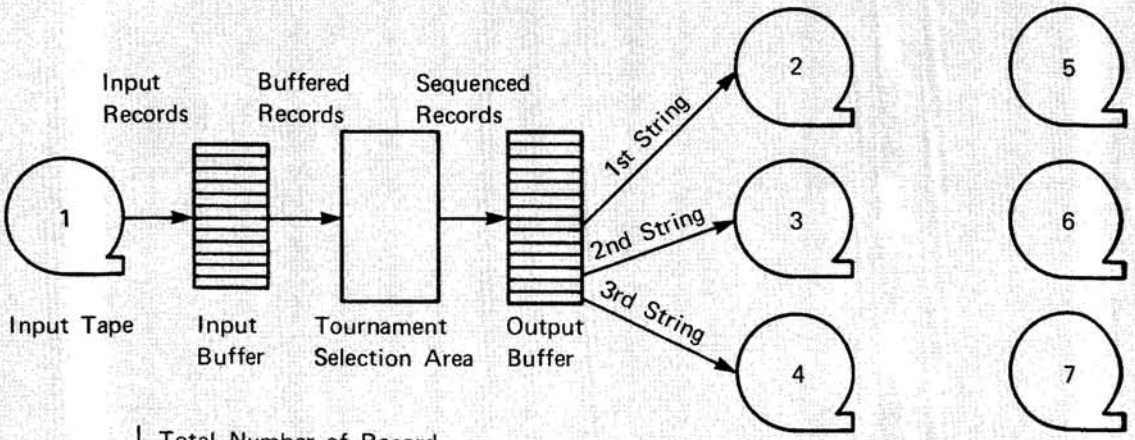
### INTERNAL SORT PHASE

Strings are written onto half of the assigned work tapes consecutively in the internal sort phase.

Example:

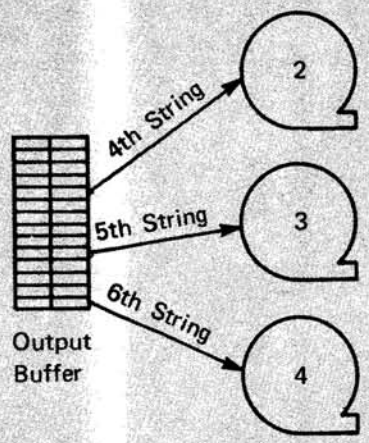
Record string distribution for the internal sort phase using the balanced sort process. The output for this example consists of nine record strings. The tape directive specified six intermediate merge tapes.

**SORT PASS 1**



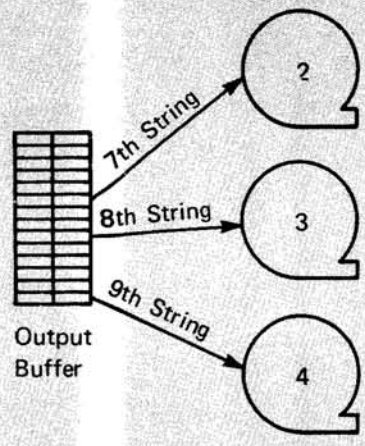

| Tape No. | Total Number of Record Strings on Tapes after Each Pass |
|---|---|
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| | Pass 1 |

**SORT PASS 2**




| Tape No. | Total Number of Strings on Tapes after Each Pass | |
|---|---|---|
| 2 | 1 | 2 |
| 3 | 1 | 2 |
| 4 | 1 | 2 |
| | Pass 1 | Pass 2 |

**SORT PASS 3**




| Tape No. | Total Number of Strings on Tapes after Each Pass | | |
|---|---|---|---|
| 2 | 1 | 2 | 3 |
| 3 | 1 | 2 | 3 |
| 4 | 1 | 2 | 3 |
| | Pass 1 | Pass 2 | Pass 3 |

## INTERMEDIATE MERGE PHASE

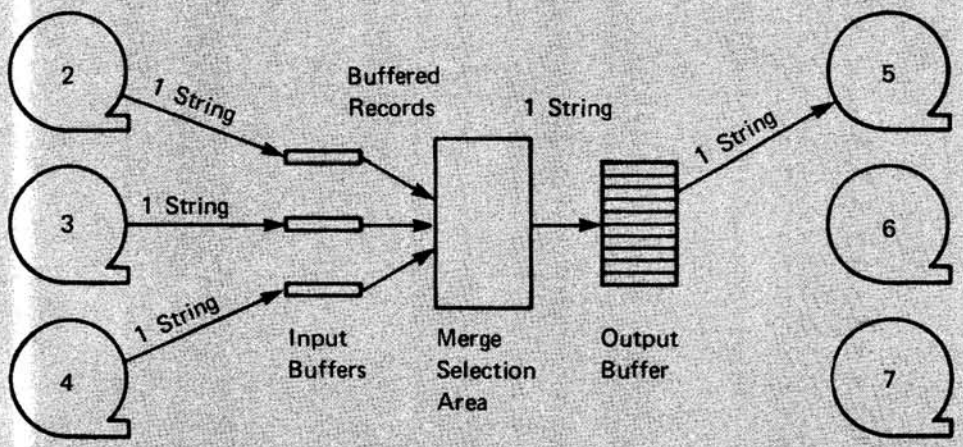
During each pass of the intermediate merge phase in a balanced merge, half of the tapes are used for input to the merge and half are used as output. The tapes are rewound after each pass and the output tapes from one pass became input tapes for the next pass.

Example:

### MERGE PASS 1

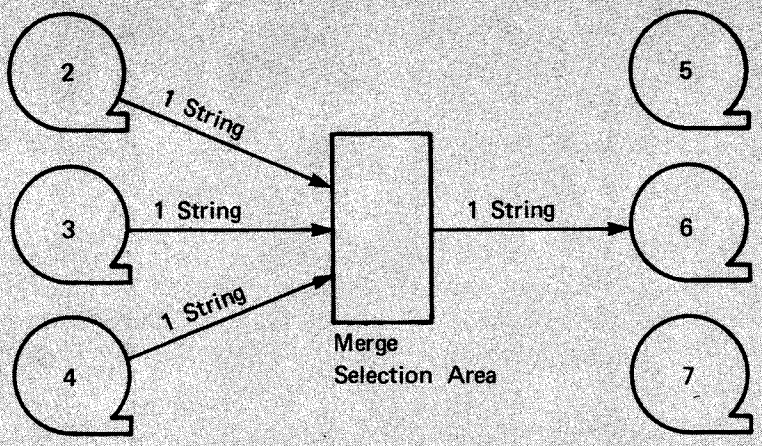
The following example of the balanced merge uses the record string distribution from the internal sort example. For the intermediate merge phase, 5, 6, and 7 function as output tapes for the first merge pass. Tapes 2, 3, and 4 will serve as input units for this pass. The distribution pattern involves record strings read off tapes 2, 3, and 4 and merged into a single string in the merge selection area. The newly merged strings are then written on tapes 5, 6, and 7 consecutively.

### FIRST MERGE




| Tape No. | Number of Strings on Tapes after Each Merge |
|---|---|
| 2 | 2 |
| 3 | 2 |
| 4 | 2 |
| 5 | 1 |
| 6 | 0 |
| 7 | 0 |
| | Merge 1 |

**SECOND MERGE**




| Tape No. | Number of Strings on Tapes after Each Merge | |
|---|---|---|
| 2 | 2 | 1 |
| 3 | 2 | 1 |
| 4 | 2 | 1 |
| 5 | 1 | 1 |
| 6 | 0 | 1 |
| 7 | 0 | 0 |
| | Merge 1 | Merge 2 |

At this point, there is at most one string on each tape. The tapes used as output on the merge pass just completed (5 and 6) are rewound. The tapes used as input (2, 3, and 4), on each of which one string remains to be read, are not rewound, and the final merge phase is entered.

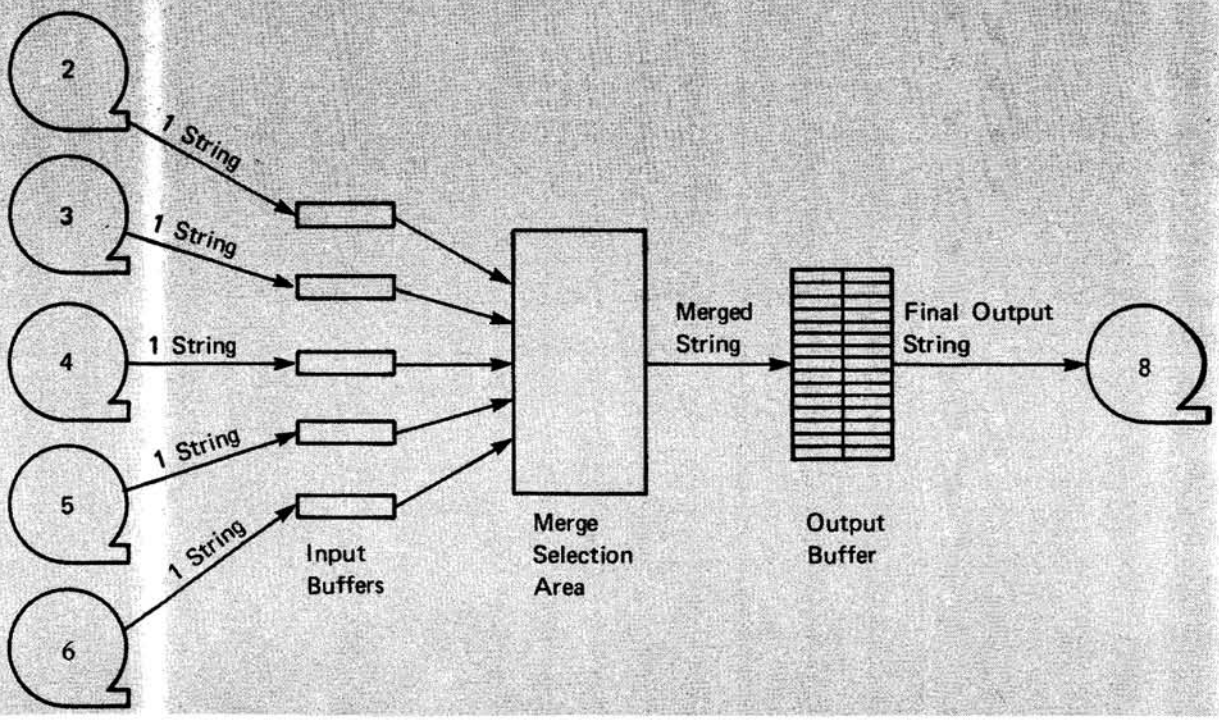
If in this example more record strings had been on tapes 2, 3, and 4, the intermediate merge would continue, following the same merge distribution pattern until all record strings were read off these three tapes. The two sets of work tapes reverse operations; that is, the input tapes become output tapes and the output tapes become input tapes. The reversal of tape functions continues until not more than a single record string is located on each tape of the work tape group.

## FINAL MERGE PHASE

The final merge involves merging of the record strings from the intermediate merge phase with any presorted files on the final output file.

Example:

The single record strings located on tapes 2, 3, 4, 5, and 6 at the completion of the intermediate merge phase are merged into a final record string and written onto the final output file specified by the user.



## POLYPHASE SORT

The polyphase sort routine during internal sort phase processing distributes record strings onto n minus 1 tapes specified by the user, where the distribution pattern is based on the Fibonacci sequence numbers. The internal sort phase distributes the sequenced record strings in the following manner:

1. Reserve the last tape ($T_n$) in a list of n tapes on the TAPE directive or macro for the first merge output. No strings are written on this tape during the internal sort phase distribution.

2. Write one string on each tape in the list $T_1, T_2, T_3 \ldots T_{n-1}$

3. Skip $T_1$, and distribute the number of strings which are on $T_1$ onto each of the other tapes in the list $T_2, T_3, T_4, \ldots T_{n-1}$

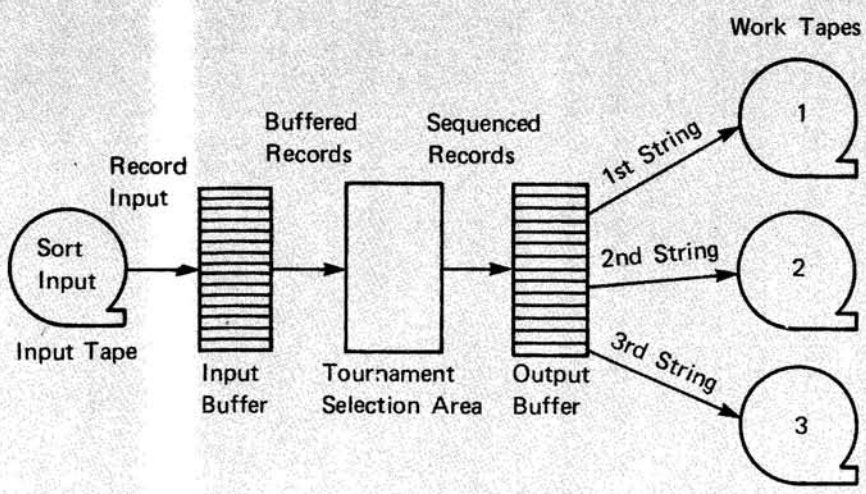
4. Skip the next tape ($T_2$) in the list and distribute the number of strings which are on $T_2$ onto each tape in the list $T_1, T_3, T_4, \ldots T_{n-1}$.

The following section gives a graphic example of this distribution pattern.

Example:

Distribution pattern of record strings for the internal sort phase using a polyphase sort. Output for this example consists of 17 record strings. The tape directive specified four intermediate tapes.

**FIRST PASS**

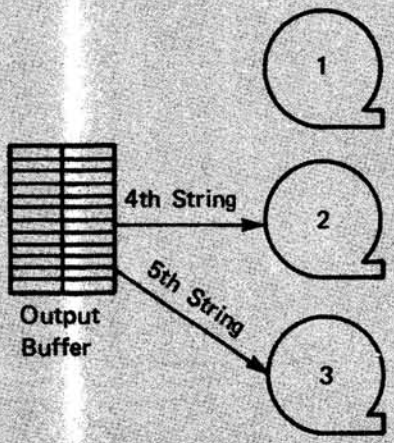



| Tape No. | Total Number of Strings/Tape |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |

Pass 1

**SECOND PASS**

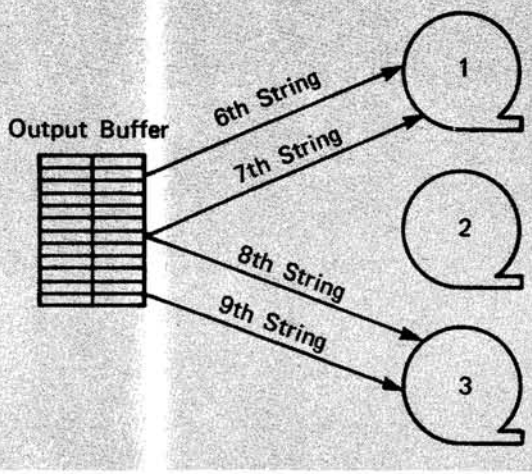
Work tape 1 is bypassed and the fourth and fifth record strings are placed on tapes 2 and 3.




| Tape No. | Total Number of Strings/Tape | |
|---|---|---|
| | Pass 1 | Pass 2 |
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 1 | 2 |

**THIRD PASS**

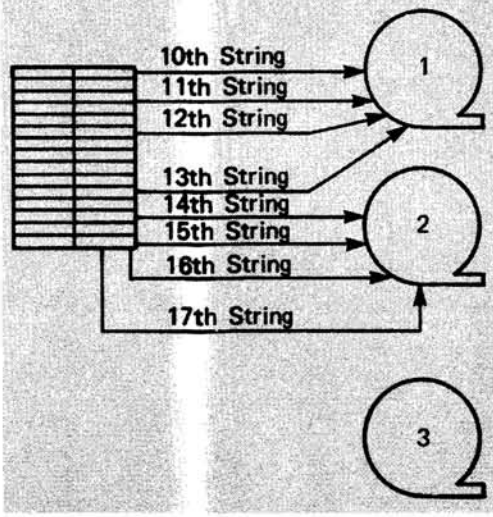
Tape 2 is bypassed and a number of record strings is written to tapes 1 and 3 equal to the number of record strings on tape 2 at the completion of pass 2. That is, two record strings are written on each of tapes 1 and 3.




| Tape No. | Total Number of Strings/Tape | | |
|---|---|---|---|
| | Pass 1 | Pass 2 | Pass 3 |
| 1 | 1 | 1 | 3 |
| 2 | 1 | 2 | 2 |
| 3 | 1 | 2 | 4 |

**FOURTH PASS**

Tape 3 is skipped and tapes 1 and 2 each receive four record strings (based on an amount equal to the number of record strings located on tape 3 at the completion of pass 3).




| Tape No. | Total Number of Strings/Tape | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 3 | 7 |
| 2 | 1 | 2 | 2 | 6 |
| 3 | 1 | 2 | 4 | 4 |
| | Pass 1 | Pass 2 | Pass 3 | Pass 4 |

The total number of record strings on tapes 1, 2, and 3 at the completion of the internal sort phase consists of the following distribution pattern:

| Tape No. | Total Number Record Strings |
|---|---|
| 1 | 7 |
| 2 | 6 |
| 3 | 4 |
| | Total |

If a larger number of record strings was to be generated during this internal sort phase, Sort/Merge would continue to distribute the remaining record strings by bypassing tape 1 and writing a number of strings onto tapes 2 and 3 equal to the number of strings located on tape 1.

## INTERMEDIATE MERGE PHASE

For a polyphase merge, the intermediate merge phase performs successive merge passes until no more than one string remains on each of the intermediate tapes. The distribution pattern for this phase involves the following operations:

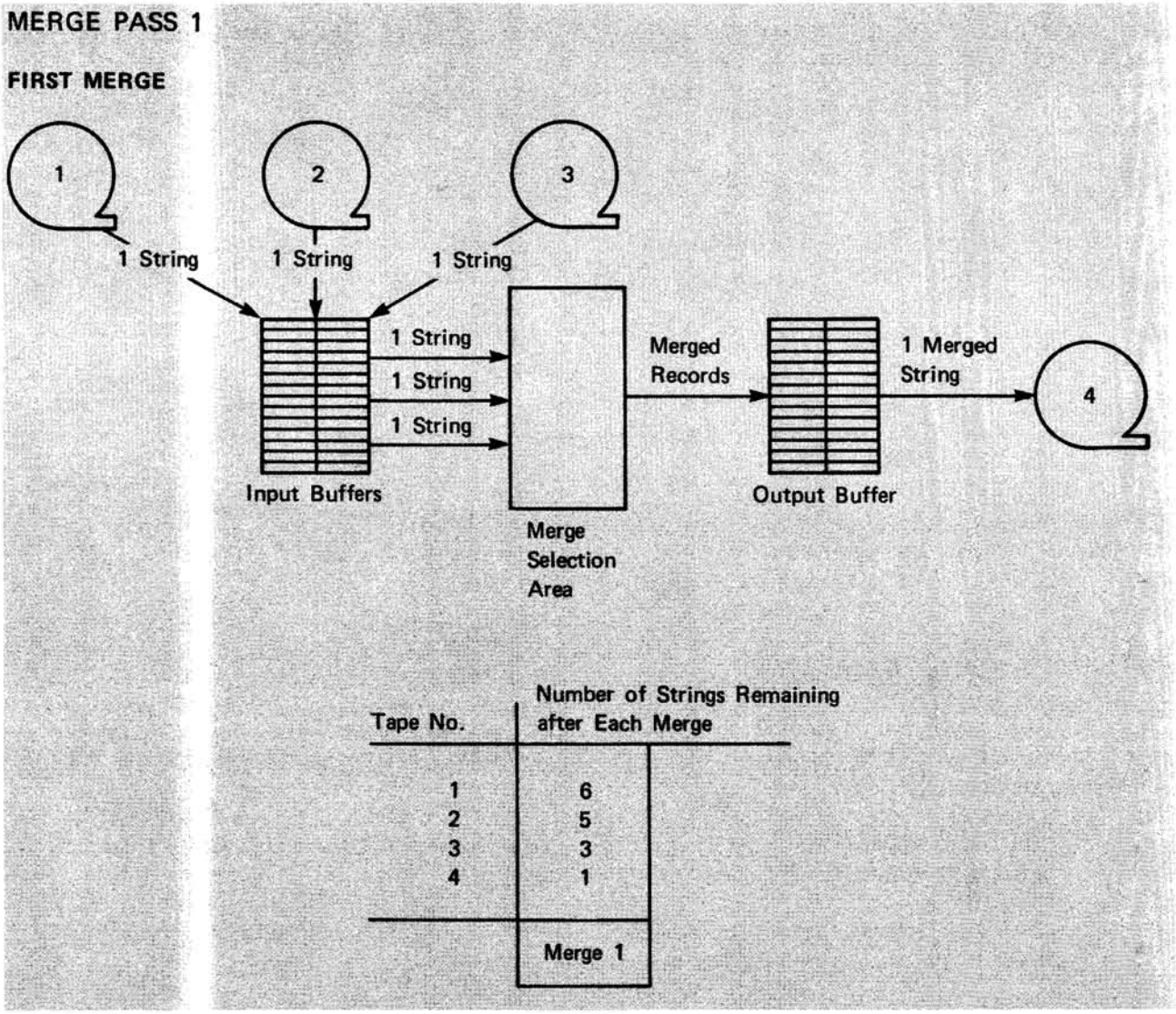
1. The last tape listed on the TAPE directive or macro is reserved as the output unit for the record string output from the first intermediate merge pass.

2. Each tape containing record strings merges onto the output tape a number of strings equal to the smallest number of strings on any of the tapes.

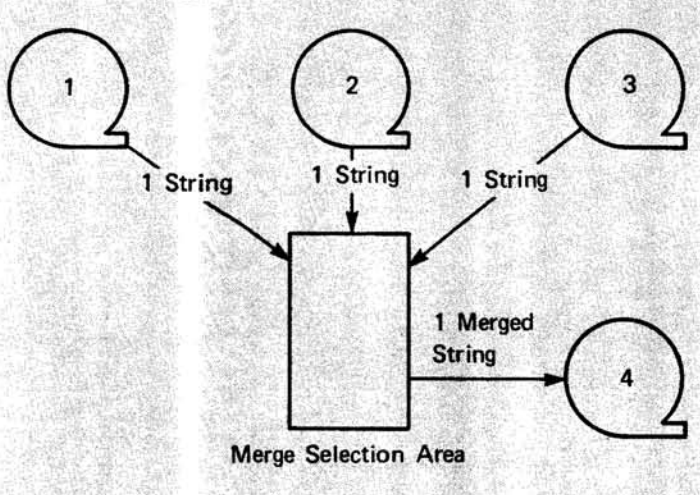
3. For the second merge pass and subsequent merge passes, the tape with no record strings at the completion of the previous merge pass is reassigned by the system as the output tape for that particular pass.

4. The number of strings distributed from each of the input tapes to the output tape during a particular merge pass is equal to the smallest number of strings remaining on one of the input tapes at the beginning of that merge pass.

The following example used the record string distribution from the internal sort example for a polyphase sort. Tape 4 is the output tape for the first merge pass. The number of record strings read off tapes 1, 2, and 3 for merging onto tape 4 equal the lowest number of record strings located on any one of the input tapes at the beginning of the merge pass (in this case, tape 3 with 4 record strings). Therefore, four record strings are read from each tape (a record string from each tape per merge) and merged into 4 record strings to be written on tape 4.

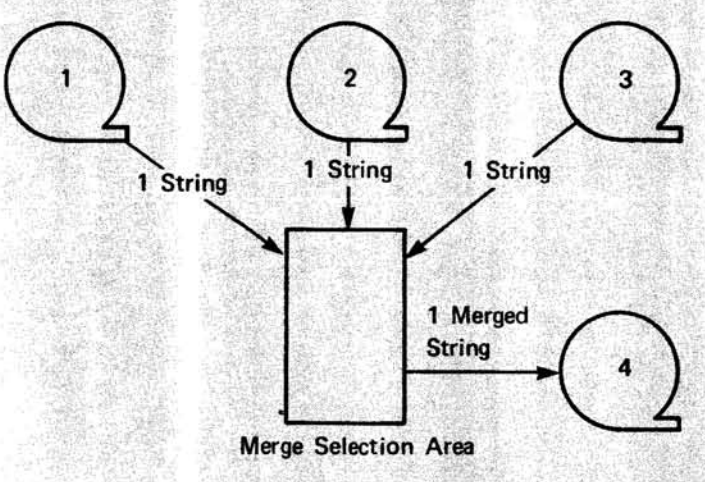

MERGE PASS 1
FIRST MERGE
1
2
3
1 String
1 String
1 String
1 String
1 String
1 String
Input Buffers
Merge
Selection
Area
Merged
Records
Output Buffer
1 Merged
String
4
Tape No.
Number of Strings Remaining
after Each Merge
1
2
3
4
6
5
3
1
Merge 1

**SECOND MERGE**



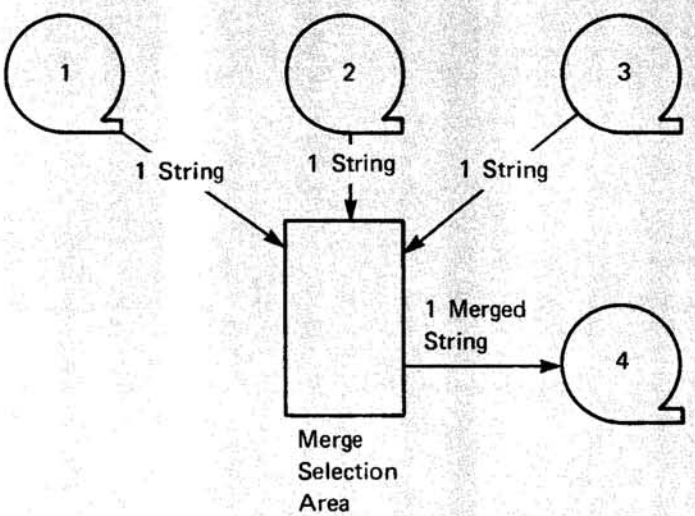

| Tape No. | Number of Strings Remaining after Each Merge | |
|---|---|---|
| 1 | 6 | 5 |
| 2 | 5 | 4 |
| 3 | 3 | 2 |
| 4 | 1 | 2 |
| | Merge 1 | Merge 2 |

**THIRD MERGE**




| Tape No. | Number of Strings Remaining after Each Merge | | |
|---|---|---|---|
| 1 | 6 | 5 | 4 |
| 2 | 5 | 4 | 3 |
| 3 | 3 | 2 | 1 |
| 4 | 1 | 2 | 3 |
| | Merge 1 | Merge 2 | Merge 3 |

**FOURTH MERGE**

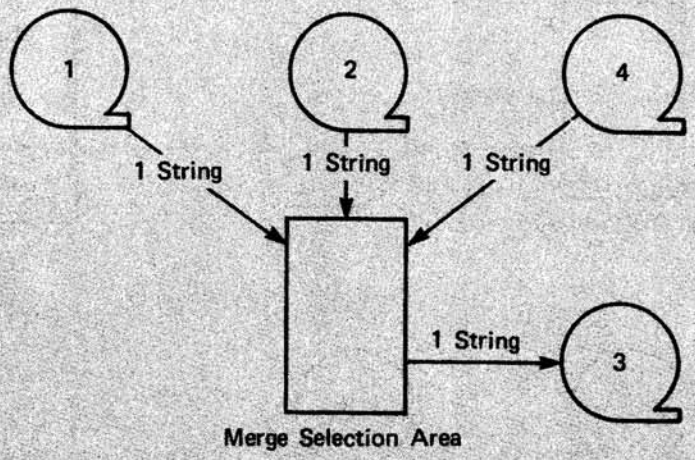



| Tape No. | Number of Strings Remaining after Each Merge | | | |
|---|---|---|---|---|
| 1 | 6 | 5 | 4 | 3 |
| 2 | 5 | 4 | 3 | 2 |
| 3 | 3 | 2 | 1 | 0 |
| 4 | 1 | 2 | 3 | 4 |
| | Merge Pass 1 | | | |

## MERGE PASS 2

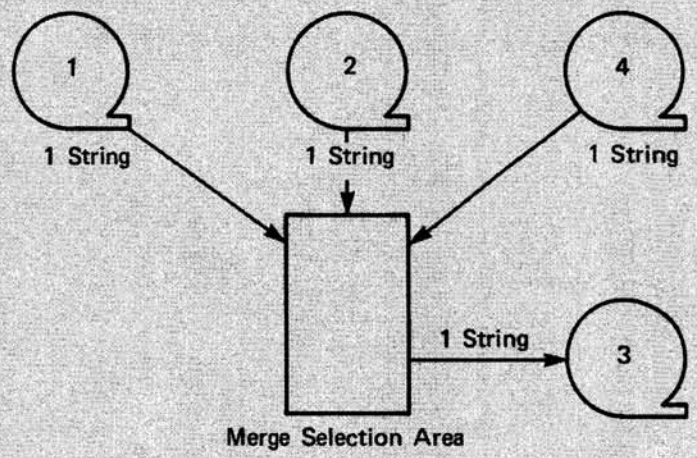
Tape 3 is assigned as the output tape. The two record strings located on tape 2 determine the number of record strings to be read from tapes 1, 2, and 4 during this merge pass.

**FIRST MERGE**




| Tape No. | Number of Strings Remaining after each Merge | | | | |
|---|---|---|---|---|---|
| 1 | 6 | 5 | 4 | 3 | 2 |
| 2 | 5 | 2 | 3 | 2 | 1 |
| 3 | 3 | 2 | 1 | 0 | 1 |
| 4 | 1 | 2 | 3 | 4 | 3 |
| | Merge Pass 1 | | | | Merge 1 |

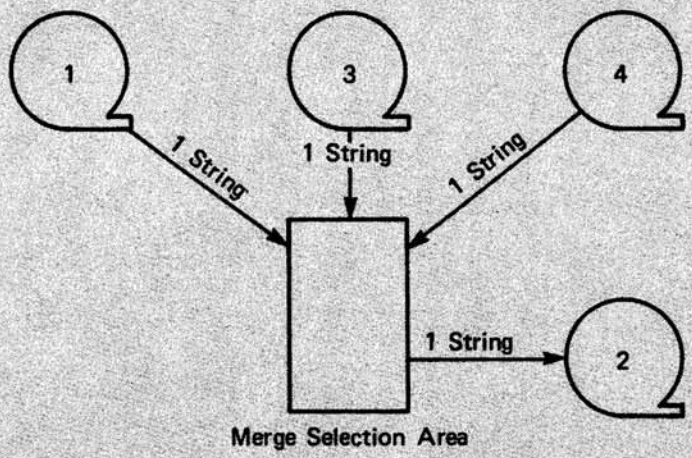
**SECOND MERGE**




| Tape No. | Number of Strings Remaining after each Merge | | | | | |
|---|---|---|---|---|---|---|
| 1 | 6 | 5 | 4 | 3 | 2 | 1 |
| 2 | 5 | 4 | 3 | 2 | 1 | 0 |
| 3 | 3 | 2 | 1 | 0 | 1 | 2 |
| 4 | 1 | 2 | 3 | 4 | 3 | 2 |
| | Merge Pass 1 | | | | Merge Pass 2 | |

## MERGE PASS 3

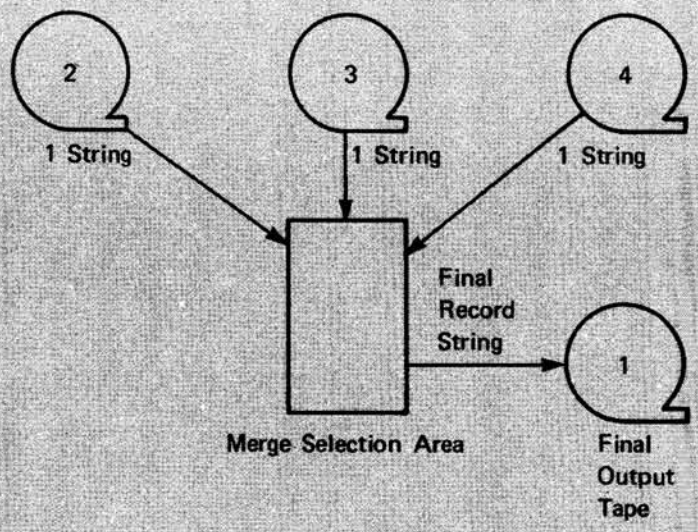
One record string is read from 1, 3, and 4, merged, and written on tape 2.




| Tape No. | Number of Strings Remaining after each Merge | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 2 | 5 | 4 | 3 | 2 | 1 | 0 | 1 |
| 3 | 3 | 2 | 1 | 0 | 1 | 2 | 1 |
| 4 | 1 | 2 | 3 | 4 | 3 | 2 | 1 |
| | Merge Pass 1 | | | | Merge Pass 2 | | Merge Pass 3 |

## FINAL MERGE PHASE

The single record strings remaining on tapes 2, 3, and 4 at the completion of the intermediate merge phase are merged into the final record string which is written onto the final output file specified by the user.




| Tape No. | Number of Strings Remaining after Final Merge |
|---|---|
| 1 | 1 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |

Final Output 1

# MERGE ORDER F

Merge order is applicable only to disk sorts under Sort/Merge Version 4. The merge order is a parameter used internally by Sort/Merge as the number of strings that can be merged in core at one time to form a longer sorted string; the merge order determines the number of merge buffers used during the intermediate merge phase. The user may specify merge order on the SORTMRG control statement (section 4), on the OPTIONS directive (section 4), or on the OPTIONS macro (section 5); or he may accept a default value calculated by Sort/Merge. Secondary users of Sort/Merge, such as users of COBOL sorts and QUERY UPDATE sorts, cannot specify merge order and are limited to the default value. Merge order is irrelevant for tape sorts, since the number of tapes provided for a balanced or polyphase sort implicitly establishes the merge order.

By specifying a merge order, it is possible to decrease the elapsed time of a sort or merge. The optimal value for merge order depends on machine configuration, quantity of sort input, input/output devices available, randomness of input data, and amount of core available for the sort. The merge order for an individual sort run should be chosen with all these factors in mind.

To understand how the merge order is used internally, it is helpful to know the procedure followed by Sort/Merge during the sort phase and the intermediate merge phase.

During the sort phase, Sort/Merge reads records from input files (or receives them from an Exit 1 owncode routine) and produces sorted strings by a tournament replacement technique. If all the records are read before the tournament is full, the sorted string is equivalent to the output file, so no intermediate strings need to be written. The size of the tournament and therefore the number of records that can be sorted in core, is dependent on the field length of the sort (as shown by Equation 1 below).

If, when the tournament is full, records still remain to be read, one or more intermediate string files must be written to make room for the new records in the tournament. Sort/Merge continues to read records and write them to string files as long as the records will be in proper order when they are written to the end of the current string. Any record which would not be in proper order is flagged and placed in the tournament. When the tournament is full, the current string is terminated and a new string begun.

After all records have been received and written to sorted strings, the intermediate merge phase begins. The number of strings that can be merged in core during this phase is dependent on the amount of central memory available for use as merge buffers (Equations 2 and 3 below). The number of buffers used is the merge order. If the number of sorted strings does not exceed the merge order, all the strings can be merged and written to the output file with no intermediate merge. If the number of sorted strings is greater than the merge order, several strings must be merged to form longer strings.

This procedure is repeated until the number of strings is less than or equal to the merge order, at which time the strings are merged onto the output file.

User merge order specification can improve or degrade Sort/Merge performance. If more buffers are available, fewer passes over the data are necessary, but input/output time might increase. If fewer buffers are available, input/output time might improve but more passes over the data might be necessary.

The default merge order has been chosen to yield optimal performance for most users. A different merge order could improve performance for a particular combination of system configuration, sort key, and input data. Changing any of these factors might make the user-selected merge order non-optimal. However, users who frequently execute very similar sorts of large data bases might wish to improve Sort/Merge performance by selecting and specifying a merge order. The following technical information is intended for use in the selection of an optimal merge order.

## MERGE ORDER FORMULAS

The number of records that can be sorted in core, N, is calculated as follows:

$$N = (LF - BFSIN)/(MLW + 2 + K)$$

The intermediate merge order, MO, is related to merge input buffer size as follows:

$$MO = (LI \times .42)/(MIBFS + MLW + K)$$

The number of passes over the data, P, is calculated as follows:

$$P = (\log NSTR / \log MO)$$

The final merge order, FMO, is calculated as follows:

$$FMO = (LF - BFSOUT - BFSM)/2003B$$

where

MIBFS = Merge input buffer size; MIBFS $\geq$ MAX (101B, MLW + 1)

NSTR = Number of strings generated in the sort phase. Can be estimated as follows:

NSTR = the greatest integer less than NREC / 2N for random data

= the greatest integer less than NREC / N for data in reverse order

= 1 for sorted data

MLW = Maximum record length in words of all input records.

K = Length of extracted keys in words, approximately 11/10 of the sum of all key lengths, if each key length is rounded up to a whole number of words. Add 1 word if the RETAIN option is selected.

BFSIN,BFSOUT = Buffer sizes in words for input and output files (see appendix D for Sort/Merge defaults).

BFSM = Sum of the merge input buffer sizes in words.

NREC = Number of records to be sorted

For directive sorts:

LF = FL - 13000B - length in words of owncode binaries

LI = FL - 6500B

For macro sorts:

LF = LI = FL - LWA - 1300B

where

FL = total current field length in words of job

LWA = address of last word of code loaded when Sort/Merge is called (contents of RA + 65B).

# GLOSSARY G

ADVANCED ACCESS METHODS (AAM) – A file manager that processes indexed sequential, direct access, and actual key file organizations, and supports the Multiple Index Processor. (See CYBER Record Manager.)

BALANCED TAPE SORT – Sort that always keeps its intermediate tapes divided into the same two groups. Sorted strings are merged from one group to another as long as possible, then the direction is reversed.

BASIC ACCESS METHODS (BAM) – A file manager that processes sequential and word addressable file organizations. (See CYBER Record Manager.)

BLOCKS – The term block has several meanings depending on context. On tape, a block is information between interrecord gaps on tape. Record Manager defines several blocks depending on organization:

| Organization | Blocks |
|---|---|
| Indexed sequential | Data block; index block |
| Direct access | Home block; overflow block |
| Actual key | Data block |
| Sequential | Block type I,C,K,E |

BOI (Beginning-of-Information) – Record Manager defines beginning-of-information as the start of the first user record in a file. System-supplied information, such as an index block or control word, does not affect beginning-of-information. Any label on a tape exists prior to beginning-of-information.

BUFFER – An intermediate storage area used to compensate for a difference in rates of data flow, or times of event occurrence, when transmitting data between central memory and an external device during input/output operations.

COLLATING SEQUENCE – Sequence that determines precedence given to character data for sorting, merging, and comparing.

CONTROL WORD – A system-supplied word that precedes each W type record in storage.

CYBER RECORD MANAGER – A generic term relating to the common products AAM and BAM that run under the NOS and NOS/BE operating systems and that allow a variety of record types, blocking types, and file organizations to be created and accessed. The execution time input/output of COBOL 4, COBOL 5, FORTRAN Extended 4, Sort/Merge 4, ALGOL 4, and the DMS-170 products is implemented through CYBER Record Manager. Neither the input/output of the NOS and NOS/BE operating systems themselves nor any of the system utilities such as COPY or SKIPF is implemented through CYBER Record Manager. All CYBER Record Manager file processing requests ultimately pass through the operating system input/output routines.

DIRECT ACCESS FILE – In the context of CYBER Record Manager, a direct access file is one of the five file organizations. It is characterized by the system hashing of the unique key within each file record to distribute records randomly in blocks called home blocks of the file.

In the context of NOS permanent files, a direct access file is a file that is accessed and modified directly, as contrasted with an indirect access permanent file.

DIRECTIVES – Instructions that supplement processing defined by the SORTMRG control statement for execution of Sort/Merge record processing.

EOI (End-of-Information) – Record Manager defines end-of-information in terms of the file organization and file residence.

| File Organization | File Residence | Physical Position |
|---|---|---|
| Sequential | Mass storage | After last user record. |
| | Labeled tape in SI,I, S,L format | After last user record and before any file trailer labels. |
| | Unlabeled tape in SI, I format | After last user record and before any file trailer labels. |
| | Unlabeled tape in S or L format | Undefined. |
| Word Addressable | Mass storage | After last word allocated to file, which might be beyond the last user record. |
| Indexed Sequential, Actual Key | Mass storage | After record with highest key value. |
| Direct Access | Mass storage | After last record in most recently created overflow block or home block with the highest relative address. |

FILE – A logically related set of information; the largest collection of information that can be addressed by a file name. Starts at beginning-of-information and ends at end-of-information.

FILE CONTROL STATEMENT – A control statement that contains parameters used to build the file information table for processing. Must be provided for every input or output file to be processed by a directive sort or merge.

FIT (File Information Table) – A table through which a user program communicates with Record Manager. All file processing executes on the basis of fields in the table. Some fields can be set by the Sort/Merge user in the FILE control statement.

HOME BLOCK – Mass storage allocated for a file with direct access organization at the time the file is created.

KEY COMPARISON – Internal technique of comparing sort keys that usually requires less elapsed time and more central processing time than key extraction.

KEY EXTRACTION – Internal technique of comparing sort keys that usually requires less central processing time and more elapsed time than key comparison.

LEVEL – For system-logical-records, an octal number 0 through 17 in the system-supplied 48-bit marker that terminates a short or zero-length PRU.

LOGICAL RECORD – Under NOS, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. Equivalent to a system-logical-record under NOS/BE.

MACRO – Sequence of source statements that are saved and then assembled when needed through a macro call. Used when Sort/Merge functions as a COMPASS subroutine for a COMPASS program or as a relocatable program generated for the COBOL SORT verb.

MERGE ORDER – Internal parameter governing the number of buffers used by Sort/Merge Version 4 in the intermediate merge phase.

NOISE RECORD – Number of characters the tape drivers discard as being extraneous noise rather than a valid record. Value depends on installation settings.

OVERFLOW BLOCK – Mass storage the system adds to a file with direct access organization when records cannot be accommodated in the home block.

OWNCODE ROUTINE – Closed COMPASS subroutine written by the user that provides the capability to insert, substitute, modify, or delete input and output records during Sort/Merge processing.

PARTITION – Record Manager defines a partition as a division within a file with sequential organization. Generally, a partition contains several records or sections. Implementation of a partition boundary is affected by file structure and residence.

| Device | RT | BT | Physical Boundary |
|---|---|---|---|
| PRU device | W | I | A short PRU of level 0 containing one-word deleted record pointing back to last I block boundary, followed by a control word with flag indicating partition boundary. |
| | W | C | A short PRU of level 0 containing a control word with a flag indicating partition boundary. |
| | D,F,R, T,U,Z | C | A short PRU of level 0 followed by a zero-length PRU of level 17. |
| S or L format tape | W | I | Separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary. |
| | W | C | Separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with a flag indicating a partition boundary. |
| | D,F,T, R,U,Z | C,K,E | Tapemark. |
| | S | — | Zero-length PRU of level number 0. |
| Any other tape format | | | Undefined. |

Notice that in a file with W type records, a short PRU of level 0 terminates both a section and a partition.

POLYPHASE TAPE SORT – Sort with only one intermediate output tape for each merge phase; however, the output tape is changed for each merge phase. A polyphase tape sort usually can sort more records than a balanced tape sort in the same amount of time and with the same number of intermediate tapes.

PRU – Under NOS and NOS/BE, the amount of information transmitted by a single physical operation of a specified device. The size of a PRU depends on the device (see table below). A PRU which is not full of user data is called a short PRU; a PRU that has a level terminator but no user data is called a zero-length PRU.

| Device | Size in Number of 60-Bit Words |
|---|---|
| Mass storage | 64 |
| Tape in SI format with coded data | 128 |
| Tape in SI format with binary data | 512 |
| Tape in I format | 512 |
| Tape in other format | Undefined |

PRU DEVICE – Under NOS and NOS/BE, a mass storage device or a tape in SI or I format, so called because records on these devices are written in PRUs.

RANDOM FILE – In the context of Record Manager, a file with word addressable, indexed sequential, direct access, or actual key organization in which individual records can be accessed by the values of their keys.

In the context of the NOS or NOS/BE operating systems, a file with the random bit set in the file information table in which individual records are accessed by their relative PRU numbers.

RECORD – Record Manager defines a record as a group of related characters. A record or a portion thereof is the smallest collection of information passed between Record Manager and a user program. Eight different record types exist, as defined by the RT field of the file information table.

Other parts of the operating systems and their products might have additional or different definition of records.

RECORD TYPE – The term record type can have one of several meanings, depending on the context. Record Manager defines eight record types established by an RT field in the file information table. Tables output by the loader are classified as record types such as text, relocatable, or absolute, depending on the first few words of the tables.

SECTION – Record Manager defines a section as a division within a file with sequential organization. Generally, a section contains more than one record and is a division within a partition of a file. A section terminates with a physical representation of a section boundary.

| Device | RT | BT | Physical Representation |
|---|---|---|---|
| PRU device | W | I | Deleted one-word record pointing back to last I block boundary followed by a control word with flags indicating a section boundary. At least the control word is in a short PRU of level 0. |
| | W | C | Control word with flags indicating a section boundary. The control word is in a short PRU of level 0. |
| | D,F,R, T,U,Z | C | Short PRU with level less than 17 octal. |
| | D,F,R, T,U,Z | K | Undefined. |
| | S | Any | Undefined. |
| S or L format tape | W | I | A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size followed by a deleted one-word record pointing back to last I block boundary followed by a control word with flags indicating a section boundary. |
| | W | C | A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size followed by a control word with flags indicating a section boundary. |
| | D,F,R, T,U,Z | C,K,E | Undefined. |
| | S | Any | Undefined. |
| Any other tape format | | | Undefined. |

The NOS and NOS/BE operating systems equate a section with a system-logical-record of level 0 through 16 octal.

SHORT PRU – A PRU that does not contain as much user data as the PRU can hold and that is terminated by a system terminator with a level number.

Under NOS, a short PRU defines EOR.

Under NOS/BE, a short PRU defines the end of a system-logical-record. In the BAM context, a short PRU can have several interpretations depending on the record and blocking types.

SIGNED NUMERIC DATA – Integer data stored internally in display code. Sorts according to the magnitude and the value of the integer the display code represents.

SORT KEY – Field of information within each record in a sort or merge input file used to determine the order in which records are written to the output file.

SORT ORDER – Order for sorting keys, either ascending or descending.

SYSTEM–LOGICAL–RECORD – Under NOS/BE, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. These records can be transferred between devices without loss of structure.

Under SCOPE 2, a data grouping that consists of one or more blocks terminated by a short block.

Equivalent to a logical record under NOS.

Equivalent to a Record Manager S type record.

TAPE SORT – Sort that has its intermediate scratch files residing on tape rather than disk. Original input file and/or final output file can reside on disk or tape.

W TYPE RECORD – One of the eight record types supported by Record Manager. Such records appear in storage preceded by a system supplied control word. The existence of the control word allows files with sequential organization to have both partition and section boundaries.

ZERO–BYTE TERMINATOR – 12 bits of zero in the low order position of a word that marks the end of the line to be displayed at a terminal or printed on a line printer. The image of cards input through the card reader or terminal also has such a terminator.

ZERO–LENGTH PRU – A PRU that contains system information, but no user data. Under BAM, a zero-length PRU of level 17 is a partition boundary. Under NOS, a zero-length PRU defines EOF.

# FUTURE SYSTEM MIGRATION GUIDELINES H

This appendix contains programming practices recommended by CDC for users of the software described in this manual. When possible, application programs based on this software should be designed and coded in conformance with these recommendations.

Two forms of guidelines are given. The general guidelines minimize application program dependence on the specific characteristics of a hardware system. The feature use guidelines ensure the easiest migration of an application program to future hardware or software systems.

## GENERAL GUIDELINES

Programmers should observe the following practices to avoid hardware dependency:

- Avoid programming with hardcoded constants. Manipulation of data should never depend on the occurrence of a type of data in a fixed multiple such as 6, 10, or 60.
- Do not manipulate data based on the binary representation of that data. Characters should be manipulated as characters, rather than as octal display-coded values or as 6-bit binary digits. Numbers should be manipulated as numeric data of a known type, rather than as binary patterns within a central memory word.
- Do not identify or classify information based on the location of a specific value within a specific set of central memory word bits.
- Avoid using COMPASS in application programs. COMPASS and other machine-dependent languages can complicate migration to future hardware or software systems. Migration is restricted by continued use of COMPASS for stand-alone programs, by COMPASS subroutines embedded in programs using higher-level languages, and by COMPASS owncode routines used with CDC standard products. COMPASS should only be used to create part or all of an application program when the function cannot be performed in a higher-level language or when execution efficiency is more important than any other consideration.

## FEATURE USE GUIDELINES

The recommendations in the remainder of this appendix ensure the easiest migration of an application program for use in future hardware or software systems. These recommendations are based on known or anticipated changes in the hardware or software system, or comply with proposed new industry standards or proposed changes to existing industry standards.

### BASIC ACCESS METHODS

The Basic Access Methods (BAM) offer several features within which choices must be made. The following paragraphs indicate preferred usage.

#### File Organizations

The recommended file organization is sequential (SQ). For files with word-addressable (WA) organization, use an accessing technique that can easily be modified to character position or byte addresses.

### Block Types

The recommended block type is C.

### Record Types

The recommended record types are F for fixed length records and W for variable length records. For purely coded files that are to be listed, Z type records can be used.

### Block Size

For C type blocks, set the maximum block length (MBL) to 640 characters for mass storage files and 5120 characters for tape files.

### Host Language Input/Output

Use of host language input/output statements (for example, a FORTRAN READ statement) to process BAM files is always a safe procedure. Host language statements provide appropriate default values for record type, block type, and block size. Do not use the CYBER Record Manager FORTRAN interface routines to process sequential files.

## SORT/MERGE 4 AND 1

Sort/Merge offers several features among which choices must be made. The following paragraphs indicate preferred usage.

### Key Alignment

Ensure that SORT keys are aligned on character or word boundaries. Do not place SORT keys in arbitrary bit positions within words.

### SORT and MERGE Statements

Always perform logically separated SORT and MERGE operations with separate control statements.

# INDEX

# COMMENT SHEET

**MANUAL TITLE:** Sort/Merge Versions 4 and 1 Reference Manual

**PUBLICATION NO.:** 60497500 **REVISION:** F

**NAME:** _____

**COMPANY:** _____

**STREET ADDRESS:** _____

**CITY:** _____ **STATE:** _____ **ZIP CODE:** _____

**This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).**

☐ Please reply ☐ No reply necessary

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

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

FOLD ON DOTTED LINES AND TAPE

APE

TAPE

OLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

**CONTROL DATA CORPORATION**

**Publications and Graphics Division**

215 Moffett Park Drive
Sunnyvale, California 94086

CUT ALONG LINE

OLD

FOLD