

# **TOPS-10 SORT/MERGE User's Guide**

AA-M063A-TB

**February 1982**

This document describes user procedures for the SORT/MERGE stand-alone utility program.

This document supersedes the SORT/MERGE User's Guide, Order No. AA-0997D-TB, published May 1978.

**OPERATING SYSTEM:** TOPS-10 V7.01

**SOFTWARE:** SORT/MERGE V04C

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

**Northeast/Mid-Atlantic Region**

Digital Equipment Corporation  
PO Box CS2008  
Nashua, New Hampshire 03061  
Telephone:(603)884-6660

**Central Region**

Digital Equipment Corporation  
Accessories and Supplies Center  
1050 East Remington Road  
Schaumburg, Illinois 60195  
Telephone:(312)640-5612

**Western Region**

Digital Equipment Corporation  
Accessories and Supplies Center  
632 Caribbean Drive  
Sunnyvale, California 94086  
Telephone:(408)734-4915

**First Printing, June 1977**  
**Revised, May 1978**  
**Revised, February 1982**


Copyright ©, 1977, 1978, 1982, Digital Equipment Corporation. All Rights Reserved.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

DEC	DECnet	IAS
DECUS	DECsystem-10	MASSBUS
DECSYSTEM-20	PDT	PDP
DECwriter	RSTS	UNIBUS
DIBOL	RSX	VAX
EduSystem	VMS	VT
	RT	

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

# Contents

## Preface

## Chapter 1 Getting Started with SORT/MERGE

1.1	Creating a Multifield ASCII Test File . . . . .	1-1
1.2	Determining Record Length, Key Position, and Key Length . . . . .	1-2
1.3	Sorting a Multifield File . . . . .	1-3
1.3.1	Sorting the File on the Name Field. . . . .	1-3
1.3.2	Sorting the File on the Age Field. . . . .	1-5
1.3.3	Sorting the File on the Grade Field . . . . .	1-5
1.3.4	Sorting the File on Two Keys . . . . .	1-6
1.4	Sorting a Line-Sequenced ASCII File . . . . .	1-7
1.5	Sorting Files Containing Tabs . . . . .	1-8
1.6	Sorting an ASCII File Containing Variable-Length Records . . . . .	1-10
1.7	Sorting a Multiline ASCII File . . . . .	1-11
1.8	Sorting Uppercase and Lowercase Text. . . . .	1-12
1.9	Sorting Nontext Files . . . . .	1-13

## Chapter 2 How to Use SORT/MERGE

2.1	Introduction . . . . .	2-1
2.2	Running SORT/MERGE . . . . .	2-2
2.3	Command Formats. . . . .	2-4
2.3.1	The SORT/MERGE Command Scanner . . . . .	2-5
2.3.2	Merging Files . . . . .	2-7
2.3.3	Using Command Files . . . . .	2-7
2.4	Using SORT/MERGE from a COBOL Program . . . . .	2-8
2.5	Using SORT/MERGE from a FORTRAN Program . . . . .	2-8

## Chapter 3 SORT/MERGE Switches

3.1	Required Switches . . . . .	3-2
3.1.1	/RECORD-SIZE Switch . . . . .	3-2
3.1.2	/KEY Switch. . . . .	3-2
3.1.2.1	/KEY Switch Format . . . . .	3-3
3.1.2.2	Key Starting Position . . . . .	3-4
3.1.2.3	Key Length . . . . .	3-5
3.1.2.4	Key Collating Order . . . . .	3-6
3.1.2.5	Key Data Type. . . . .	3-6
3.1.2.6	Key Sign Status . . . . .	3-8
3.2	Recording Mode Switches . . . . .	3-10
3.2.1	/ASCII Switch . . . . .	3-10
3.2.2	/SIXBIT Switch . . . . .	3-10
3.2.3	/EBCDIC Switch. . . . .	3-11
3.2.4	/BINARY Switch. . . . .	3-11
3.3	File Switches . . . . .	3-11
3.3.1	/AFTER-ADVANCING Switch. . . . .	3-11
3.3.2	/ALIGN Switch . . . . .	3-12
3.3.3	/BEFORE-ADVANCING Switch . . . . .	3-12
3.3.4	/BLOCKED Switch. . . . .	3-12
3.3.5	/FIXED Switch. . . . .	3-14
3.3.6	/FORTRAN Switch. . . . .	3-15
3.3.7	/NOCRLF Switch . . . . .	3-15
3.3.8	/RANDOM Switch . . . . .	3-15
3.3.9	/SEQUENTIAL Switch. . . . .	3-16
3.3.10	/VARIABLE Switch . . . . .	3-16
3.4	Control Switches. . . . .	3-16
3.4.1	/BUFFER-PAGES Switch . . . . .	3-16
3.4.2	/CHECK Switch . . . . .	3-17
3.4.3	/COLLATE Switch . . . . .	3-17
3.4.4	/ERROR-RETURN Switch . . . . .	3-19
3.4.5	/FATAL-ERROR-CODE Switch . . . . .	3-20
3.4.6	/LEAVES Switch. . . . .	3-21
3.4.7	/MAX-TEMP-FILES Switch . . . . .	3-21
3.4.8	/PHYSICAL Switch . . . . .	3-21
3.4.9	/STATISTICS Switch. . . . .	3-22
3.4.10	/SUPPRESS-ERROR Switch . . . . .	3-22
3.4.11	/TEMPORARY-AREA Switch . . . . .	3-23
3.5	Tape Switches . . . . .	3-23
3.5.1	/ANSI-ASCII Switch. . . . .	3-23
3.5.2	/DENSITY Switch . . . . .	3-23
3.5.3	/INDUSTRY-COMPATIBLE Switch . . . . .	3-24
3.5.4	/LABEL Switch . . . . .	3-24
3.5.5	/PARITY Switch . . . . .	3-25
3.5.6	/POSITION Switch. . . . .	3-25
3.5.7	/REWIND Switch . . . . .	3-26
3.5.8	/UNLOAD Switch . . . . .	3-26

## Chapter 4 File Formats

4.1	Recording Modes . . . . .	4-1
4.1.1	ASCII Recording Mode . . . . .	4-2
4.1.2	SIXBIT Recording Mode . . . . .	4-2
4.1.3	EBCDIC Recording Mode . . . . .	4-2
4.1.4	BINARY Recording Mode . . . . .	4-3
4.2	File Formats . . . . .	4-3
4.2.1	Fixed-Length ASCII . . . . .	4-5
4.2.1.1	COBOL Fixed-Length ASCII . . . . .	4-5
4.2.1.2	FORTTRAN Fixed-Length (Random) ASCII . . . . .	4-6
4.2.2	Variable-Length ASCII . . . . .	4-8
4.2.2.1	COBOL Variable-Length ASCII . . . . .	4-9
4.2.2.2	FORTTRAN Variable-Length (Sequential) ASCII . . . . .	4-10
4.3	Fixed-Length SIXBIT . . . . .	4-12
4.3.1	COBOL Fixed-Length SIXBIT . . . . .	4-13
4.4	Variable-Length SIXBIT . . . . .	4-14
4.4.1	COBOL Variable-Length SIXBIT . . . . .	4-15
4.5	EBCDIC File Formats . . . . .	4-16
4.5.1	COBOL Fixed-Length EBCDIC . . . . .	4-17
4.5.2	COBOL Variable-Length EBCDIC . . . . .	4-18
4.5.3	COBOL Blocked Fixed-Length EBCDIC . . . . .	4-20
4.5.4	COBOL Blocked Variable-Length EBCDIC . . . . .	4-21
4.6	BINARY File Formats . . . . .	4-25
4.6.1	COBOL Binary File Formats . . . . .	4-25
4.6.1.1	COBOL ASCII Mixed-Mode Binary . . . . .	4-25
4.6.1.2	COBOL SIXBIT Mixed-Mode Binary . . . . .	4-27
4.6.1.3	COBOL EBCDIC Mixed-Mode Binary . . . . .	4-28
4.6.2	FORTTRAN Binary File Formats . . . . .	4-30
4.6.2.1	FORTTRAN Random Binary (with LSCWs) . . . . .	4-31
4.6.2.2	FORTTRAN Random Binary (without LSCWs) . . . . .	4-33
4.6.2.3	FORTTRAN Sequential Binary (with LSCWs) . . . . .	4-35
4.6.2.4	FORTTRAN Sequential Binary (without LSCWs) . . . . .	4-37

## Chapter 5 SORT/MERGE Error Messages

5.1	Message Format . . . . .	5-1
5.2	Error Messages . . . . .	5-1

## Chapter 6 SORT/MERGE Performance Considerations

6.1	Performance Overview . . . . .	6-1
6.1.1	The Sort Phase . . . . .	6-2
6.1.2	The Merge Phase . . . . .	6-3

6.2	Performance Considerations . . . . .	6-9
6.2.1	Tree Size . . . . .	6-9
6.2.2	Number of I/O Buffers . . . . .	6-10
6.2.3	Number of Merge Passes . . . . .	6-11

## Appendix A Summary of SORT/MERGE Commands and Switches

A.1	Commands. . . . .	A-1
A.2	Switches (in Alphabetic Order). . . . .	A-2
A.3	Switches (by Function). . . . .	A-10
A.4	Switches (by Type). . . . .	A-11

## Appendix B Collating Sequences and Conversion Tables

### Glossary

### Index

### Figures

3-1	Key Argument Tree . . . . .	3-4
4-1	COBOL Fixed-Length ASCII . . . . .	4-6
4-2	FORTRAN Fixed-Length (Random) ASCII . . . . .	4-7
4-3	COBOL Variable-Length ASCII . . . . .	4-9
4-4	FORTRAN Variable-Length (Sequential) ASCII . . . . .	4-11
4-5	COBOL Fixed-Length SIXBIT . . . . .	4-13
4-6	COBOL Variable-Length SIXBIT . . . . .	4-16
4-7	COBOL Fixed-Length EBCDIC . . . . .	4-17
4-8	COBOL Variable-Length EBCDIC . . . . .	4-19
4-9	COBOL Blocked Fixed-Length EBCDIC . . . . .	4-21
4-10	COBOL Blocked Variable-Length EBCDIC . . . . .	4-24
4-11	COBOL Standard Binary and ASCII Mixed-Mode Binary . . . . .	4-26
4-12	COBOL Standard Binary and SIXBIT Mixed-Mode Binary . . . . .	4-27
4-13	COBOL Standard Binary and EBCDIC Mixed-Mode Binary . . . . .	4-29
4-14	FORTRAN Standard and Mixed-Mode Random Binary with LSCWs . . . . .	4-32
4-15	FORTRAN Standard and Mixed-Mode Random Binary . . . . .	4-34
4-16	FORTRAN Standard/Mixed-Mode Sequential Binary with LSCWs . . . . .	4-36
4-17	FORTRAN Standard and Mixed-Mode Sequential Binary . . . . .	4-38
6-1	Binary Tree . . . . .	6-2
6-2	The Operation of the SORT/MERGE Binary Tree. . . . .	6-4

### Tables

3-1	Field Descriptors for COBOL. . . . .	3-7
3-2	Field Descriptors for FORTRAN . . . . .	3-8
B-1	ASCII and SIXBIT Collating Sequence and Conversion to EBCDIC . . . . .	B-2
B-2	ASCII to SIXBIT Conversion. . . . .	B-4
B-3	EBCDIC Collating Sequence and Conversion to ASCII . . . . .	B-6

## Preface

This manual is written for those who want to sort data using the SORT/MERGE utility program. The intended audience is composed of programmers and/or persons having user experience with a computer system.

As a user's guide, this manual is a combination of reference and explanatory material that should be sufficient for anyone with a reasonable amount of experience to successfully sort any of the file formats supported by SORT/MERGE. This experience should include some programming experience in a high-level language or simply user experience with a computer system. Specifically, you should be knowledgeable of the various internal representations of data and with the file format of the data you intend to sort.

However, if you are a beginning user who has never used a sorting utility, Chapter 1 has been written to help you get started with the SORT/MERGE utility. This chapter is devoted to sorting ASCII text files, and includes some transitional material as a bridge into more complex types of sorts.

Where applicable, shaded areas in the examples show changes in the SORT/MERGE command strings.

If there is any term used in this manual that you are unfamiliar with, refer to the Glossary at the end of this manual for a definition of the term.

The software required to run SORT/MERGE is MACRO version 53A or later and LINK version 4A or later. This manual reflects version 4C of the SORT/MERGE software.

References are made in this manual to the latest editions of the following documents:

- *TOPS-10/TOPS-20 COBOL-68 Language Reference Manual*
- *TOPS-10/TOPS-20 COBOL-74 Language Reference Manual*
- *TOPS-10 Monitor Calls Manual*
- *TOPS-10/TOPS-20 FORTRAN Language Manual*
- *TOPS-10 Operating System Commands Manual*



# Chapter 1

## Getting Started with SORT/MERGE

This chapter is written for the beginning SORT/MERGE user who has a simple ASCII or line-sequenced ASCII text file to sort. It demonstrates how to create an ASCII file using SOS, how to calculate the position and length of the key field, and how to use SORT/MERGE to sort the file.

### 1.1 Creating a Multifield ASCII Test File

To create and use a practice file, log in and create the sample file below. (The examples use the SOS editor.) The file is a student record file consisting of last name, age, and final grade. It is 17 characters long and has the following format:

name field — position 1 to 10  
 age field — positions 14 and 15  
 grade field — position 17

The following illustrates the format of each entry in the file:

0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7
N	N	N	N	N	N	N	N	N	N					A	A	G

name field
age field
grade field

MR-S-1712-81

The following example illustrates creating the file with SOS. Each field of the record is separated from the other fields by blanks (not tabs).

```
.SOS TEST,IN(RET)
Input: TEST,IN
00100 12345678901234567(RET)
00200 SMITH 19 C(RET)
00300 JONES 32 A(RET)
00400 ABERNATHY 19 B(RET)
00500 BROWNING 20 C(RET)
00600 HART 19 A(RET)
00700 WILSON 20 B(RET)
00800 HILL 20 A(RET)
00900 $
      ↑
      (ESC)

*D100(RET)
1 Lines (00100/1) deleted
*ES(RET)

[TEST,IN]
.
```

### NOTE

Line 00100 is temporarily used as a line of numbers to help format the file.

Before the file is saved, the line of formatting numbers is deleted (line 00100). Also, the ES command, in SOS, is used to save the file without line numbers. The saved version of the file looks like this:

```
.TYPE TEST,IN(RET)
SMITH 19 C
JONES 32 A
ABERNATHY 19 B
BROWNING 20 C
HART 19 A
WILSON 20 B
HILL 20 A
```

## 1.2 Determining Record Length, Key Position, and Key Length

Now that the file is created, you must determine the following:

1. The record length.
2. Whether the records are fixed-length or variable-length.
3. The starting position and length of the key field. (The key field is that part of the record you want to sort on.)

Each line in the sample file is 17 characters long. Because each line is a separate record, the record length is 17 characters long also. You do not count line terminators, such as RETURN, as part of the record length. Because each record is exactly 17 characters long, the records are fixed-length.

The starting position and length of the key field depends on which field of the record you wish to sort (name, age, or grade). If you want to sort on the name field, the key starts at position 1 and is 10 characters long:

0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
N	N	N	N	N	N	N	N	N	N				A	A	G

key field

MR-S-1713-81

If you desire to sort on the age field, the key starts at character position 14 and is 2 characters long:

0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
N	N	N	N	N	N	N	N	N					A	A	G

key field

MR-S-1714-81

If you wish to sort on the grade field, the key starts at character position 17 and is 1 character long:

0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
N	N	N	N	N	N	N	N	N						A	G

key field

MR-S-1715-81

## 1.3 Sorting a Multifield File

### 1.3.1 Sorting the File on the Name Field

Run SORT/MERGE by giving the command R SORT at monitor level:

```
. R SORT(RET)
*
```

#### NOTE

Those areas of the command string that change in the following examples are shaded.

When SORT/MERGE responds with the prompt '\*', type the output file specification:

```
*TEST.OUT
```

Type an equals sign (=):

```
*TEST.OUT=
```

Type the input file specification:

```
*TEST.OUT=TEST.IN
```

Type the /FIXED switch to indicate that you are sorting fixed-length records:

```
*TEST.OUT=TEST.IN/FIXED
```

Type the /RECORD switch and give the record length of 17:

```
*TEST.OUT=TEST.IN/FIXED/RECORD:17
```

Type the /KEY switch, then type the starting position of the key as 1, a colon (:), and the length of the key as 10:

```
*TEST.OUT=TEST.IN/FIXED/RECORD:17/KEY:1:10
```

Type a colon and indicate if the file is to be sorted in ASCENDING or DESCENDING order:

```
*TEST.OUT=TEST.IN/FIXED/RECORD:17/KEY:1:10:ASCENDING(RET)
```

Type a carriage return (RET) and wait for SORT/MERGE to perform the sort. When the sort is complete, SORT/MERGE responds with the following display:

```
[SRTXPN Expanding to 46P]
Sorted 7 records
12 KEY comparisons, 1.71 per record
40 record leaves in memory
0 runs
0:00:00 CPU time, 16.43 MS per record
0:00:42 elapsed time
```

The sorted file, TEST.OUT appears as:

```
.TYPE TEST.OUT(RET)
```

```
ABERNATHY    19 B
BROWNING     20 C
HART         19 A
HILL         20 A
JONES        32 A
SMITH        19 C
WILSON       20 B
```

To sort the file in descending order, enter the following command string to SORT/MERGE:

```
*TEST,OUT=TEST,IN/FIXED/RECORD:17/KEY:1:10:DESCENDING(RET)
```

The sorted file appears as:

```
.TYPE TEST,OUT(RET)
```

WILSON	20	B
SMITH	19	C
JONES	32	A
HILL	20	A
HART	19	A
BROWNING	20	C
ABERNATHY	19	B

### 1.3.2 Sorting the File on the Age Field

To sort the file on the age field, simply change the /KEY: arguments in the command string to indicate that field, for example,

```
*TEST,OUT=TEST,IN/FIXED/RECORD:17/KEY:14:2:ASCENDING(RET)
```

The sorted file appears as:

```
.TYPE TEST,OUT(RET)
```

SMITH	19	C
ABERNATHY	19	B
HART	19	A
BROWNING	20	C
WILSON	20	B
HILL	20	A
JONES	32	A

### 1.3.3 Sorting the File on the Grade Field

To sort the file on the grade field, simply change the /KEY: arguments in the command string to indicate that field, for example,

```
*TEST,OUT=TEST,IN/FIXED/RECORD:17/KEY:17:1:ASCENDING(RET)
```

The sorted file appears as:

```
.TYPE TEST,OUT(RET)
```

JONES	32	A
HART	19	A
HILL	20	A
ABERNATHY	19	B
WILSON	20	B
SMITH	19	C
BROWNING	20	C

### 1.3.4 Sorting the File on Two Keys

The previous examples demonstrated how to sort the file on any one of the fields in the record by changing the /KEY: switch arguments. It is often necessary to sort a file on more than one field. For example, you might want to sort your test file by age and also have each of the age groups sorted by grade. To do this, type the following command string to SORT/MERGE:

```
*TEST,OUT=TEST,IN/FIXED/RECORD:17/KEY:14:2:ASCENDING (RET)
#/KEY:17:1:ASCENDING(RET)
```

#### NOTE

To continue a command string line, you must type a dash (–) followed by a carriage–return/line–feed. SORT/MERGE then prompts you with a pound–sign (#). You can then continue typing the command string as shown above.

The sorted file appears as:

```
.TYPE TEST,OUT(RET)
```

```
HART          19 A
ABERNATHY     19 B
SMITH         19 C
HILL          20 A
WILSON        20 B
BROWNING      20 C
JONES         32 A
```

## 1.4 Sorting a Line–Sequenced ASCII File

In the first example, you saved TEST.IN without line numbers. The internal octal representation of the first record of TEST.IN would appear as:

Internal Representation →	123	115	111	124	110	0	← Bit 35
Data →	"S"	"M"	"I"	"T"	"H"		
	040	040	040	040	040	0	
	" "	" "	" "	" "	" "		
	040	040	040	061	071	0	
	" "	" "	" "	"1"	"9"		
	040	103	015	012	000	0	
	" "	"C"	CR	LF	NUL		

MR-S-1716-81

If the file is saved with line numbers, an additional six characters are added to each record. These characters are the five ASCII digits of the line number plus the horizontal tab. If TEST.IN has record-sequence numbers, the octal representation of its first record appears as:

060 "0"	060 "0"	061 "1"	060 "0"	060 "0"	1
011 HT	123 "S"	115 "M"	111 "I"	124 "T"	0
110 "H"	040 " "	040 " "	040 " "	040 " "	0
040 " "	040 " "	040 " "	040 " "	061 "1"	0
071 "9"	040 " "	103 "C"	015 CR	012 LF	0

MR-S-1717-81

### NOTE

Bit 35 is turned on in a line-sequence word. Thus, the line-sequence sequence word shown above is not the same as a data word containing the ASCII characters '00100'.

The addition of the six characters increases the record length by six and shifts the key starting position six places to the right. To sort a line-sequenced version of TEST.IN on the name field, you must give the following command:

```
*TEST.OUT=TEST.IN/FIXE/RECORD:23/KEY:7:10:ASCENDING(RET)
```

Thus, the /RECORD: switch argument and the /KEY: switch argument were increased to include the length of the line sequence number.

The sorted file appears as:

```
,TYPE TEST.OUT(RET)
00300  ABERNATHY  19 B
00400  BROWNING  20 C
00500  HART      19 A
00700  HILL      20 A
00200  JONES     32 A
00100  SMITH     19 C
00600  WILSON    20 B
```

Note that the line-sequence numbers are now out of order. To correct this, you can edit the file (with SOS) and renumber it with the N command. However, as soon as you open the file, SOS generates a warning message for each line that is out of sequence. If you are working with a large file,

you can have a tedious wait (while the warning messages are displayed) before you can renumber the file. For this reason, it is recommended that you sort ASCII files that are not line-sequenced and defer adding line-sequence numbers until after the file is sorted.

## 1.5 Sorting Files Containing Tabs

Tab characters are peculiar in that they are never printed or displayed on an output device. Rather, they constitute an instruction to the output device (or the monitor if TTY NO TAB is set) to skip a certain number of spaces before printing the next character. Tab characters can have the following problems for you when using SORT/MERGE:

1. While a single tab character can introduce as many as seven blank spaces into a printed line, those seven "blanks" are not stored internally as a group of ASCII blank characters. (An ASCII blank is represented by octal 040, but a single ASCII tab character is represented by octal 011.) If you mistakenly interpret a single tab character to be the number of spaces it produces on the output device, then you are calculating an incorrect record length and an erroneous starting position for the key field. If the tab character falls within the key field, then you are calculating an erroneous length for the key field.
2. Because tab characters are not printed, they are impossible to detect on an ordinary printout or terminal display.
3. Tabs that are used randomly make it impossible to calculate a record length or key field starting position that is accurate for each record in the file. If tabs are used in the key fields of some records, but blanks are used in the key fields of other records, then you are unable to sort the file correctly.

For example, the following lines appear identical, but the first line contains blanks, while the second line contains tabs.

SMITH	24	C
SMITH	24	C



The octal representation of the first line appears as:

123	115	111	124	110
"S"	"M"	"I"	"T"	"H"
040	040	040	040	040
" "	" "	" "	" "	" "
040	040	040	040	040
" "	" "	" "	" "	" "
040	062	064	040	040
" "	"2"	"4"	" "	" "
040	040	040	040	103
" "	" "	" "	" "	"C"
015	012	000	000	000
CR	LF	NUL	NUL	NUL

MR-S-1718-81

The octal representation of the second line appears as:

123	115	111	124	110
"S"	"M"	"I"	"T"	"H"
011	011	062	064	011
HT	HT	2	4	HT
103	015	012	000	000
"C"	CR	LF	NUL	NUL

MR-S-1719-81

Although the two records (lines) shown above appear to be identical when printed, their internal octal representation is very different. The record length of the line using blanks is 25, but the record length of the line using tabs is only 11. While the name field occurs in the same place in both records, the age field and grade field do not.

The solution to the problem of tab characters is to avoid using them altogether or to use them only in a consistent manner. However, sometimes you cannot know whether the file contains tab characters. A solution to this problem is to use the PIP program to convert any tabs that may exist in your file to the correct number of spaces that each tab represents. The following example illustrates the commands you can give to PIP.

```
.R PIP(RET)
*NOTAB,FIL=TEST,IN/W(RET)
```

The /W switch causes PIP to convert the tabs in the file to the appropriate number of spaces.

## 1.6 Sorting an ASCII File Containing Variable–Length Records

The previous examples concentrated on files containing fixed–length records. SORT/MERGE can also sort variable–length records, providing you specify a record length that is equal to the length of the largest record in the file. For example, examine the following modified version of TEST.IN:

```
.TYPE TEST.IN(RET)
1234567890123456
19  C  SMITH
32  A  JONES
19  B  ABERNATHY
20  C  BROWNING
19  A  HART
20  B  WILSON
20  A  HILL
```

Note how the record lengths vary from 11 characters to 16 characters. To sort the above file on the name field, type the following command string to SORT/MERGE:

```
*TEST.OUT=TEST.IN/VARIABLE/RECORD:16/KEY:8:9:ASCENDING(RET)
```

The following notes apply to the sorting of variable–length records:

1. If you specify a record length that is less than the maximum size record in your file, then those records that are longer than the value you specified are truncated on output.
2. Although the /VARIABLE switch is shown in the command string above, you can omit it and still sort the file correctly. This is because the /VARIABLE switch is the default record–length type.
3. The /KEY switch is specified with the length of the longest field in the record. Thus, for shorter fields, the key field extends beyond the end of the field. This is permissible only if the data type switch is /ALPHANUMERIC. (Data types are discussed in the last section of this chapter.) However, you cannot specify a key field that extends beyond the value given with the /RECORD switch.

The sorted file appears as:

```
.TYPE TEST.OUT(RET)
1234567890123456
19  B  ABERNATHY
20  C  BROWNING
19  A  HART
20  A  HILL
32  A  JONES
19  C  SMITH
20  B  WILSON
```

## 1.7 Sorting a Multiline ASCII File

It is sometimes desirable to sort an ASCII text file in which each record consists of two or more lines. For example, you may wish to sort a file of records having the following format:

```
SMITH
  19
  C
JONES
 32
  A
```

Unfortunately, the control characters (carriage return/line feed) that are used to format the text into lines are really ASCII record delimiters. If you attempt to sort such a file, you will find that what you intended as fields of a single record are interpreted as separate records by SORT/MERGE. The results are not what you expected.

There are no elegant solutions to this problem. You could avoid using carriage returns until the actual end of your record, but you lose the ability to format the record into separate lines. A better solution is to write a program that preprocesses your file before sorting and postprocesses your file after sorting. In the preprocess phase, the program converts all carriage return/line feeds, except the last one in the record, to some other character pair that does not occur in the file. You sort the file, and then postprocess it with the program to restore the carriage return/line feeds. The best solution is to actually create the record with a program, rather than with a text editor. This allows generating only one carriage return/line feed per record and allows you to sort the records on any of the fields. However, as the solutions to the problem grow more technically acceptable, you lose the ease and informality of generating the file with a text editor.

## 1.8 Sorting Uppercase and Lowercase Text

The ASCII collating sequence (see Appendix B) is structured so that lowercase alphabetic characters have a greater value than their uppercase equivalents. As a result, if fields are composed of both uppercase and lowercase characters and you sort your file according to the ASCII collating sequence, then the file is sorted so that characters "A" through "Z" come before characters "a" through "z", but all other characters have their normal position. By using SORT/MERGE's alternate collating sequence facility, you can specify a collating sequence in which uppercase and lowercase alphabetic

characters are equivalent. This allows you to sort uppercase and lowercase characters in normal alphabetic order. First, create a collating sequence file of the form:

```
.SOS UPPER.TXT(RET)
Input: UPPER.TXT
00100 000-"@", "A"="a", "B"="b", "C"="c", "D"="d", (RET)
00200 "E"="e", "F"="f", "G"="g", "H"="h", "I"="i", (RET)
00300 "J"="j", "K"="k", "L"="l", "M"="m", "N"="n", (RET)
00400 "O"="o", "P"="p", "Q"="q", "R"="r", "S"="s", (RET)
00500 "T"="t", "U"="u", "V"="v", "W"="w", "X"="x", (RET)
00600 "Y"="y", "Z"="z" (RET)
00700 $
      ↑
      ESC
*ES(RET)
[UPPER.TXT]
```

When you give a command string, use the /COLLATE switch with the FILE argument and the name of the file containing your alternate collating sequence. For example, the above collating sequence was stored in a file named UPPER.TXT. To use the first command string shown in this chapter for uppercase and lowercase text, specify the following:

```
*TEST.OUT=TEST.IN/FIXED/RECORD:17/KEY:1:10:ASCENDING -(RET)
*/COLLATE:FILE:UPPER.TXT(RET)
```

The result of the command string shown above is to sort file TEST.IN according to the collating sequence stored in file UPPER.TXT. The results are written to file TEST.OUT.

## 1.9 Sorting Nontext Files

This section is designed to help you extend your knowledge of more complex types of sorts that SORT/MERGE can perform. If you are only interested in sorting text files, then you do not need to read this section.

The following is the first command string shown in this chapter:

```
*TEST.OUT=TEST.IN/FIXED/RECORD:17/KEY:1:10:ASCENDING
```

For simplicity, you relied upon the default values for certain specifications that SORT/MERGE requires. The complete, explicit command string appears as:

```
*TEST.OUT=TEST.IN/ASCII/FIXED/RECORD:17/KEY:1:10:ASCENDING -(RET)
*/ALPHANUMERIC(RET)
```

Two new items appear in the above command string:

1. /ASCII switch
2. /ALPHANUMERIC switch

The /ALPHANUMERIC switch specifies the data type of the key field. The data type specifies how the data is to be interpreted. For example, you read in Section 1.5 that ASCII characters are represented internally by a 3-digit octal number (or a 7-bit binary number). Because a single word of data on TOPS-10 contains 36 bits, a word of data holds five ASCII characters, with one bit unused ( $7*5=35$ ). However, examined out of context, this word of data appears to be simply a 36-bit binary number.

For example, consider the following 36-bit binary value:

```
011000101101010111001100001010000110
```

This value can be correctly interpreted as any of the following:

- |                              |                              |
|------------------------------|------------------------------|
| 1. 26530857606               | Fixed point decimal value    |
| 2. 3956710900000000000000.00 | Floating point decimal value |
| 3. 3.9567109E + 20           | Scientific notation          |
| 4. '159BC'                   | ASCII characters             |

Remember that any file you create is a series of these 36-bit words that are subject to the various interpretations shown above. The interpretation of a given word depends entirely on context. If you have created a text file, then you are only interested in interpreting the data as characters. If you have created a file of fixed-point decimal values, then you interpret the file accordingly. You must specify the appropriate interpretation to SORT/MERGE. If you specify an incorrect interpretation, then the results of the sort, while technically correct, are not the results you expect.

"Data type" is the term used to refer to the interpretation of data; and the data-type switch refers to the /KEY switch which is used to specify the correct interpretation of data to SORT/MERGE. The switches for the simpler data types are:

- |                   |                                   |
|-------------------|-----------------------------------|
| 1. /ALPHANUMERIC  | Alphabetic and numeric characters |
| 2. /NUMERIC       | Numeric characters                |
| 3. /COMPUTATIONAL | Fixed-point decimal numbers       |
| 4. /COMP1         | Floating-point decimal numbers    |
| 5. /COMP3         | Packed (4 bits) decimal           |
| 6. /PACKED        | Same as /COMP3                    |

See Section 3.2 for more information on data types.

The second new switch, /ASCII, specifies the recording mode. The recording modes that SORT/MERGE recognizes are:

1. ASCII            7-bit bytes
2. SIXBIT          6-bit bytes
3. EBCDIC          9-bit bytes
4. BINARY         36-bit bytes

All recording modes have a byte size associated with them. The byte size specifies how many bits are used to represent a single unit of data in that recording mode. For the first three recording modes, a byte is equivalent to a character. (The first three recording modes have character sets associated with them.) Thus, an ASCII character is represented by a group of 7 bits, a SIXBIT character is represented by a group of 6 bits, and an EBCDIC character is represented by a group of 9 bits. (See Appendix B for the ASCII, SIXBIT, and EBCDIC character set.) A unit of data in binary recording mode requires all 36 bits of the word. Except for data produced by a COBOL program, you can assume that any character data you generate is ASCII data. COBOL programs generate SIXBIT data by default, unless you have specified ASCII as the data type.

Refer to Section 4.1 for diagrams depicting the recording modes recognized by SORT/MERGE. Also, see Chapter 3 for the discussion of each recording mode switch.

There is also a new concept illustrated by the second command string. That is the concept of file format, which is illustrated by the /ASCII /FIXED switch combination. This particular file format is termed 'fixed-length ASCII'. The file format specifies the type of control characters, control words, or header words that are used to format the data into individual records. Because these format words and control characters are not actually part of the data, SORT/MERGE must strip them from the input, before the actual sorting process can begin. (They are restored on output.) If you incorrectly specify the file format to SORT/MERGE, then the format words and control characters are not properly stripped off and are sorted and output as data. The results are quite unpredictable. (See Chapter 4 for a discussion of file formats.)

## Chapter 2

# How to Use SORT/MERGE

This chapter lists the information needed to sort a file and describes general command formats. Other topics, such as merging files and using command files, are also described. If you are a new user of SORT/MERGE, then you should read Chapter 1 before you read this chapter.

### 2.1 Introduction

SORT/MERGE is a system program that arranges the records of one or more files according to a user-specified sequence. You specify the key fields on which the actual comparisons are to be made, and you also specify the collating sequence to be used in ordering the file. The collating sequence can be the normal ASCII, SIXBIT, or EBCDIC collating sequence; or it can be a unique collating sequence which you have constructed for a particular application. Additionally, you can specify that the file be sorted in ascending or descending order. SORT/MERGE performs two processes when sorting a file.

1. The sort phase, where the input files are read and the records are sifted into runs. These runs are output to one or more temporary areas. You can specify up to 26 temporary areas.
2. The merge phase, where the runs are merged into fewer but longer runs.

This process of sorting and merging continues until a single run remains; this run becomes the output file.

Depending on file size, file order, and available memory, it is possible for the runs to remain in memory. In such a case, temporary areas are not used.

The time required for a given sort diminishes progressively as more disk units and more memory are available. Although tape units can be used for input and output files, disk units must be used for temporary areas. If the input files are on disk, you must have enough temporary space to hold the input files plus two additional copies of each input file.

The minimum memory required is an area large enough to contain the following:

1. Double buffers for
  - a. Each input device
  - b. The output device
  - c. One temporary disk file
2. At least 16 records
3. One record from each temporary file

## **2.2 Running SORT/MERGE**

You must know the following information about the input files in order to sort or merge them:

1. Record size
2. Key size and location
3. Key collating order
  - a. Ascending
  - b. Descending
4. Key data type
  - a. Alphanumeric
  - b. COMP1
  - c. COMP3
  - d. COMPUTATIONAL
  - e. FORMAT (FORTRAN ASCII floating point)
  - f. Numeric
5. Key sign status
  - a. Algebraic sign used/not used for compares



6. Recording mode

- a. ASCII
- b. BINARY
- c. EBCDIC
- d. SIXBIT

7. File attributes

- a. Blocked
- b. Fixed (random)
- c. Variable (sequential)
- d. FORTRAN (ASCII or binary)

In addition, there are control attributes and tape attributes that you can specify. SORT/MERGE must have all this information in order to correctly sort a file. However, you can rely on default switches and default key attributes, so that you need not specify all of the information required by SORT/MERGE.

Although sorting a file can be a simple operation, several factors are often introduced that make it difficult for an uninformed user to sort a file:

1. The existence of control words in the file.

The use of control words is interwoven with the concept of file format, and you may not understand one without understanding the other. Chapter 4 contains diagrams of all major file formats supported by SORT/MERGE, code segments from COBOL and FORTRAN programs that generate these formats, and the SORT/MERGE switch or switch combination used to specify these formats. You should study this chapter until you know what switches to use in specifying your file's format to SORT/MERGE.

2. The existence of line-sequence numbers in the file.

Chapter 1 describes the problems caused by line-sequence numbers.

3. The existence of tab characters in the file.

Chapter 1 describes the problems caused by tab characters in the file.

4. Using a key data type other than alphanumeric or numeric.

Calculating key position and key length for alphanumeric or numeric keys is a relatively straightforward task. (See Chapter 1, if you do not understand how to do this.) However, when using other key data types such as COMPUTATIONAL or FORMAT, calculating key position and length becomes more complex. If you are using a key data type other than ALPHANUMERIC or NUMERIC, be sure to read the description

of the /KEY switch in Chapter 3. You can also find the file format diagram in Chapter 4 that describes your file, and study the key specifications in the diagram.

5. Sorting character data stored in a binary file.

If you specify your file as a standard binary file (/BINARY switch), then you cannot use an alphanumeric or numeric field as a key field. If you specify your file as a mixed-mode binary file (/BINARY/ASCII, /BINARY/SIXBIT, or /BINARY/EBCDIC switches), then you can use an alphanumeric or numeric field as a key field. However, the specification of record length and key starting position changes from words for a binary file to characters for a mixed-mode binary file. See Chapter 4 for a more detailed discussion of standard and mixed-mode binary files.

## 2.3 Command Formats

There are five different functions that can be specified in a command string to SORT/MERGE. The following describes each of these functions and illustrates the general format of the command string that performs the function.

1. SORT — sort the specified files.

switches OUTFIL switches = switches INFIL1 switches, switches  
INFIL2 switches

2. MERGE — merge the specified files.

switches OUTFIL switches = switches INFIL1 switches, switches  
INFIL2 switches /MERGE

3. RUN — leave SORT/MERGE command level and run the specified program.

/RUN (specified anywhere in the command string — program is started when the sort or merge is complete.)

4. EXIT — exit SORT/MERGE and return to monitor level.

/EXIT (specified anywhere in the command string — control returns to monitor level when the sort or merge is complete.)

5. HELP — print the text of the help file.

/HELP (/HELP:SWITCHES prints available SORT/MERGE and SCAN switches.)

SORT/MERGE uses SCAN, the TOPS-10 command scanner to parse the command string. The effect of a switch in a command string depends on the type of switch and its place in the command string. See Chapter 3 for more details. Certain switches are not usable in FORTRAN-called sorts. These are described in Section 2.3.4.

File specifications are given in the standard form:

filnam.ext

Where 'filnam' is a 1- to 6-character filename, and 'ext' is a 1- to 3-character file extension. At least one input file specification and one output file specification must be given. Multiple file specifications are separated by commas. Note that multiple output file specifications are allowed only if the output device is magnetic tape.

You can continue a command string onto a second line by typing a dash. SORT/MERGE prompts the next line with '#', and you can continue typing the command string.

### 2.3.1 Merging Files

If you have two or more files that have been previously sorted on the same key(s) and have the same file format, the files can be merged into one file with the /MERGE switch. When merging files, you must specify the same information as for a sort, plus the /MERGE switch, for example:

```
OUTFIL=INFIL1,INFIL2/RECORD:70/KEY:40:15:ASCENDING/MERGE
```

If you attempt to merge files that do not have the same file formats and are not already in sorted order on the same key field(s), then you receive unpredictable results. To safeguard against this, you can specify the /CHECK switch. This switch causes SORT/MERGE to check the files and generate an error message if any out-of-sequence records are found. See Chapter 3 for more information on the /CHECK switch.

### 2.3.2 Using Command Files

A command file is an ASCII file with SORT/MERGE commands stored in it. By specifying '@filespec', you can cause SORT/MERGE to read a command file and execute the commands stored in it. The following illustrates a command file created with SOS. Note that SORT/MERGE accepts the file with or without line-sequence numbers.

```
.SOS FILE.CCL(RET)
Input: FILE.CCL
00100  SORT1,FIL=IN1,FIL/RECORD:80/KEY:1:10(RET)
00200  SORT2,FIL=IN2,FIL/RECORD:80/KEY:1:10(RET)
00300  MERGE,FIL=SORT1,FIL,SORT2,FIL/RECORD:80/KEY:1:10/MERGE(RET)
00400  /EXIT(RET)
00500  $
      ↑
      ESC
*ES(RET)

[FILE.CCL]
```

The following command causes SORT/MERGE to execute the command file shown above:

```
*@FILE.CCL(RET)
```

If you do not specify a file extension, SORT/MERGE searches for a file specification of the form: filnam.CCL.

## 2.4 Using SORT/MERGE from a COBOL Program

COBOL users can run SORT/MERGE from a COBOL program by using the COBOL SORT and MERGE statements. Because this statement is part of the COBOL syntax, it is not described here. See the COBOL manual (*COBOL-68* and/or *COBOL-74 Language Reference Manual*) for information on the SORT or MERGE statement.

## 2.5 Using SORT/MERGE from a FORTRAN Program

FORTRAN users can call SORT/MERGE from a FORTRAN program by using a CALL statement, for example:

```
CALL SORT('OUTPUT.FIL=INPUT.FIL/RECORD:50/KEY:1:5')
```

Note that the actual SORT/MERGE command string can be enclosed in single quotation marks, or passed directly as an array containing the text of the command. The following example illustrates a FORTRAN subroutine that accepts a SORT/MERGE command string typed from a time-sharing terminal, writes the string into an array, and passes the entire array to SORT/MERGE.

### NOTE

When you pass an array to SORT/MERGE in a FORTRAN program, the array must end with a null (for example, CMD0(21) equivalence to CMD(20)).

```
      DIMENSION CMD0(21)
      DIMENSION CMD(20)
      EQUIVALENCE(CMD,CMD0)
10    TYPE 1000
1000  FORMAT (' *$)
      ACCEPT 2000,CMD
2000  FORMAT (20A5)
      CALL SORT(CMD)
      GO TO 10
      END
```

If your installation has built a FORTRAN VM SORT program (FSORT), then the CALL SORT statement in the above FORTRAN program executes FSORT. FSORT contains the same features as the stand-alone SORT/MERGE program.

However, if your installation has not built a FORTRAN VM SORT, then the FORTRAN-called sort is somewhat smaller but does not have access to all the features of stand-alone sorts. The following are the restrictions:

1. All switches must be specified to uniqueness. This means that /RECORD must be specified at least to /REC, since this version of sort does not have the concept of a switch unique in any abbreviation. For example, if you give the following switch from SORT/MERGE command level:

/H:S

you receive a display of all the switches that can be specified to SORT/MERGE. Any switch marked with an asterisk (\*) is recognized even if only one letter of the switch is typed. For instance, /RECORD can be specified as /R. This version of sort does not have this facility and generates an error message if you do not type enough characters of the switch to make it unique.

2. Switches must always follow file specifications, for example,

/switch1/switch2 TEST.OUT=/switch3/switch4 TEST.IN...

is illegal, while:

TEST.OUT/switch1/switch2=TEST.IN/switch3/switch4...

is legal.

There is no concept of modified switches.

3. Switches cannot be read from the SWITCH.INI file.
4. The following switches are not recognized:
  - /BLOCKED
  - /COMP3 (or /PACKED)
  - /COMPUTATIONAL
  - /EBCDIC
  - /SIXBIT
  - all SCAN switches (except /DENSITY)
5. There is no continuation line feature. If a command string is excessively long, it should be passed directly to a FORTRAN sort.

There are two SORT/MERGE switches that are of special interest to FORTRAN users:

1. /ERROR:
2. /FATAL:

The first switch, /ERROR:, allows you to specify an error-condition GOTO address. If SORT/MERGE encounters an error, control transfers to the specified address in the FORTRAN program that called SORT/MERGE. The second switch, /FATAL:, allows you to save a SORT/MERGE error code in a specified address. See Chapter 3 for more details on these two switches.

## Chapter 3

# **SORT/MERGE Switches**

This chapter gives detailed explanations of all SORT/MERGE switches and certain SCAN switches that are useful to you. (For a quick-reference list of SORT/MERGE switches, see Appendix A, SORT/MERGE Functions and Switches.)

There are six categories of switches in SORT/MERGE:

1. Required switches
2. Key data type switches
3. Recording-mode switches
4. File switches
5. Control switches
6. Tape switches

Each of these switch groups varies according to the range of its effect, which can be:

1. Global — Affects the entire process. Global switches can be placed anywhere in the command string without changing their effect.
2. Local — Affects only the preceding file specification. Local switches can also be placed in front of file specifications, but they still affect only that file specification.

3. Position dependent — Depends on its position in the command string. If placed before a file specification, a position dependent switch affects that file and all following files until an equal sign or an end-of-line character is encountered. Thus, the switch is position dependent over the file specifications that follow it. If placed after a file specification, it affects only that file specification, for example,

```
*/Switch-A OUT.FIL/Switch-B = /Switch-C FIRST.FIL/Switch-D -(RET)  
#/Switch-E SECOND.FIL/Switch-F/Switch-G THIRD.FIL/Switch-H(RET)
```

Switches A and B apply to the output file (OUT.FIL). Switch C applies to all input files. Switch D applies only to the first input file (FIRST.FIL). Switch E applies to the second and third input files (SECOND.FIL and THIRD.FIL). Switch F applies only to the second input file. Switches G and H apply to the third input file.

4. Modified position dependent — Modified position dependent switches work as follows (the /BLOCKED switch is used for example purposes):

- a. If /BLOCKED is not specified anywhere in the command string, then no files are blocked.
- b. If /BLOCKED is specified anywhere in the command string, then those files for which no explicit /BLOCKED switch is specified have a blocking factor determined by default. The default blocking factor is taken from the last position dependent /BLOCKED switch in the command string. If there is no position dependent /BLOCKED switch, then the first /BLOCKED switch in the command string is used. Thus, in the following command,

```
OUT.FIL = IN1.FIL.IN2.FIL.IN3.FIL
```

all files are unblocked. In this command,

```
OUT.FIL = IN1.FIL.IN2.FIL.IN3.FIL/BLOCKED:3
```

all files have a blocking factor of 3. In this command,

```
OUT.FIL = IN1.FIL.IN2.FIL/BLOCKED:2/BLOCKED:3 IN3.FIL.IN4.FIL
```

OUT.FIL, IN1.FIL, IN3.FIL, and IN4.FIL have blocking factors of 3. IN2.FIL has a blocking factor of 2.

#### NOTE

Key data type switches are in a special category and can only be specified after the key switch they are intended to modify.



## 3.1 Required Switches

Every SORT/MERGE command string must contain the following two switches: /RECORD and /KEY. Both are global and required. These switches are described below.

### 3.1.1 /RECORD Switch

The /RECORD switch specifies the length of the record and has the following format:

/RECORD:n

For all file formats except standard binary (described with the /BINARY switch), n indicates the number of characters in the record. For a standard binary file, n indicates the number of words in the record.

If the file contains any control words (such as SIXBIT and EBCDIC header words or FORTRAN Logical Segment Control Words) or control characters (such as ASCII carriage return/line feed), you should not include these control words in the record length. If you specify a value with the /RECORD switch that is less than the actual size of the records you are sorting, the records are truncated for variable-length records. For fixed-length records, you receive an error message or undefined results. If you are sorting variable-length records, you should specify the length of the largest record in the file.

### 3.1.2 /KEY Switch

The /KEY switch describes the characteristics of the key field, the field on which comparisons are made when the records in a file are sorted. Three characteristics of the key field are:

1. key starting position
2. key length
3. key collating order

#### NOTE

Chapter 4 contains diagrams of key fields and key specifications for all major file formats recognized by SORT/MERGE. You should read that chapter after reading this section.

**3.1.2.1 /KEY Switch Format** — The /KEY switch has the format:

/KEY:n:m [:x]

where:

n is the position of the first character or word of the key. (If the key begins with the first character or word of the record, then n = 1.)

m is the length of the key.

x is the key collating order, either ASCENDING or DESCENDING.

The key position and key length values must be specified. If you do not specify a collating order, then it defaults to ASCENDING.

**3.1.2.2 Key Starting Position** — The key starting position indicates where in the record the key field begins. This position is usually specified in characters. However, for COMPU and COMP1 data types, there are file formats that require the key starting position to be given in words, rather than characters. The following illustrates when to use characters and when to use words:

<u>Data Type</u>	<u>Use</u>
ALPHA	Characters
NUMERIC	Characters
FORMAT	Characters
COMP3	Characters
COMPU	Characters (mixed-mode binary) words (standard binary)
COMP1	Characters (mixed-mode binary) words (standard binary)

If the file is a mixed-mode binary file (a binary file containing characters) and is specified to SORT/MERGE as a mixed-mode binary file (for example, /BINARY/ASCII), then the key starting position is always calculated in characters. This is true even for COMPU or COMP1 keys. However, if a mixed-mode binary file is specified to SORT/MERGE as being a standard binary file (for example, /BINARY), then the key starting position for COMPU or COMP1 keys is specified in words; and the fields containing characters cannot be used as key fields. See Chapter 4 for more details on mixed-mode binary files.

If you are using two adjacent key fields such that some key field is preceded by a COMP3 (or PACKED) key field, then use the following formula to calculate the first character position in the second key field.

$$\frac{(n + 1 + 1)}{2} + s$$

where:

n is the number of decimal digits in the COMP3 field.

s is the starting position of the COMP3 field.

When using this formula, round a fractional result to the next lower integer.

**3.1.2.3 Key Length** — The key length refers to the length of the key field. Key length is specified either in characters or decimal digits, depending on the data type:

<u>Data Type</u>	<u>Use</u>
ALPHA	Characters
NUMERIC	Characters
FORMAT	Characters
COMPU	Decimal digits
COMP1	Decimal digits
COMP3	Decimal digits

**3.1.2.4 Key Collating Order** — You can specify either ASCENDING or DESCENDING collating order for any file that you wish to sort. If the key data type is ALPHA, then ASCENDING causes SORT/MERGE to sort the file in direct agreement with the character set used in the file (ASCII, SIXBIT, or EBCDIC). DESCENDING causes SORT/MERGE to sort the file in reverse agreement with the character set used. If the key data type is FORMAT, NUMERIC, COMPUTATIONAL, COMP1, or COMP3, then ASCENDING causes SORT/MERGE to sort the file in ascending numeric order. DESCENDING causes SORT/MERGE to sort the file in descending numeric order.

**3.1.2.5 Key Data Type** — Data type refers to the type of data that makes up the key field. All key data type switches modify the key switch that they follow. The data type switch can be one of the following:

/ALPHANUMERIC (/ALPHA)

indicates that the key field is composed of alphabetic and numeric characters.

/NUMERIC

indicates that the key field is composed of numeric characters. If the key field contains an algebraic sign ('+' or '-'), or over-punched trailing sign), and you wish to exclude the sign in the key comparisons, then specify the /UNSIGNED switch also. Otherwise, the sign status defaults to /SIGNED. This data type can be used only for data without a decimal point or with an implied decimal point. The data cannot include the character '.'.

## **/FORMAT**

indicates that the key field is in one of four FORTRAN ASCII numeric formats. The /FORMAT switch has the following format:

**/FORMAT:nPaw.d**

where 'n' is a scaling factor, 'a' is a FORTRAN format type, 'w' is the width of the key field, and 'd' is the number of decimal places in the number. The 'n' can be a positive or negative number. If the 'nP' is not specified, no scaling is done. If the 'w.d' is not specified, FORTRAN free format is assumed. The 'a' can be one of the following arguments:

<u>Argument</u>	<u>Format</u>
D	Double-precision scientific format
E	Scientific format
F	Floating-point format
G	General format

### **NOTE**

A key having a FORMAT data type always implies that the recording mode is ASCII.

## **/COMPUTATIONAL (/COMPU)**

indicates that the key field is a fixed-point binary number. If the number is 10 decimal digits or less, then the key occupies one word of storage. If the number is greater than 10 decimal digits, then the number occupies two words of storage. This data type is signed by default; so if you wish to exclude the sign from the key comparisons, you should also specify the /UNSIGNED switch.

## **/COMP1**

indicates that the key field is a floating-point binary number. If the number is 10 decimal digits or less, then the key occupies one word of storage. If the number is greater than 10 decimal digits, then the number occupies two words of storage. This data type is signed by default; so if you wish to exclude the sign from the key comparisons, you should also specify the /UNSIGNED switch.

## **/COMP3 (or /PACKED)**

indicates that the key field is an /EBCDIC packed-decimal number. This data type is signed by default; so if you wish to exclude the sign from the key comparisons, you should also specify the /UNSIGNED switch.

The following two tables illustrate the various field descriptors for COBOL and FORTRAN programs, and give the appropriate SORT/MERGE data type for each program field descriptor:

**Table 3–1: Field Descriptors for COBOL**

COBOL PHRASE	SORT/MERGE KEY DATA TYPE
PIC A( )	/ALPHA
PIC X( )	/ALPHA
PIC 9( )	/NUMERIC/UNSIGNED
PIC S9( )	/NUMERIC/SIGNED
PIC 9( ) COMPUTATIONAL	/COMP
PIC 9( ) COMP–1	/COMP1
PIC 9( ) COMP–3	/COMP3

**NOTE**

Data types /NUMERIC, /COMP, /COMP1, and /COMP3 can be further specified as /SIGNED or /UNSIGNED. COBOL COMP items can be single precision (1–10 decimal digits, occupying one word of storage) or double precision (11–18 decimal digits, occupying two words of storage).

**Table 3–2: Field Descriptors for FORTRAN**

FORTRAN FORMAT DESCRIPTOR	SORT/MERGE KEY DATA TYPE
A	/ALPHA
D	/FORMAT:Dw.d
E	/FORMAT:Ew.d
F	/FORMAT:Fw.d
G	/FORMAT:Gw.d
H	/ALPHA
I	/NUMERIC
L	/ALPHA
O	/ALPHA
R	/ALPHA
FORTRAN DATA TYPE	SORT/MERGE KEY DATA TYPE
INTEGER	/COMPU
REAL	/COMP1

**NOTE**

Data types /NUMERIC, /COMPU, /COMP1, and /FORMAT can be further specified as /SIGNED or /UNSIGNED. If the specified length of a COMPU or COMP1 key is 10 decimal digits or less, SORT/MERGE interprets the key as one word long. If the specified length is 11 decimal digits or more, SORT/MERGE interprets the key as two words long.

**3.1.2.6 Key Sign Status** — The key sign status indicates whether or not the algebraic sign of the key field is to be used in making key comparisons. Key fields can incorporate the algebraic sign in the following ways:

<u>Key Data Type</u>	<u>Sign Representation</u>
NUMERIC	Characters '+' or '-', or over-punched character
COMPU	Sign bit (bit 0)
COMP1	Sign bit (bit 0)
COMP3	4-bit trailing sign bit

Specifying /SIGNED causes SORT/MERGE to use the algebraic sign in making key comparisons. Specifying /UNSIGNED causes SORT/MERGE to ignore the algebraic sign in making key comparisons.

### WARNING

If you specify the /SIGNED switch, then the method of sign representation used in your key field can affect the calculation of key starting position and length. Note the effect of various sign representations on the key field:

#### NUMERIC data type

1. If the key field is written with a COBOL field descriptor of the form: PICTURE S99, then the sign is represented by an overpunch on the last numeric character in the field. For example, -521 would be represented by 52J. This type of sign representation does not use an extra character position to represent the sign, and the key field is not affected.
2. If the key field contains '-123' (where the entire field consists of characters), and you specify the /SIGNED switch, then the sign occupies a character position and is part of the key. As the sign character precedes the key, the key starting position begins with the sign character, not with the first numeric character in the field. Therefore, the key length must include the sign character.

#### FORMAT data type

If the key field was written in any FORTRAN scientific notation format, then the sign is represented by the character '+' or '-'. For example, in the following data item: -0.51E+03, the sign (as well as the period and the letter 'E', the exponent's sign, and the exponent) occupy character positions within the key and are considered part of the key. This is true for either /SIGNED or /UNSIGNED sign status.

#### COMPUTATIONAL and COMP1 data types

The sign is represented by a single bit for these data types, and the key field is not affected by sign status.

COMP3 data type (or PACKED)

In EBCDIC packed-decimal notation, the sign is represented by a 4-bit pattern in the last digit position of the field. The length of the sign representation is never added to the key length. The key field is not affected by the sign status. However, the sign representation can affect a key field position following the COMP3 key field.

To determine the length of a COMP3 (or PACKED) data type key field in 9-bit bytes, add the length (L) of the field in digits plus two divided by two, truncating any remainder. Thus, a nine digit key field equals a key field length of five bytes.

$$L + 2 / 2 = \text{key length field}$$
$$9 + 2 / 2 = 5 \text{ bytes}$$

## 3.2 Recording Mode Switches

Recording-mode switches designate the coding scheme used in the data files that you are sorting. There are four recording-mode switches in SORT/MERGE, and all are global:

1. /ASCII
2. /EBCDIC
3. /SIXBIT
4. /BINARY

The first three switches (/ASCII, /EBCDIC, and /SIXBIT) have collating sequences associated with them, and only one of these three switches can be specified in a command string.

### 3.2.1 /ASCII Switch

The /ASCII switch indicates that the file is recorded in ASCII mode. This switch has the following format:

/ASCII

Characteristics:

1. Defaults to /VARIABLE, though /FIXED can be specified instead. If /ASCII/BINARY is specified, then the default is /FIXED, and /VARIABLE can only be given with /ASCII/BINARY/FORTRAN.
2. The /COMP3 (or /PACKED) switch cannot be given with the /KEY switch when the recording-mode is ASCII.

### 3.2.2 /SIXBIT Switch

The /SIXBIT switch indicates that the file is recorded in SIXBIT mode. This switch has the following format:

/SIXBIT

Characteristics:

1. Defaults to /VARIABLE, although /FIXED can be specified. If /SIXBIT/BINARY is specified, /FIXED is the default, and /VARIABLE can only be specified with /SIXBIT/BINARY/FORTRAN.
2. The /COMP3 (or /PACKED) and /FORMAT data type switches cannot be specified with the /SIXBIT switch.

### 3.2.3 /EBCDIC Switch

The /EBCDIC switch indicates that the file is recorded in EBCDIC mode. This switch has the following format:

/EBCDIC

Characteristics:

1. Defaults to /FIXED, though /VARIABLE can be specified. If /EBCDIC/BINARY is specified, then /VARIABLE can only be specified with /EBCDIC/BINARY/FORTRAN.
2. The /FORMAT data type switch cannot be specified with the /EBCDIC switch.

### 3.2.4 /BINARY Switch

The /BINARY switch indicates that the file is recorded in binary mode. This switch has the following format:

/BINARY

Characteristics:

1. Defaults to /FIXED. The /VARIABLE switch is allowed only for /BINARY/FORTRAN.
2. If the /COMP3 (or /PACKED) switch is specified with the /KEY switch, then the default is /BINARY/EBCDIC, and /ASCII or /SIXBIT cannot be specified.
3. If the /COMPU or /COMP1 switch is specified with the /KEY switch, then the default is /BINARY/SIXBIT, although /ASCII or /EBCDIC can be specified to override this and the key size is specified in characters, not words.
4. If the /FORMAT switch is specified with the /KEY switch, then the recording mode is /BINARY/ASCII, and neither /SIXBIT nor /EBCDIC can be specified.



## 3.3 File Switches

File switches are used to specify various file attributes such as word alignment, blocked files, and file type (fixed or variable). All file switches are global, except the /BLOCKED switch, which is a modified position dependent switch.

### 3.3.1 /AFTER Switch

The /AFTER switch indicates where the output record is written in relation to the carriage–return/line–feed characters. This switch has the following format:

/AFTER

Characteristics:

1. The /AFTER switch specifies to place the output record after a carriage–return/line–feed.
2. This switch is designed to be compatible with COBOL–74 programs.
3. The /AFTER switch is a local switch.
4. The /AFTER switch is not required for input files.

### 3.3.2 /ALIGN Switch

The /ALIGN switch indicates that each output record begins on a word boundary. This switch has the following format:

/ALIGN

Characteristics:

1. Your file must be recorded in ASCII mode.
2. This is a global switch, but affects only the output file.
3. This switch increases the rate of data transfer, but produces a somewhat larger output file.
4. Line–sequenced ASCII files must be word aligned and are output in word–aligned format whether the /ALIGN switch is specified or not.

### 3.3.3 /BEFORE Switch

The /BEFORE switch indicates where the output record is written in relation to the carriage–return/line–feed characters. This switch has the following format:

/BEFORE

Characteristics:

1. The /BEFORE switch places the output record before the carriage-return/line-feed.
2. This switch is the default to SORT/MERGE.
3. The /BEFORE switch is a local switch.
4. The /BEFORE switch is not required for input files.

### 3.3.4 /BLOCKED Switch

The /BLOCKED switch indicates the blocking factor to be used for blocked files. This is a modified position dependent switch and has the following format:

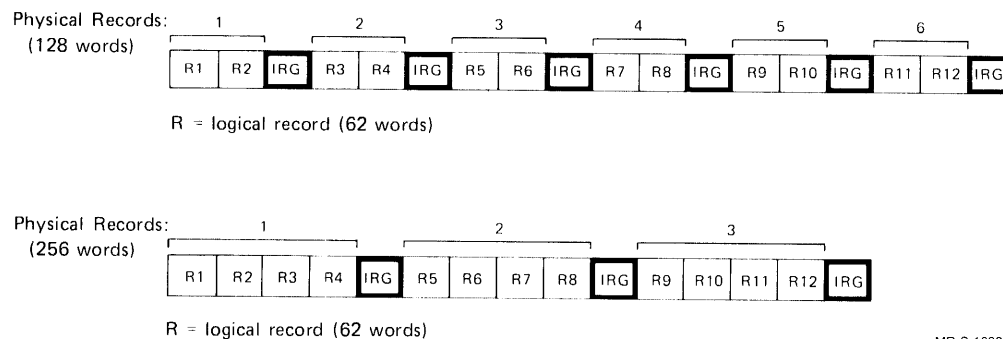
/BLOCKED:n

where:

n is a decimal number indicating the number of logical records per logical block.

Characteristics:

On tape, each physical record is separated from the next record by an inter-record-gap (IRG), which is a space containing no data. As each IRG is approximately 95 words long, a large amount of tape storage is wasted if there is an excessive number of IRGs. This occurs if the length of a physical tape record is not appreciably larger than the size of an IRG. The default size of a physical tape record is set by the monitor at 200 (octal) words. You can override the default value with the SET BLOCKSIZE monitor command (see the *TOPS-10 Operating System Commands Manual*). An alternative to relying on the system parameter for physical tape record size is to specify the /BLOCKED switch to SORT/MERGE. The argument to this switch specifies how many logical records are contained in a physical tape record. The optimal range of physical tape record size is about 500 to 2000 words. By specifying a blocking factor that creates a physical record size that falls within that range, you can substantially improve utilization of tape storage space by reducing the number of IRGs. This is illustrated in the following diagram:



MR-S-1698-81

Note that the second file uses half as many IRGs as the first file.

#### NOTE

EBCDIC magnetic tapes must be blocked. If no blocking factor is specified for an EBCDIC magnetic tape, a blocking factor of 1 is assumed.

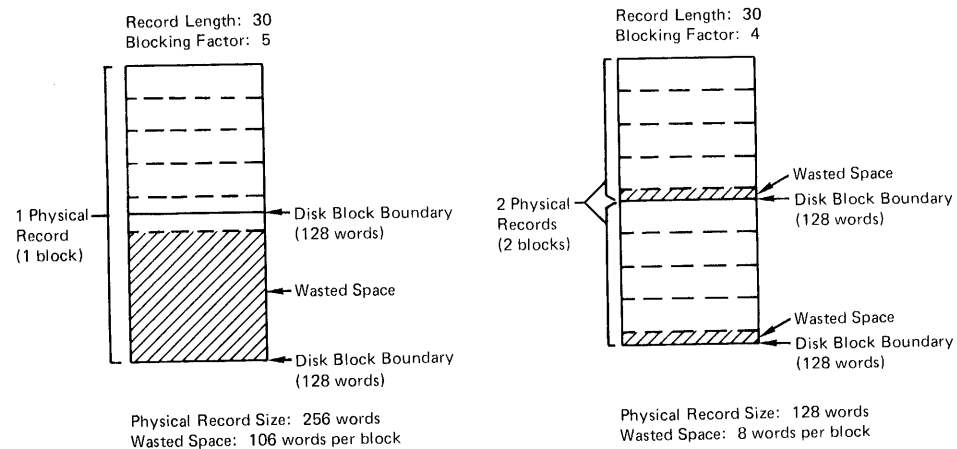
Blocking files also causes somewhat faster access to disk. However, as the size of the physical disk record is computed on the basis of disk blocks (128 words), the blocking factor must be chosen carefully; or you can cause an excessive waste of disk storage space. Use the following formula to calculate physical record size:

$$(n + c1) * m + c2$$

where:

- n = record length in words
- m = blocking factor
- c1 = record control words in a single record
- c2 = block control words in a single block

Now compare the calculated value with the closest multiples of 128 (that is, 128, 256, 384, 512, and so on). If the calculated value is slightly less than or exactly equal to a multiple of 128, then the blocking factor makes efficient use of disk storage. If the calculated value does not meet the previous conditions, then you should try increasing or decreasing the blocking factor until the conditions are met. The following example demonstrates the effect that selection of blocking factors has on disk storage utilization:



MR-S-1699-81

### 3.3.5 /FIXED Switch

The /FIXED switch indicates that the file contains fixed-length records. This switch has the following format:

/FIXED

Characteristics:

This switch may or may not be in effect by default, depending on the particular recording-mode switch being used. See the description of the appropriate recording-mode switch to determine when /FIXED is in effect by default.

### **3.3.6 /FORTRAN Switch**

The /FORTRAN switch indicates that the file is a FORTRAN file written in either ASCII or binary recording mode. This switch has the following format:

/FORTRAN

Characteristics:

1. If you specify /FORTRAN/ASCII, then /ALIGN is in effect by default.
2. If you specify /FORTRAN/BINARY, then SORT/MERGE checks for Logical Segment Control Words (LSCWs). Refer to Chapter 4 for a description of LSCWs.

### **3.3.7 /NOCRLF Switch**

The /NOCRLF switch (No Carriage Return/Line Feed switch) indicates that both the input and the output files are /FIXED /ASCII records containing no carriage control characters. This switch has the following format:

/NOCRLF

Characteristics:

1. The /NOCRLF switch is a global switch.
2. Both the input and output files must be fixed length.
3. This switch should be specified when you sort text files that contain no carriage-return/line-feed. Such files may come from a foreign source or you may be sorting the files to go to a foreign source.

### **3.3.8 /RANDOM Switch**

The /RANDOM switch indicates that a FORTRAN random file is to be processed by SORT/MERGE. This switch has the following format:

/RANDOM

Characteristics:

This switch has exactly the same effect as the /FIXED switch. The /RANDOM switch is provided for the convenience of FORTRAN users.

### 3.3.9 /SEQUENTIAL Switch

The /SEQUENTIAL switch indicates that FORTRAN sequential files are to be processed by SORT/MERGE. This switch has the following format:

/SEQUENTIAL

Characteristics:

This switch has exactly the same effect as the /VARIABLE switch. The /SEQUENTIAL switch is provided for the convenience of FORTRAN users.

### 3.3.10 /VARIABLE Switch

The /VARIABLE switch indicates that the input file contains variable-length records. This switch has the following format:

/VARIABLE

Characteristics:

1. The /VARIABLE switch may or may not be in effect by default, depending on the particular recording mode you are using. See the description of the appropriate recording-mode switch to determine when /VARIABLE is in effect by default.
2. For general processing, variable length records process slower than fixed length records.

## 3.4 Control Switches

The control switches are used to control parameters such as memory size, tree size, and temporary area allocation, and to perform other functions such as checking files to be merged and specifying alternate collating sequences. All control switches are global, except for the /PHYSICAL switch, which is a position dependent switch.

### 3.4.1 /CHECK Switch

The /CHECK switch indicates to check the order of input files during a merge. This switch has the following format:

/CHECK

Characteristics:

1. The /CHECK switch can be used only if the /MERGE switch has been specified.
2. This switch increases the number of comparisons and causes some increase in run time. However, it protects against erroneously merging unsorted files.

3. If out-of-sequence records are found, then the following error message is generated:

?SRTMRS MERGE RECORD *n* NOT IN SEQUENCE FOR filename

### 3.4.2 /COLLATE Switch

The /COLLATE switch indicates the collating sequence to be used when sorting a file. This switch has the following format:

/COLLATE:argument

Characteristics:

One of the following arguments must be specified with the /COLLATE switch:

1. ASCII
2. EBCDIC
3. FILE:filename
4. LITERAL:/collating sequence/
5. ADDRESS:address

The first two arguments (ASCII and EBCDIC) allow you to sort data in one recording mode according to the collating sequence associated with another recording mode. For example, if you have specified /ASCII/COLLATE:EBCDIC for a file, then the data is interpreted as ASCII data, but is sorted according to the EBCDIC collating sequence. Similarly, you can specify /EBCDIC/COLLATE:ASCII and sort EBCDIC data according to the ASCII collating sequence.

The remaining arguments (FILE:, LITERAL:, and ADDRESS:) allow you to specify your own collating sequence, either as a file or as a literal. The ADDRESS argument is for FORTRAN users in a format of ^*n* construction, where "*n*" is the number of the calling argument in the calling argument list. (See the "addr" description of the /ERROR: switch in Section 3.4.4.) It is essential the FORTRAN user supplies a new complete sequence that replaces the existing ASCII, SIXBIT, or EBCDIC sequence for purposes of sorting and collating. The individual items in your collating sequence can be specified as follows:

"A","B","C","D","E"="F","G"-"N",117="P",121-"S","T"=120,125=126

There are three functions illustrated in the above example:

1. Equivalence — allows you to make particular characters sort as if they were equal to other characters.

- a. "E"="F" indicates that "F" is assigned the same internal value as "E". Thus, "F" has been moved from its normal position in the collating sequence (whichever sequence you are using) and now has the same position as "E". "E"s and "F"s are equivalent and are sorted accordingly.
  - b. 117="P" indicates that "P" is equal to the character with the octal value 117 (that is, "O") in the ASCII sequence. This does not affect the character whose normal internal representation is 117.
  - c. "T"=120 indicates that the character whose octal value is 120 (that is, "P") is now equivalent to T in the new collating sequence.
  - d. 125=126 indicates that the character whose octal value is 126 (that is, "V") is equivalent to the character whose octal value is 125 (that is, "U").
2. Abbreviation — allows you to specify a range of characters without having to type each one individually. For example, "G"—"N" is an abbreviated form of "G","H","I","J","K","L","M","N".
  3. Completion — SORT/MERGE must have a complete alternate collating sequence before it can correctly sort or merge the data. (The term 'complete alternate collating sequence' means an alternate collating sequence in which every character from the recording mode in effect is specified.) If you do not specify an alternate collating sequence that is complete, then SORT/MERGE completes it by adding the missing characters (in their normal order) to the end of the partial collating sequence you specified. For example, if you specify only the following items for your collating sequence:

"D",105,"F"

(which is equal to "D"—"F") and the recording mode is ASCII, then the actual collating sequence is made up of the characters you specified, followed by the normal ASCII collating sequence (minus the characters you specified). The complete alternate collating sequence is as follows:

"D",105,"F",000—"C","G"—177

To further illustrate the various ways an alternate collating sequence can be defined, note how the following four sequences are equivalent:

1. "A","B","C","D","E"="G","Z"
2. "A"—"D",105="G","Z"
3. 101—"D","E"=107,"Z"
4. 101–104,105=107,132

## NOTE

All data in the collating sequence file or in the literal is ASCII data.

If you wish to store your alternate collating sequence in a file, you can create the file using a text editor and save it with or without line sequence numbers.

If you specify your alternate collating sequence as a literal, the sequence must be delimited by a character that does not appear in the literal itself, for example:

```
/COLLATE:/LITERAL:/"A","B","C"—"N"/
```

or

```
/COLLATE:/LITERAL:M"A","B","C"—"N"M
```

If you specify a complete collating sequence as a literal, then you may not have any character to use as a delimiter, unless you do one of the following:

1. Specify a range of characters, such as "A"—"Z". This frees characters B through Y for use as delimiters.
2. Specify an octal value for some character. For instance, if you specify 057 instead of "/", then you can use the slash character (/) as a delimiter.

If the recording mode is SIXBIT, then you can successfully use any lower-case character as a delimiter.

### 3.4.3 /CORE Switch

The /CORE switch indicates the size of SORT/MERGE's low segment. This switch has the following format:

/CORE:nK      specifies low segment in K (1024 words)

/CORE:nP      specifies low segment in pages (512 words)

Characteristics:

SORT/MERGE's default memory-allocation algorithm generally tries to use about half of the available physical memory on the system, up to your job's physical memory limit. If necessary, SORT/MERGE exceeds your job's physical limit (causing the sort to become virtual), but it never exceeds 128K. The /CORE switch allows you to override the defaulting algorithm. See Chapter 6, SORT/MERGE Performance Considerations, for more information on the use of the /CORE switch.



### 3.4.4 /ERROR Switch

The /ERROR switch indicates the absolute address that execution control should transfer to in the event that SORT/MERGE encounters a fatal error. This switch has the following format:

/ERROR:addr

Characteristics:

The 'addr' is an octal integer specifying the absolute address that control should transfer to. This switch was designed for FORTRAN users, who can specify it in one of two ways:

```
CALL SORT('OUTFIL=INFIL/switches/ERROR:3454673')
```

or

```
CALL SORT('OUTFIL=INFIL/switches/ERROR:'2',$99)
```

In the first example, an absolute octal address is used.

In the second example, the expression ``2' is used to indicate that the argument to the /ERROR switch is the second (^2) calling argument in the calling argument list, the statement label 99 (specified as \$99). This allows you to indirectly specify a symbolic address. At compile time, the symbolic address is converted to a relocatable address. Then, at load time, the relocatable address is converted to the absolute address that SORT/MERGE requires.

The /ERROR switch is a global switch.

### 3.4.5 /EXIT Switch

The /EXIT switch indicates that you want to stop execution and exit to TOPS-10 command level. This switch has the following format:

/EXIT

### 3.4.6 /FATAL Switch

The /FATAL switch indicates the absolute address in which to store the error code if SORT/MERGE encounters a fatal error. This switch has the following format:

/FATAL:addr

Characteristics:

The 'addr' is an octal integer specifying the absolute address where the error code should be stored. The error code is a ASCII value of the form SRTxxx, where xxx is a 3-letter code for the error type. These codes and their associated error messages and explanations are listed in Chapter 5, Error Messages.

This switch is designed for FORTRAN users who can specify it in one of two ways:

```
CALL SORT('OUTFIL = INFIL/switches/FATAL:347231')
```

or

```
CALL SORT('OUTFIL = INFIL/switches/FATAL:'2',ERRCOD)
```

In the first example, an absolute address is given.

In the second example, the expression ``2' indicates that the argument to the /FATAL switch is the second calling argument in the call string: ERRCOD. This allows you to indirectly specify a symbolic address. At compile time, the symbolic address is converted to a relocatable address. Then, at load time, the relocatable address is converted to the absolute address that SORT/MERGE requires.

### 3.4.7 /LEAVES Switch

The /LEAVES switch indicates the size of the binary tree that SORT/MERGE is to use during the sort phase of a sort. This switch has the following format:

```
/LEAVES:n
```

Characteristics:

1. The argument 'n' is a decimal number that indicates how many leaves (or terminal nodes) the binary tree is to have.
2. This switch does not affect the merge phase of a sort and is illegal with the MERGE command.

Ordinarily, you would rely on SORT/MERGE's default tree algorithm to define the size of the tree. However, if you are concerned with improving the performance of a particular sort, then you can specify the tree size with the /LEAVES switch. See Chapter 6, SORT/MERGE Performance Considerations, for more information on using the /LEAVES switch.

### 3.4.8 /MAXTEMP Switch

The /MAXTEMP switch indicates the maximum number of temporary files that can be used during a sort or merge. This switch has the following format:

```
/MAXTEMP:n
```

Characteristics:

1. This switch is a global switch.
2. The maximum number (n) of temporary files that you can specify is 26. This is also the default number, if you do not specify the /MAXTEMP switch.

### 3.4.9 /MESSAGE Switch

The /MESSAGE switch indicates how much of an error warning or informational message you want printed. This switch has the following format:

/MESSAGE:argument

Characteristics:

The possible arguments are:

1. [NO]PREFIX — determines whether or not the message code (for example, SRTxxx) is printed.
2. [NO]FIRST — determines whether or not the text portion of the message is printed.

The arguments can be combined by specifying them in parenthesis. For example, /MESSAGE:(NOPREFIX,FIRST) prints the text of the message while suppressing the message code. If you specify /MESSAGE:(NOPREFIX,NOFIRST), however, you still get the text portion of the message.

### 3.4.10 /OPTION Switch

The /OPTION switch indicates the lines in SWITCH.INI that you want to reference. This switch has the following format:

/OPTION:name

Characteristics:

1. The 'name' argument must appear in the SWITCH.INI file in the form of:  
  
SORT:name/switch1/switch2/switch3...
2. By specifying this switch, you can force SORT/MERGE to read that particular line(s) in SWITCH.INI that 'name' refers to and apply any switches in the line to your sort.

### 3.4.11 /PHYSICAL Switch

The /PHYSICAL switch indicates to suppress logical name assignments in file specifications. The result is that only physical devices are searched for the data file(s). This switch has the following format:

/PHYSICAL

### 3.4.12 /PRIORITY Switch

The /PRIORITY switch indicates the disk priority for your SORT/MERGE job. This switch has the following format:

/PRIORITY:n

#### Characteristics:

1. The argument "n" can be a numeric value from -3 to 3. A value of +3 is the highest priority that can be specified; -3 is the lowest priority. Higher priority jobs may process faster depending upon the queue priority.
2. The default setting for the /PRIORITY switch is 0. However, the default can be changed with the DISK. monitor call. (See the *TOPS-10 Monitor Calls Manual*.)

#### 3.4.13 /RUN Switch

The /RUN switch indicates that you wish to run a specified program from SORT/MERGE command level. This switch has the following format:

/RUN:filespec [/switch(es)]

#### Characteristics:

The 'filespec' is the name of the program, and '/switch(es)' can be any of the following:

1. /RUNCORE:n — Run the program with a memory allocation of nK (K = 1024 words) or nP (P = 1 page or 512 words).
2. /RUNOFFSET:n — Run the program, starting at the address + n. If this switch is omitted, the default value is 0. If the argument 'n' is omitted, the argument defaults to 1. This switch is generally only useful to compilers. Once this switch is used, the argument in effect applies to all subsequent runs, unless /RUNOFFSET is specified again with a new argument.
3. /TMPFIL:nam:'string' — Store the string in TMPCOR file 'nam'. The 'nam' can be no more than three characters long. This switch is useful for passing a command string to a program.

#### 3.4.14 /SUPPRESS Switch

The /SUPPRESS switch indicates to suppress various kinds of messages, depending on which argument you specify. This switch has the following format:

/SUPPRESS:argument

#### Characteristics:

The possible arguments are as follows:

1. NONE — Suppress no messages.
2. INFORMATION — Suppress all information messages (those beginning with 'I').

3. **WARNING** — Suppress all information messages and all warning messages (those beginning with '%').
4. **FATAL** — Suppress all information message, all warning messages, and all fatal error messages (those beginning with '?').
5. **ALL** — Suppress all messages, except messages beginning with '\$'. (This has the same effect as FATAL.)

#### **NOTE**

It is not possible to suppress those messages beginning with '\$', since these are operator intervention messages (for example, to mount the next reel of a multireel tape file).

### **3.4.15 /TEMP Switch**

The /TEMP switch indicates that you wish to specify the names of the devices for temporary file storage. This switch has the following format:

/TEMP

Characteristics:

1. For stand-alone sorts, the number of devices is fixed at 26.
2. For COBOL and FORTRAN-called sorts, the devices you specify depends on the number of I/O channels available to SORT/MERGE.
3. One temporary file is created on each device that you specify. For example, if you specify the following temporary areas:

DSKA:/TEMP,DSKC:/TEMP,DSKB:/TEMP

The runs are appended to the above areas as follows:

DSKA: — /RUN 1/RUN 4/RUN 7/...RUN n  
 DSKC: — /RUN 2/RUN 5/RUN 8/...RUN n + 1  
 DSKB: — /RUN 3/RUN 6/RUN 9/...RUN n + 2

4. If you do not specify this switch, then SORT/MERGE writes its temporary files (up to 26, as needed) on DSK:.

## **3.5 Tape Switches**

Tape switches are used to define attributes of the tape read/write process and to control the tape drive itself.

### **3.5.1 /DENSITY Switch**

The /DENSITY switch indicates the tape density to be used for reading or writing a magnetic tape. This switch has the following format:

/DENSITY:n

Characteristics:

1. The argument 'n' is a decimal number indicating the density and can have one of the following values:
  - 200
  - 556
  - 800
  - 1600
  - 6250
2. The /DENSITY switch is a position dependent switch.
3. The default is installation dependent and can be set with the SET DENSITY command. (See the *TOPS-10 Operating System Commands Manual*.)

### 3.5.2 /INDUSTRY Switch

The /INDUSTRY switch indicates that a magnetic tape is to be read or written in industry-compatible mode. This switch has the following format:

/INDUSTRY

Characteristics:

The /INDUSTRY switch is meaningful only for EBCDIC magnetic tapes.

### 3.5.3 /LABEL Switch

The /LABEL switch indicates the tape label status to SORT/MERGE. This is a modified position dependent switch. This switch has the following format:

/LABEL:argument

Characteristics:

The possible arguments are:

1. ANSI — If a tape label processor is present, ANSI-STANDARD labels are read or written. Otherwise, this is similar to /LABEL:NONSTANDARD. Thus, labels are skipped over on input and omitted on output.
2. DEC — Currently equivalent to /LABEL:STANDARD, but preferred for specification of COBOL-standard labels.
3. IBM — If a tape label processor is present, IBM-STANDARD labels are read or written. Otherwise, this is similar to /LABEL:NONSTANDARD. Thus, labels are skipped over on input and omitted on output.

4. **NONSTANDARD** — The file has nonstandard labels. SORT/MERGE bypasses them on input and omits them on output. These labels are assumed to be the size of one physical record.
5. **OMITTED** — The file has no labels, and SORT/MERGE does not perform any label checks.
6. **STANDARD** — The file has COBOL-standard labels. SORT/MERGE reads and writes them on magnetic tape.

If you do not specify the /LABEL switch, then SORT/MERGE expects:

1. Standard labels for a file on magnetic tape
2. No labels for a file on any other device

#### **NOTE**

Labels are ignored on disk because disk is a directory device.

### **3.5.4 /PARITY Switch**

The /PARITY switch indicates the parity to be used when reading or writing a magnetic tape. This switch has the following format:

/PARITY:argument

Characteristics:

1. This switch is a position dependent switch.
2. The possible arguments are:
  - EVEN
  - ODD
3. The default is ODD parity.

### **3.5.5 /POSITION Switch**

The /POSITION switch indicates that you wish to position the magnetic tape before the file is read or written, but after rewinding (if required). This switch has the following format:

/POSITION:argument

Characteristics:

1. The argument can be a signed positive or negative number. Positive arguments skip that number of files. Negative arguments backspace that number of files.
2. This switch is ignored for devices other than tape drives.
3. The default is to position at the first file on the tape.

### **3.5.6 /REWIND Switch**

The /REWIND switch indicates to rewind the tape before the file is read or written. This switch has the following format:

/REWIND

Characteristics:

1. This switch is ignored for devices other than tape drives.
2. The /REWIND switch is a local switch.

### **3.5.7 /STANDARD Switch**

The /STANDARD switch indicates that the data is to be read and written in STANDARD-ASCII mode. This switch has the following format:

/STANDARD

Characteristics:

1. This switch is a modified position dependent switch.
2. The /STANDARD switch is meaningful only for ASCII recording mode.

### **3.5.8 /UNLOAD Switch**

The /UNLOAD switch indicates to rewind and unload the tape after the file is read or written. This switch has the following format:

/UNLOAD

Characteristics:

1. This switch is ignored for any devices other than tape drives.
2. For multireel files, this switch affects only the disposition of the last tape. All intermediate tapes are UNLOADED regardless of the setting of this switch.
3. The /UNLOAD switch is a local switch.
4. Once the tape is unloaded, the tape cannot be read from or written to unless the operator reloads the tape drive.



## Chapter 4

### File Formats

SORT/MERGE is capable of reading and writing data in four recording modes and in a variety of file formats. The following sections describe the four recording modes and all major file formats that are recognized by SORT/MERGE. Each file format is described in detail, and the COBOL code segments and/or FORTRAN programs that generate each file format are given. In addition, each code segment or program has a detailed illustration of the records it produces, and a description of the SORT/MERGE command strings needed to sort the record on any of its fields.

#### 4.1 Recording Modes

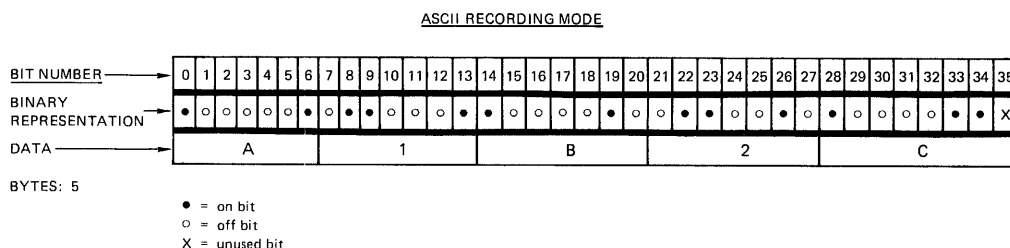
The recording mode specifies the byte size of the data and, except for binary mode, also specifies the character set used. The four recording modes and their respective byte sizes are:

<u>RECORDING MODE</u>	<u>BYTE SIZE</u>
ASCII	7 bits
SIXBIT	6 bits
EBCDIC	8 bits
Binary	36 bits (1 word)

The following sections describe the recording modes in more detail.

### 4.1.1 ASCII Recording Mode

An ASCII word consists of five characters left-justified in the word. Each character is represented by a 7-bit byte:



MR-S-030-79

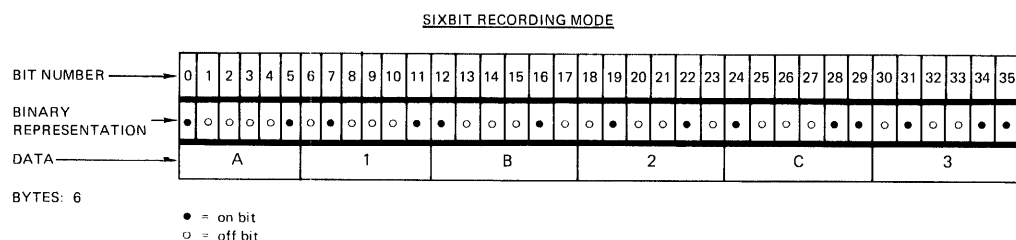
#### NOTE

A variant form of ASCII, line-sequence ASCII, sets bit 35 of the line-sequence word to 1.

ASCII recording mode is specified to SORT/MERGE with the /ASCII switch.

### 4.1.2 SIXBIT Recording Mode

SIXBIT is a compressed form of ASCII in which lowercase letters and a few special characters are not used. A SIXBIT word consists of six characters per word, with each character represented by a 6-bit byte:

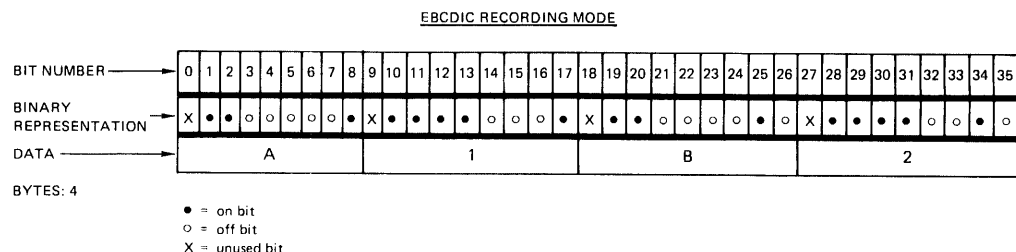


MR-S-031-79

SIXBIT recording mode is specified to SORT/MERGE with the /SIXBIT switch.

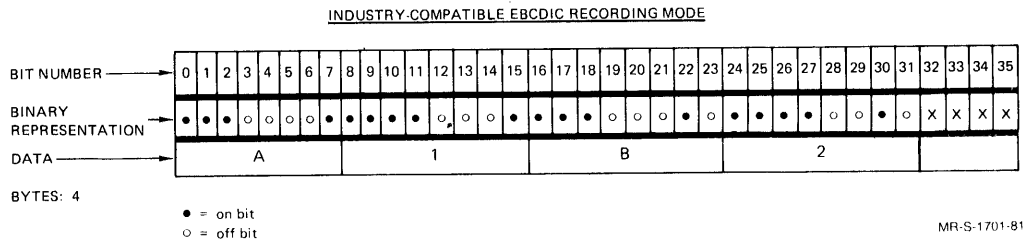
### 4.1.3 EBCDIC Recording Mode

An EBCDIC word consists of four characters per word. Each byte is nine bits long, but the first bit in each byte is unused. Each character is represented by eight bits:



MR-S-1700-81

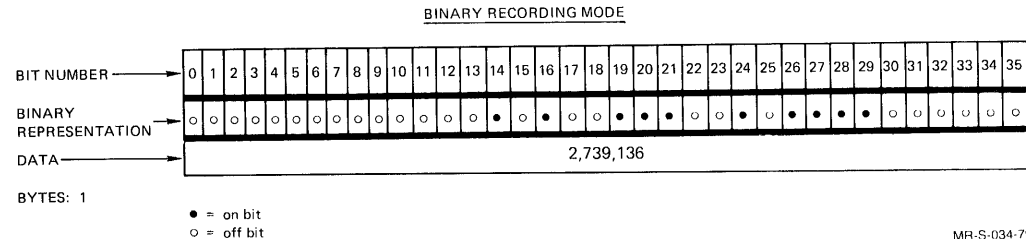
A variant form, used only for magnetic tape, is industry-compatible EBCDIC. In this form of EBCDIC, there are four characters per word, left-justified within the word. Each character is represented by an 8-bit byte. The last four bits in the word are unused:



Standard EBCDIC recording mode is specified to SORT/MERGE with the /EBCDIC switch. Industry-compatible EBCDIC recording mode is specified with the /EBCDIC/INDUSTRY switch combination.

**4.1.4 BINARY Recording Mode**

Unlike the recording modes previously mentioned, binary mode does not specify a character set for the data. In binary mode, the entire 36-bit word is interpreted as a single byte of binary data:



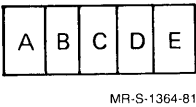
Binary recording mode is specified to SORT/MERGE with the /BINARY switch.

**4.2 File Formats**

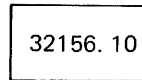
The file format specifies the structure of the record used to store the data. The following sections describe all major file formats recognized by SORT/MERGE. Each section includes a diagram of the file format and a COBOL code segment or FORTRAN program that generates the file format.

Note the following conventions that are used in the diagrams:

1. Alphanumeric or numeric character data in a word is shown with each individual character enclosed in a box. The box represents one byte. Thus, a word of ASCII data is shown as follows:

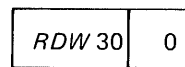


2. Binary data in a word (fixed- and floating-point numbers) is shown by a number in the word:



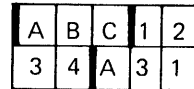
MR-S-1365-81

3. EBCDIC packed-decimal values are shown as two decimal digits per EBCDIC byte. The right half of the rightmost byte contains the sign. Note that neither the digits nor the sign are EBCDIC characters.
4. COBOL signed numeric data, such as produced by PIC S9(n), is shown with the overpunched character, if the sign is negative. For example, -12345 is shown as 1234N, with the N representing both the negative sign and the value 5. DIGITAL's COBOL does not use overpunched characters for positive sign representation, so diagrams depicting positive, signed numeric data do not show a sign. (Note that SORT/MERGE accepts positive overpunched characters.)
5. Italicized characters in a diagram do not depict data; they label or clarify parts of the diagram:



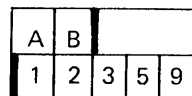
MR-S-1366-81

6. Heavy vertical lines are used to delimit individual fields within a record:



MR-S-1367-81

7. Padding, the use of blanks or nulls to force the next record to begin on some boundary (for example, a word or disk-block boundary), is shown by white space in the word:

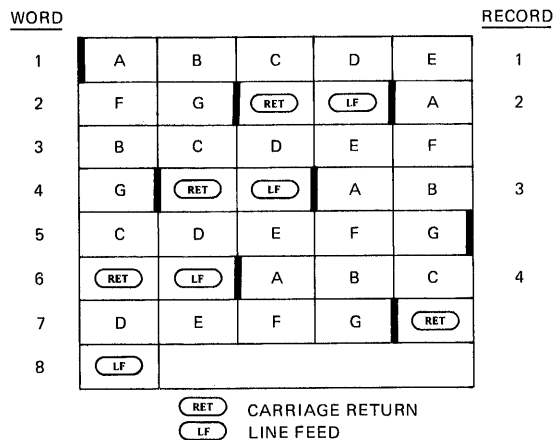


MR-S-1368-81

Note that you cannot consider padding as part of a record field, nor can you use padding as part of a key field. However, the length of any padding must be taken into account when calculating record length and key starting position.

## 4.2.1 Fixed-Length ASCII

A fixed-length ASCII file consists of records containing five characters per 36-bit word, with each group of five characters left-justified within the word. Fixed-length ASCII records must end with a carriage-return/line-feed. The following diagram illustrates the format of fixed-length ASCII records:



MR-S-035-79

The format is specified with the following SORT/MERGE switch combination:

/ASCII/FIXED

### 4.2.1.1 COBOL Fixed-Length ASCII —

CODE SEGMENT:

```

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

SELECT filename ASSIGN TO DSK
                  RECORDING MODE IS ASCII.

DATA DIVISION.
FILE SECTION.

FD filename      VALUE OF ID "DATA  FIL".
01 record-1      DISPLAY-7.
   02 field-1     PIC X(6) VALUE "AB12EF".
   02 field-2     PIC A(3) VALUE "GHI".
   02 field-3     PIC 9(4) VALUE 3249.
   02 field-4     PIC S9(6) VALUE -481253.
   02 field-5     PIC S9(6)V9999 VALUE +31458.5012.
  
```

Figure 4-1 illustrates the record produced by the code segment shown above:

**Figure 4-1: COBOL Fixed-Length ASCII**

WORD						
1	A	B	1	2	E	/KEY:1:6 /ALPHA
2	F	G	H	I	3	/KEY:7:3 /ALPHA
3	2	4	9	4	8	/KEY:10:4 /NUMERIC/UNSIGNED
4	1	2	5	L	0	/KEY:14:6 /NUMERIC/SIGNED
5	3	1	4	5	8	/KEY:20:10 /NUMERIC/SIGNED
6	5	0	1	2	RET	
7	LF					

MR-S-1728-81

**SORT/MERGE Command String:**

\*SORTED,FIL=DAT,FIL/ASCII/FIXED/RECORD:29/KEY:1:6/ALPHA(RET)

### NOTE

To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4-1. Also, to retain the FIXED format of the file shown in the example, no ADVANCING clauses on WRITE statements can be used in the COBOL code.

#### 4.2.1.2 FORTRAN Fixed-Length (Random) ASCII —

PROGRAM:

```

DIMENSION A(2),C(2)
DOUBLE PRECISION Y
INTEGER J
REAL X

A(1)='ABCDE'; A(2)='F12'
B  ='LMNO'
C(1)='12345'; C(2)='6789'
D  ='-1234'; I=1
J  =1234567890
X  =123456.4321
Y  =-1435789432.456
OPEN (UNIT=1,DEVICE='DSK',FILE='TEST.DAT',MODE='ASCII',
1ACCESS='RANDOM',RECORDSIZE=68)
WRITE (1#I,100)A,B,C,D,J,X,Y
100  FORMAT(1A5,1A3,1A4,1A5,1A4,1A5,I10,E17.8,D15.4)
CLOSE (UNIT=1)
END

```

Figure 4-2 illustrates the record produced by the program shown above:

**Figure 4-2: FORTRAN Fixed-Length (Random) ASCII**

WORD						
1	A	B	C	D	E	/KEY:1:8 /ALPHA
2	F	1	2	L	M	/KEY:9:4 /ALPHA
3	N	0	1	2	3	/KEY:13:9 /NUMERIC/UNSIGNED
4	4	5	6	7	8	
5	9	-	1	2	3	/KEY:22:5 /NUMERIC/SIGNED
6	4	1	2	3	4	/KEY:27:10 /NUMERIC/UNSIGNED
7	5	6	7	8	9	
8	0				0	/KEY:37:17 /FORMAT:E17.8
9	.	1	2	3	4	
10	5	6	4	3	E	
11	+	0	6			/KEY:54:15 /FORMAT:D15.4
12			-	0	.	
13	1	4	3	6	0	
14	+	1	0	MI	LI	

= blank

MR-S-1729-81

**SORT/MERGE Command String:**

\*SORTED,FIL=TEST,DAT/FORTRAN/ASCII/RANDOM/RECORD:68/KEY:1:8/ALPHA(RET)

FORTRAN users can describe this file format with either of the following SORT/MERGE switch combinations:

1. /FORTRAN/ASCII/RANDOM
2. /FORTRAN/ASCII/FIXED

**NOTE**

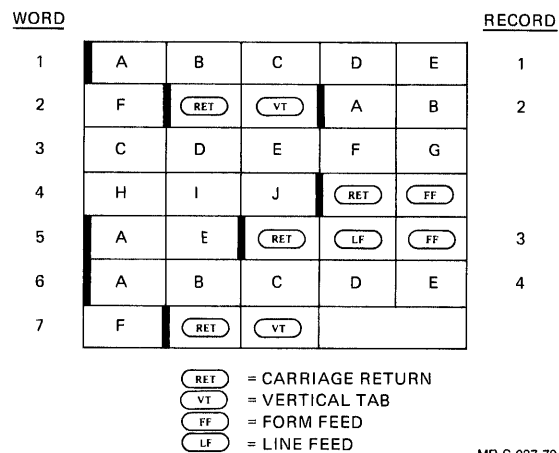
To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4-2.

### 4.2.2 Variable–Length ASCII

Variable–length ASCII consists of records containing five characters per 36–bit word, with each group of five characters left–justified within the word. Variable–length ASCII records must end with some combination of the following:

1. carriage return
2. line feed
3. vertical tab
4. form feed

The following diagram illustrates the format of variable–length ASCII records:



SORT/MERGE strips away all end–of–line characters and replaces them with carriage–return/line–feed on output.

This format is specified with any one of the following SORT/MERGE switch combinations:

1. /ASCII/VARIABLE
2. /ASCII(/VARIABLE in effect by default)



#### 4.2.2.1 COBOL Variable-Length ASCII —

CODE SEGMENT:

```

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

SELECT filename ASSIGN TO DSK
      RECORDING MODE IS ASCII.

DATA DIVISION.
FILE SECTION.

FD filename          VALUE OF ID "DATA  FIL".
01 record-1          DISPLAY-7.
   02 field-1         PIC X(7) VALUE "AB13521".
   02 field-2         PIC S9(7)V99 VALUE -3269.02.
   02 field-3         PIC A(3) VALUE "ILM".
   02 field-4         PIC 9(4) VALUE 1359.

01 record-2          DISPLAY-7.
   02 field-1         PIC X(7) VALUE "EFGHI95".
   02 field-2         PIC S9(7)V99 VALUE 42553.40.
   02 field-3         PIC A(3) VALUE "LMN".
   02 field-4         PIC 9(7) VALUE 3712536.

PROCEDURE DIVISION.

WRITE record-1 BEFORE ADVANCING.
WRITE record-2 BEFORE ADVANCING.

```

Figure 4–3 illustrates the record produced by the code segment shown above:

**Figure 4–3: COBOL Variable-Length ASCII**

WORD

1	A	B	1	3	5	/KEY:1:7/ALPHA
2	2	1	0	3	2	/KEY:8:7/NUMERIC/SIGNED
3	6	9	0	K	I	/KEY:15:3/ALPHA
4	L	M	1	3	5	
5	9	RI	II	E	F	
6	G	H	I	9	5	
7	4	2	5	5	3	
8	4	0	L	M	N	
9	3	7	1	2	5	
10	3	6	RI	II		

NOTE

It is unwise to specify a key that extends outside the shorter records in a variable length file, unless the data type is ALPHA-NUMERIC. For other data types, the result is undefined and will result in a warning message. For this reason, a key specification is not given for the last field in the record.

#### NOTE

It is unwise to specify a key that extends outside the shorter records in a variable length file, unless the data type is ALPHA-NUMERIC. For other data types, the result is undefined and will result in a warning message. For this reason, a key specification is not given for the last field in the record.

MR-S-1730-81

## SORT/MERGE Command String:

```
*SORTED,FIL=DATA,FIL/ASCII/VARIABLE/RECORD:47/KEY:1:7/ALPHA(RET)
```

### NOTE

To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4–3.

#### 4.2.2.2 FORTRAN Variable–Length (Sequential) ASCII —

PROGRAM:

```

      DIMENSION A(2),C(2)
      DOUBLE PRECISION Y
      INTEGER J
      REAL X

      A(1)='ABCDE'; A(2)='F12'
      B  ='-1234'
      J  =1234567890
      X  =123456.4321
      Y  =-1435789432.456
      C(1)='12345'
      OPEN (UNIT=1,DEVICE='DSK',FILE='TEST1.DAT',MODE='ASCII',
1ACCESS='SEQOUT')
      WRITE (1,100)A,B,J,X,Y,C(1)
100  FORMAT(1A5,1A3,1A5,I10,E9.2,D12.4,1A5)

      A(1)='LMNOP'; A(2)='22C'
      B  =' +3500'
      J  =4567912343
      X  =4569.723
      Y  =+45982341234.234
      C(1)='54402'; C(2)='6789'
      WRITE (1,200)A,B,J,X,Y,C
200  FORMAT(1A5,1A3,1A5,I10,E9.2,D12.4,1A5,1A4)
      CLOSE (UNIT=1)
      END
```

Figure 4–4 illustrates the record produced by the program shown above:

**Figure 4-4: FORTRAN Variable-Length (Sequential) ASCII**

WORD						
1	A	B	C	D	E	/KEY:1:8 /ALPHA
2	F	1	2	—	1	/KEY:9:5 /NUMERIC/SIGNED
3	2	3	4	1	2	/KEY:14:10 /NUMERIC/SIGNED
4	3	4	5	6	7	
5	8	9	0	␣	0	/KEY:24:9 /FORTRAN:E9.2
6	.	1	2	E	+	
7	0	6	␣	—	0	/KEY:33:12 /FORMAT:D12.4
8	.	1	4	3	6	
9	0	+	1	0	1	
10	2	3	4	5	RET	
11	IF					
12	L	M	N	O	P	
13	2	2	C	+	3	
14	5	0	0	4	5	
15	6	7	9	1	2	
16	3	4	3	␣	O	
17	.	4	6	E	4	
18	0	4	␣	␣	0	
19	.	4	5	9	8	
20	D	+	1	1	5	
21	4	4	0	2	6	
22	7	8	9	RET	IF	

␣ = blank

**NOTE**  
It is unwise to specify a key that extends outside the shorter records in a variable-length file, unless the data type is ALPHA-NUMERIC. For other data types, the result is undefined and will result in a warning message. For this reason, a key specification is not given for the last field in the record.

MR-S-1731-81

**SORT/MERGE Command String:**

```
*SORTED,FIL=DATA,FIL/FORTRAN/ASCII/SEQUENTIAL -(RET)
#/RECORD:49/KEY:1:8/ALPHA(RET)
```

FORTTRAN users can specify this format with any one of the following SORT/MERG switch combinations:

1. /FORTRAN/ASCII/SEQUENTIAL
2. /FORTRAN/ASCII/VARIABLE
3. /FORTRAN/ASCII/(VARIABLE in effect by default)

## NOTE

To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4-4.

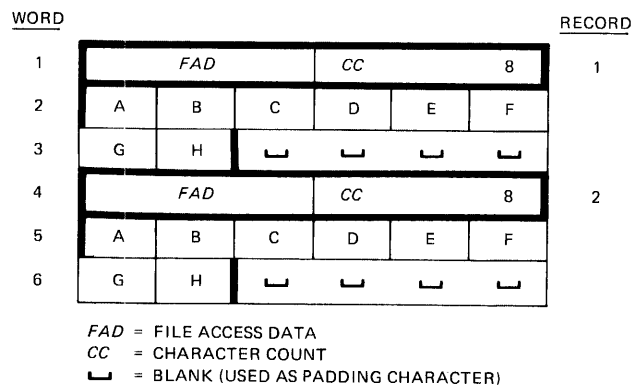
### 4.3 Fixed-Length SIXBIT

In a SIXBIT file, characters are stored six per 36-bit word, and a SIXBIT record must start and end on a word boundary. The left half of the first word in the record contains one of the following:

1. The record sequence number of COBOL magnetic tape records
2. Data specific to COBOL ISAM records (ISAM files cannot be directly sorted by SORT/MERGE.)
3. Binary zeros

The right half of the first word contains the number of characters in the record. To ensure that the record ends on a word boundary, the last word in the record is padded with blanks, if necessary. When determining the size of the record for memory considerations, you must take into account the first word of the record (containing file-access information and a character count) and the possible existence of padding characters (blanks) to enable the record to end on a word boundary.

The following diagram illustrates the format of fixed-length SIXBIT records. Note that the character count is the same for each record:



MR-S-039-79

This format is specified by the following SORT/MERGE switch combination:

/SIXBIT/FIXED

### 4.3.1 COBOL Fixed-Length SIXBIT

CODE SEGMENT:

```

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

SELECT filename ASSIGN TO DSK
      RECORDING MODE IS SIXBIT.

DATA DIVISION.
FILE SECTION.

FD filename          VALUE OF ID "DATA  FIL".
01 record-1          DISPLAY-6.
   02 field-1        PIC X(4) VALUE "A13B".
   02 field-2        PIC A(5) VALUE "CDEFG".
   02 field-3        PIC 9(10) COMP VALUE 9654839218.
   02 field-4        PIC X(2) VALUE "HI".
   02 field-5        PIC 9(11) COMP VALUE 34567982314.
   02 field-6        PIC 9(4) VALUE 1289.
   02 field-7        PIC 9(5) COMP-1 VALUE 123.45.
   02 field-8        PIC 9(11) COMP VALUE 12398756983.
  
```

Figure 4-5 illustrates the record produced by the code segment shown above:

**Figure 4-5: COBOL Fixed-Length SIXBIT**

WORD

	FAD			CC			60		
1	A	1	3	B	C	D		/KEY:1:4 /ALPHA	
2	E	F	G						/KEY:5:5 /ALPHA
3	9654839218							/KEY:13:10 /COMP	
4	H	I							/KEY:19:2 /ALPHA
5	34567982314							/KEY:25:11 /COMP	
6									
7	1	2	8	9					/KEY:37:4 /NUMERIC/UNSIGNED
8	123.45							/KEY:43:5 /COMP1	
9	12398756983							/KEY:49:11 /COMP	
10									

MB-S-1732-B1

MR-S-1732-81

**SORT/MERGE Command String:**

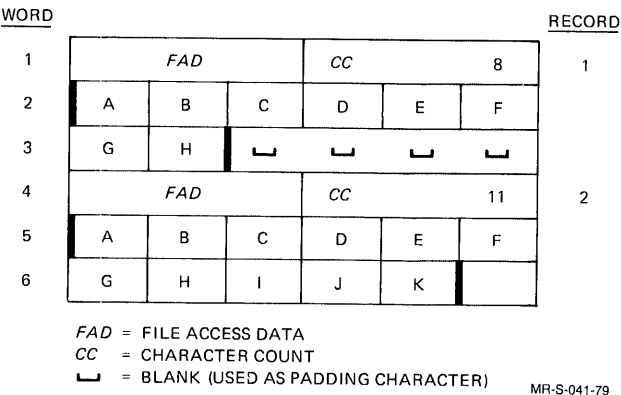
\*SORTED.FIL=DATA.FIL/SIXBIT/FIXED/RECORD:60/KEY:1:4/ALPHA(RET)

#### NOTE

To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4-5.

# 4.4 Variable–Length Sixbit

This format is the same as fixed–length SIXBIT, except that the character count can vary from record to record. The following diagram illustrates the format of variable–length SIXBIT records:



This format is specified by either of the following SORT/MERGE switch combinations:

- 1. /SIXBIT/VARIABLE
- 2. /SIXBIT(/VARIABLE is in effect by default)

#### 4.4.1 COBOL Variable-Length SIXBIT

CODE SEGMENT:

```
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
  
SELECT filename ASSIGN TO DSK  
RECORDING MODE IS SIXBIT.  
  
DATA DIVISION.  
FILE SECTION.  
  
FD filename          VALUE OF ID "DATA  FIL".  
01 record-1          DISPLAY-6.  
    02 field-1        PIC 9(7) COMP-1 VALUE 123.4567.  
    02 field-2        PIC X(3) VALUE "A3C".  
    02 field-3        PIC A(3) VALUE "DEF".  
    02 field-4        PIC 9(3) VALUE -55.  
    02 field-5        PIC 9(10) COMP VALUE 1234567809.  
    02 field-6        PIC 9(11) COMP VALUE 98765432108.  
    02 field-7        PIC X(2) VALUE "A2".  
    02 field-8        PIC 9(5) COMP VALUE 32571.  
  
01 record-2          DISPLAY-6.  
    02 field-1        PIC 9(7) COMP-1 VALUE 1395.678.  
    02 field-2        PIC X(3) VALUE "B5L".  
    02 field-3        PIC A(3) VALUE "LMN".  
    02 field-4        PIC 9(3) VALUE 79.  
    02 field-5        PIC 9(10) COMP VALUE 8176596821.  
    02 field-6        PIC 9(11) COMP VALUE 18976532150.  
    02 field-7        PIC X(2) VALUE "M5".  
    02 field-8        PIC 9(11) COMP VALUE 12357986183.  
  
PROCEDURE DIVISION.  
  
    WRITE record-1.  
    WRITE record-2.
```

Figure 4-6 illustrates the record produced by the code segment shown above:

Figure 4–6: COBOL Variable–Length SIXBIT

WORD

	FAD			CC			48
1	123.4567						/KEY:1:7 /COMP1
2	A	3	C	D	E	F	/KEY:7:3 /ALPHA
3	0	5	N				/KEY:10:3 /ALPHA
4	1234567809						/KEY:13:3 /NUMERIC/SIGNED
5	98765432108						/KEY:19:10 /COMP
6							/KEY:25:11 /COMP
7	A	2					/KEY:37:2 /ALPHA
8	32571						
	FAD			CC			54
1	1395.678						
2	B	5	L	L	M	N	
3	0	7	9				
4	8176596821						
5	18976532150						
6							
7	M	5					
8	12357986183						
9							

NOTE

It is unwise to specify a key that extends outside the shorter records in a variable-length file, unless the data type is ALPHA-NUMERIC. For other data types, the result is undefined and will result in a warning message. For this reason, a key specification is not given for the last field in the record.

MR-S-1733-8

MR-S-1733-81

SORT/MERGE Command String:

\*SORTED,FIL=DATA,FIL/SIXBIT/VARIABLE/RECORD:48/KEY:1:7/COMP1(RET)

NOTE

To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4–6.

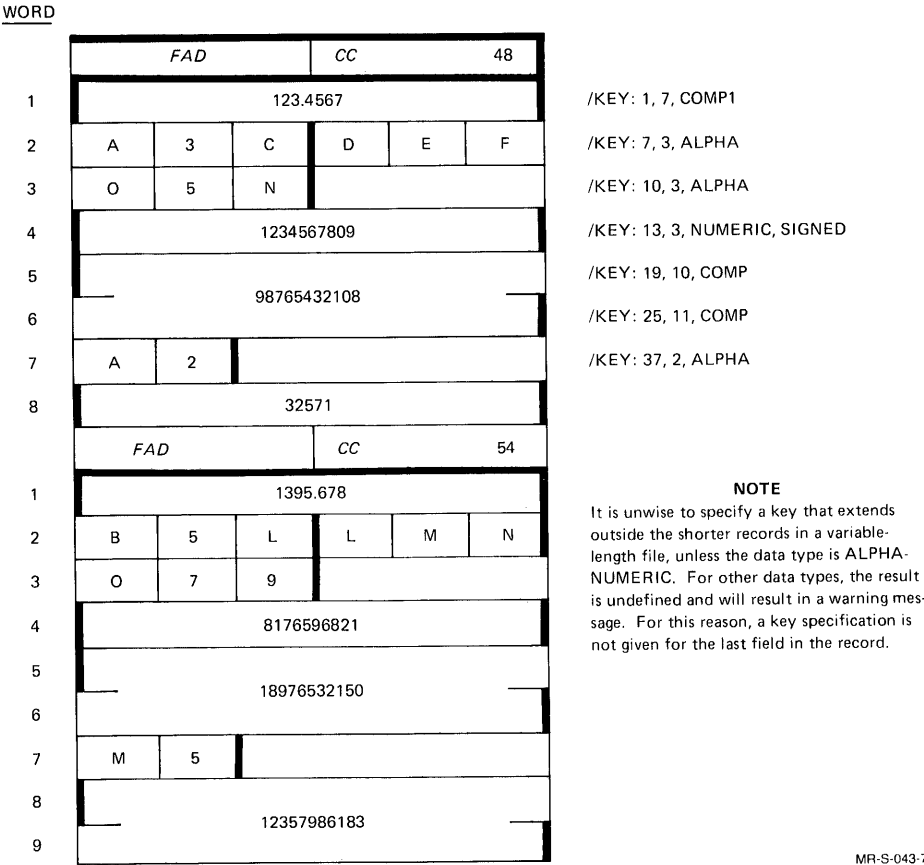


# 4.5 EBCDIC File Formats

On disk and in memory, the characters in an EBCDIC file are represented by 8 bits right-justified in 9-bit bytes. On tape, the characters in an EBCDIC file are represented by 8-bit bytes, and 4 bytes occur per 36-bit word. EBCDIC records are written only by COBOL programs.

## 4.5.1 COBOL Fixed-Length EBCDIC

Within a given file, fixed-length EBCDIC records all have the same record length, and the record need not begin or end on a word boundary. The following diagram illustrates the format of fixed-length EBCDIC records:



The file format is specified by the following SORT/MERGE switch combination:

/EBCDIC/FIXED

#### CODE SEGMENT:

```
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
  
SELECT filename ASSIGN TO DSK  
RECORDING MODE IS F.  
  
DATA DIVISION.  
FILE SECTION.  
  
FD filename VALUE OF ID "DATA FIL".  
01 record-1 DISPLAY-9.  
02 field-1 PIC 9(3) VALUE 123.  
02 field-2 PIC X(5) VALUE "ABCDE".  
02 field-3 PIC A(2) VALUE "LM".  
02 field-4 PIC 9(9) COMP-3 VALUE 137958795.  
02 field-5 PIC S9(6) COMP-3 VALUE -351235.
```

Figure 4–7 illustrates the record produced by the code segment shown above:

**Figure 4–7: COBOL Fixed–Length EBCDIC**

WORD					
1	1	2	3	A	/KEY:1:3 /NUMERIC/UNSIGNED
2	B	C	D	E	/KEY:4:5 /ALPHA
3	L	M	1 3	7 9	/KEY:9:2 /ALPHA
4	5 8	7 9	5 +	3	/KEY:11:9 /COMP3/UNSIGNED
5	5 1	2 3	5 -		/KEY:16:6 /COMP3/SIGNED

MR-S-1734-81

#### SORT/MERGE Command String:

```
*SORTED,FIL=DATA,FIL/EBCDIC/FIXED/RECORD:19/KEY:1:3/ALPHA(RET)
```

#### NOTE

To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4–7.

### 4.5.2 COBOL Variable–Length EBCDIC

In this file format, the record lengths can vary from record to record. Each record contains a 4–byte Record Descriptor Word (RDW) at the head of the record. The left half word of the RDW specifies a value equal to the number of bytes in the record plus 4 (to allow for the length of the RDW itself). The rightmost two bytes of the RDW must be zero. If they are nonzero, they indicate spanned records, which are unsupported. The following diagram illustrates the format of variable–length EBCDIC records:

WORD					RECORD
1	RDW		12	0	1
2	A	B	C	D	
3	E	F	G	H	
4	RDW		16	0	2
5	A	B	C	D	
6	E	F	G	H	
7	I	J	K	L	3
8	RDW		12	0	
9	A	B	C	D	
10	E	F	G	H	

RDW = RECORD DESCRIPTOR WORD

MR-S-045-79

This format is specified by the following SORT/MERGE switch combination:

/EBCDIC/VARIABLE

CODE SEGMENT:

```

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

SELECT filename ASSIGN TO DSK
    RECORDING MODE IS V.

DATA DIVISION.
FILE SECTION.

FD filename          VALUE OF ID "DATA  FIL".
01 record-1          DISPLAY-9.
    02 field-1        PIC S9(7) COMP-3 VALUE -1398569.
    02 field-2        PIC S9(8) COMP-3 VALUE 57635937.
    02 field-3        PIC 9(3) VALUE 596.
    02 field-4        PIC A(2) VALUE "AB".
    02 field-5        PIC X(5) VALUE "A13DE".

01 record-2          DISPLAY-9.
    02 field-1        PIC S9(7) COMP-3 VALUE 5369787.
    02 field-2        PIC S9(8) COMP-3 VALUE -53896156.
    02 field-3        PIC 9(3) VALUE 593.
    02 field-4        PIC A(2) VALUE "MN".
    02 field-5        PIC X(8) VALUE "ILH5MLXY".

PROCEDURE DIVISION.

    WRITE record-1.
    WRITE record-2.

```

Figure 4-8 illustrates the record produced by the code segment shown above:

# CODE SEGMENT:

ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.

SELECT filename ASSIGN TO DSK  
RECORDING MODE IS F.

DATA DIVISION.  
FILE SECTION.

FD filename VALUE OF ID "DATA FIL"  
BLOCK CONTAINS 1 RECORDS.

01 record-1 DISPLAY-9.  
02 field-1 PIC 9(3) VALUE "194".  
02 field-2 PIC X(5) VALUE "BDEFG".  
02 field-3 PIC A(2) VALUE "MN".  
02 field-4 PIC 9(5) COMP-3 VALUE 13796.  
02 field-5 PIC S9(4) COMP-3 VALUE 1985.

01 record-2 DISPLAY-9.  
02 field-1 PIC X(3) VALUE "762".  
02 field-2 PIC X(5) VALUE "LANBH".  
02 field-3 PIC A(2) VALUE "AB".  
02 field-4 PIC 9(5) COMP-3 VALUE 76543.  
02 field-5 PIC S9(4) COMP-3 VALUE -9764.

PROCEDURE DIVISION.

WRITE record-1.  
WRITE record-2.

Figure 4-9 illustrates the record produced by the code segment shown above:

**Figure 4-9: COBOL Blocked Fixed-Length EBCDIC**

WORD				BLOCK	
<u>BLOCK 1</u>				1	/KEY:1:3/NUMERIC/UNSIGNED
1	1	9	4	B	
2	D	E	F	G	/KEY:4:5/ALPHA
3	M	N	1 3	7 9	/KEY:9:2/ALPHA
4	6 +	1	9 8	5 +	/KEY:11:5/COMP3/UNSIGNED
					/KEY:14:4/COMP3/UNSIGNED
<u>BLOCK 2</u>				2	
1	7	6	2	L	
2	A	N	B	H	
3	A	B	7 6	5 4	
4	3 +	9	7 6	4 -	

MR-S-1736-81

## SORT/MERGE Command String:

```
*SORTED,FIL=DATA,FIL/EBCDIC/FIXED/BLOCKED:1/RECORD:16 -(RET)
*/KEY:1:3/NUMERIC/SIGNED(RET)
```

### NOTE

To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4–9.

### 4.5.4 COBOL Blocked Variable–Length EBCDIC

In this file format, the record length can vary from record to record. Each record contains a 1–word Record Descriptor Word (RDW) at the head of the record. This word contains (in the left half word) a count of all bytes in the record and in the RDW itself. The right half of the RDW must be zero. The records are read and written in groups called blocks. The actual number of records in a block depends on the blocking factor specified when the file was created. Each block of a record contains a 1–word Block Descriptor Word (BDW) which contains a count (in the left half word) of the bytes in the block. The bytes of data, the bytes of the RDW for each record in the block, and the four bytes of the BDW itself are included in the block count. The following illustrates the format of blocked variable–length EBCDIC records:

WORD	RECORD				BLOCK
1	BDW 20		0		1
2	RDW 10		0		1
3	A	B	C	D	
4	E	F	RDW 6		2
5	0	0	A	B	
201	BDW 28		0		2
202	RDW 6		0		3
203	A	B	RDW 10		4
204	0		A	B	
205	C	D	E	F	
206	RDW 8		0		5
207	A	B	C	D	

BDW = BLOCK DESCRIPTOR WORD  
RDW = RECORD DESCRIPTOR WORD

MR-S-048-79

This format is specified with the following SORT/MERGE switch combinations:

1. /EBCDIC/VARIABLE/BLOCKED:n
2. /EBCDIC/BLOCKED:n(/VARIABLE is in effect by default)

#### NOTE

SORT/MERGE accepts this format from disk only.  
SORT/MERGE does not correctly sort blocked  
variable-length EBCDIC files on tape.

CODE SEGMENT:

```
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
  
SELECT filename ASSIGN TO DSK  
RECORDING MODE IS V.  
  
DATA DIVISION.  
FILE SECTION.  
  
FD filename          VALUE OF ID "DATA  FIL"  
BLOCK CONTAINS 1 RECORDS.  
01 record-1          DISPLAY-9.  
  02 field-1          PIC S9(7) COMP-3 VALUE +9356127.  
  02 field-2          PIC 9(7) COMP-3 VALUE 3987156.  
  02 field-3          PIC X(3) VALUE "198".  
  02 field-4          PIC A(2) VALUE "MN".  
  02 field-5          PIC S9(9) COMP-3 VALUE -569138279.  
  02 field-6          PIC X(6) VALUE "ABCDEF".  
  
01 record-2          DISPLAY-9.  
  02 field-1          PIC S9(7) COMP-3 VALUE -3295865.  
  02 field-2          PIC 9(7) COMP-3 VALUE 9378518.  
  02 field-3          PIC X(3) VALUE "196".  
  02 field-4          PIC A(2) VALUE "AL".  
  02 field-5          PIC 9(9) COMP-3 VALUE 569138279.  
  02 field-6          PIC X(9) VALUE "ABCDEFGHI".  
  
PROCEDURE DIVISION.  
  
  WRITE record-1.  
  WRITE record-2.
```

Figure 4-10 illustrates the record produced by the code segment shown above:

Figure 4–10: COBOL Blocked Variable–Length EBCDIC

WORD

BLOCK

	BDW	32		0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											</
--	-----	----	--	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

MR-S-1737-81

**SORT/MERGE Command String:**

\*SORTED,FIL=DATA,FIL/EBCDIC/VARIABLE/BLOCKED:1/RECORD:27 -(RET)  
\*/KEY:1:7/COMP3/SIGNED(RET)

**NOTE**

To sort the file on the various fields of the record, replace the shaded portion of the command string with one or more of the key specifications shown in Figure 4–10.

## 4.6 Binary File Formats

Binary records consist of contiguous 36-bit words. Each record starts and ends on a word boundary. Binary is the only recording mode which does not have a character set associated with it, and standard binary records can only be interpreted as COMPUTATIONAL and COMP1 binary numbers. However, it is possible to associate a character set with binary records by writing mixed-mode records. These records are specified with one of the following SORT/MERGE switch combinations:

1. /BINARY/ASCII
2. /BINARY/SIXBIT
3. /BINARY/EBCDIC

These formats are discussed in the following sections. Note that for each file format diagram, two sets of key specifications and command strings are given. The first set illustrates sorting the file in standard binary, and the second set illustrates sorting the file in mixed-mode binary.

### 4.6.1 COBOL Binary File Formats

COBOL programs are capable of writing all three mixed-mode binary formats. Each format is discussed below.

#### 4.6.1.1 COBOL ASCII Mixed-Mode Binary —

CODE SEGMENT:

```
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
  
SELECT filename ASSIGN TO DSK  
                  RECORDING MODE IS BINARY.  
  
DATA DIVISION.  
FILE SECTION.  
  
FD filename      VALUE OF ID "DATA  FIL".  
01 BINARY-REC    DISPLAY-7.  
   02 field-1     PIC S9(10) COMP VALUE 12345678910.  
   02 field-2     PIC S9(10) COMP-1 VALUE 1246.597892.  
   02 field-3     PIC X(7) VALUE "ABCDE12".  
   02 field-4     PIC 9(11) COMP VALUE 12345678954.  
   02 field-5     PIC 9(3) VALUE "532".  
   02 field-6     PIC 9(14) COMP VALUE 12345678954.  
   02 field-7     PIC A(2) VALUE "LM".
```

Figure 4-11 illustrates the record produced by the code segment shown above:



Figure 4-11: COBOL Standard Binary and ASCII Mixed-Mode Binary

STANDARD BINARY

/BINARY

ASCII MIXED-MODE BINARY

/BINARY/ASCII

WORD

1

1234568910

/KEY:1:10 /COMP/SIGNED

2

1246.597892

/KEY:6:10 /COMP1/SIGNED

CAN'T BE SORTED AS CHARACTERS

3

A

B

C

D

E

/KEY:11:7 /ALPHA

4

1

2

/KEY:5:2 /UNSIGNED

5

12345678954

/KEY:21:11 /COMP/UNSIGNED

6

CAN'T BE SORTED AS CHARACTERS

7

5

3

2

/KEY:31:3 /NUMERIC/UNSIGNED

/KEY:8:2/UNSIGNED

8

12345678954967

/KEY:36:14 /COMP/UNSIGNED

9

CAN'T BE SORTED AS CHARACTERS

10

L

M

/KEY:46:2 /ALPHA

MR-S-1738-81

MR-S-1738-81

SORT/MERGE Command Strings:

Standard Binary:

\*SORTED,FIL=DATA,FIL/BINARY/RECORD:10/KEY:1:1/SIGNED(RET)

Mixed-Mode Binary:

\*SORTED,FIL=DATA,FIL/BINARY/ASCII/RECORD:47 -(RET)  
#/KEY:1:10/COMP1/SIGNED(RET)

NOTE

To sort the file on the various fields of the record, replace the shaded portion of the standard binary command string with one or more of the key specifications given on the left-hand side of Figure 4-11. Likewise, replace the shaded portion of the mixed-mode binary command string with one or more of the key specifications given on the right-hand side of Figure 4-11.

#### 4.6.1.2 COBOL SIXBIT Mixed-Mode Binary —

CODE SEGMENT:

```

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

SELECT filename ASSIGN TO DSK
      RECORDING MODE IS BINARY.

DATA DIVISION.
FILE SECTION.

FD filename          VALUE OF ID "DATA  FIL".
01 BINARY-REC        DISPLAY-6.
   02 field-1         PIC S9(10) COMP VALUE 12345678910.
   02 field-2         PIC S9(10) COMP-1 VALUE 1234.592175.
   02 field-3         PIC X(7) VALUE "ABCDE12".
   02 field-4         PIC 9(11) COMP VALUE 12345678954.
   02 field-5         PIC X(3) VALUE "532".
   02 field-6         PIC 9(14) COMP VALUE 12345678954.
   02 field-7         PIC A(2) VALUE "LM".
  
```

Figure 4-12 illustrates the record produced by the code segment shown above:

**Figure 4-12: COBOL Standard Binary and SIXBIT Mixed-Mode Binary**

STANDARD BINARY /BINARY		SIXBIT MIXED-MODE BINARY /BINARY/SIXBIT	
		WORD	
/KEY:1:1/SIGNED	1	12345678910	/KEY:1:10 /COMP/UNSIGNED
/KEY:2:1/SIGNED	2	1234.592175	/KEY:7:10 /COMP1/SIGNED
CAN'T BE SORTED AS CHARACTERS	3	A B C D E 1	/KEY:13:7 /ALPHA
	4	2	
/KEY:5:2/UNSIGNED	5	12345678954	/KEY:25:11 /COMP/UNSIGNED
	6		
CAN'T BE SORTED AS CHARACTERS	7	5 3 2	/KEY:37:3 /NUMERIC/UNSIGNED
/KEY:8:2/UNSIGNED	8	12345678954967	/KEY:43:14 /COMP/UNSIGNED
	9		
CAN'T BE SORTED AS CHARACTERS	10	L M	/KEY:55:2 /ALPHA

MR-S-1739-81

**SORT/MERGE Command Strings:**

**Standard Binary:**

```

*SORTED.FIL=DATA.FIL/BINARY/RECORD:10 -RET
*/KEY:5:2/UNSIGNED(RET)
  
```

## Mixed-Mode Binary:

```
*SORTED,FIL=DATA,FIL/BINARY/SIXBIT/RECORD:56 -(RET)  
*/KEY:25:11/COMPU/UNSIGNED(RET)
```

### NOTE

To sort the file on the various fields of the record, replace the shaded portion of the standard binary command string with one or more of the key specifications given on the left-hand side of Figure 4-12. Likewise, replace the shaded portion of the mixed-mode binary command string with one or more of the key specifications given on the right-hand side of Figure 4-12.

#### 4.6.1.3 COBOL EBCDIC Mixed-Mode Binary —

CODE SEGMENT:

```
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
  
SELECT filename ASSIGN TO DSK  
RECORDING MODE IS BINARY.  
  
DATA DIVISION.  
FILE SECTION.  
  
FD filename VALUE OF ID "DATA FIL".  
01 BINARY-REC DISPLAY-9.  
02 field-1 PIC S9(10) COMP VALUE 12345678910.  
02 field-2 COMP-1 VALUE 1246.597861.  
02 field-3 PIC X(7) VALUE "ABCDE12".  
02 field-4 PIC 9(11) COMP VALUE 12345678954.  
02 field-5 PIC 9(3) VALUE "532".  
02 field-6 PIC 9(14) COMP VALUE 12345678954967.  
02 field-7 PIC A(2) VALUE "LM".  
02 field-8 PIC S9(5) COMP-3 VALUE -72539.  
02 field-9 PIC 9(8) COMP-3 VALUE 36193586.
```

Figure 4-13 illustrates the record produced by the code segment shown above:

**Figure 4–13: COBOL Standard Binary and EBCDIC Mixed–Mode Binary**

STANDARD BINARY /BINARY	WORD	EBCDIC MIXED-MODE BINARY /BINARY/EBCDIC
/KEY:1:1/SIGNED	1 12345678910	/KEY:1:10 /COMP/SIGNED
/KEY:2:1/SIGNED	2 1246.597861	/KEY:5:10 /COMP1/SIGNED
CAN'T BE SORTED AS CHARACTERS	3 A B C D	/KEY:9:7 /ALPHA
	4 E 1 2	
/KEY:5:2 /UNSIGNED	5 12345678954	/KEY:17:11 /COMP/UNSIGNED
	6	
CAN'T BE SORTED AS CHARACTERS	7 5 3 2	/KEY:25:3 /NUMERIC/UNSIGNED
/KEY:8:2 /UNSIGNED	8 12345678954967	/KEY:29:14 /COMP/UNSIGNED
	9	
CAN'T BE SORTED AS CHARACTERS	10 L M 7 2 5 3	/KEY:37:2 /ALPHA
CAN'T BE SORTED AS CHARACTERS	11 9 - 3 6 1 9 3	/KEY:39:5 /COMP3/UNSIGNED
CAN'T BE SORTED AS CHARACTERS	12 5 8 6 +	/KEY:42:8 /COMP3/SIGNED

MR-S-1740-81

**SORT/MERGE Command Strings:**

**Standard Binary:**

```
*SORTED,FIL=DATA,FIL/BINARY/RECORD:12 -(RET)
# /KEY:5:2/UNSIGNED(RET)
```

**Mixed–Mode Binary:**

```
*SORTED,FIL=DATA,FIL/BINARY/EBCDIC/RECORD:46 -(RET)
# /KEY:17:11/COMP/UNSIGNED(RET)
```

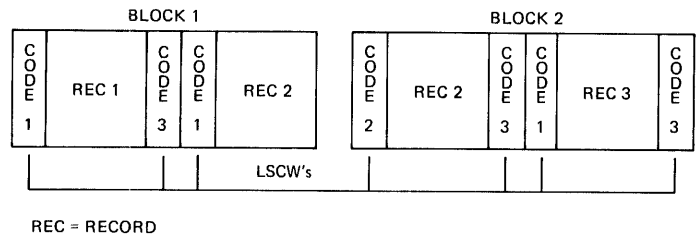
**NOTE**

To sort the file on the various fields of the record, replace the shaded portion of the standard binary command string with one or more of the key specifications given on the left–hand side of Figure 4–13. Likewise, replace the shaded portion of the mixed–mode binary command string with one or more of the key specifications given on the right–hand side of Figure 4–13.

## 4.6.2 FORTRAN Binary File Formats

FORTRAN programs can generate two types of binary files: those with Logical Segment Control Words (where MODE='BINARY') and those without Logical Segment Control Words (where MODE='IMAGE'). Also, each of these file types can be written, randomly (where ACCESS='RANDOM') or sequentially (where ACCESS='SEQOUT').

Logical Segment Control Words (LSCWs) are used to delimit each record of a file written with MODE='BINARY'. If the file is random, only two LSCWs occur per record: a code 1 LSCW at the beginning of the record and a code 3 LSCW at the end of the record. If the file is sequential, any record that crosses a block boundary has an additional LSCW at the beginning of the block containing the remainder of the record. This additional LSCW is a code 2 LSCW. The following diagram illustrates a sequential file containing all three types of LSCWs:



MR-S-1707-81

The LSCW is a 36-bit word consisting of an octal code value (1, 2, or 3) in the first nine bits of the word and a count value in the right half word. For random files and sequential files that do not cross block boundaries, the count values are as follows:

### LSCW      COUNT

CODE 1    Number of words in record + 1  
CODE 3    Number of words in record + 2

For sequential records that cross a block boundary, the count values are as follows:

### LSCW      COUNT

CODE 1    Number of words in the block  
CODE 2    Number of words in the record that cross the block boundary + 1  
CODE 3    Number of words in the record + 3

A further complication is that the four file types previously discussed can each be written in standard binary or mixed-mode binary (using the ASCII character set). Thus, there are actually eight major binary file formats written by FORTRAN programs:

1. Random Standard Binary with LSCWs
2. Random Standard Binary without LSCWs

3. Random Mixed-Mode Binary with LSCWs
4. Random Mixed-Mode Binary without LSCWs
5. Sequential Standard Binary with LSCWs
6. Sequential Standard Binary without LSCWs
7. Sequential Mixed-Mode Binary with LSCWs
8. Sequential Mixed-Mode Binary without LSCWs

The values for record length, key position, and length change, depending on whether or not the file is written in standard binary or mixed-mode binary. Also, the existence of control words and the use of sequential versus random I/O requires changes in the SORT/MERGE command string used to sort the file. Thus, it is very important that you understand the characteristics of a given FORTRAN-generated binary file before you attempt to sort it.

The following sections describe all of the previously mentioned file formats. Note that each file format diagram includes two sets of key specifications and command strings. The first set illustrates sorting the file in standard binary mode, and the second set illustrates sorting the file in mixed-mode binary. Also, note that, for the sake of program simplicity, the sample FORTRAN programs use literals to produce mixed-mode binary. The more common practice is to use ENCODE, DECODE, and FORMAT statements to produce mixed-mode binary.

#### 4.6.2.1 FORTRAN Random Binary (with LSCWs) —

PROGRAM:

```

DIMENSION A(128)
DOUBLE PRECISION Y; REAL X; INTEGER J
A(1)='ABCDE'; A(2)='FG'
A(3)='LM135'
A(4)='-1234'
A(5)='.1256'; A(6)='7E+03'
J   =1234567890
X   =98765.43212
Y   =500400300.200101
I   =1
OPEN (UNIT=1,DEVICE='DSK',FILE='DATA.FIL',MODE='BINARY',
1ACCESS='RANDOM',RECORDSIZE=132)
WRITE (1#I)A,J,X,Y
CLOSE (UNIT=1)
END

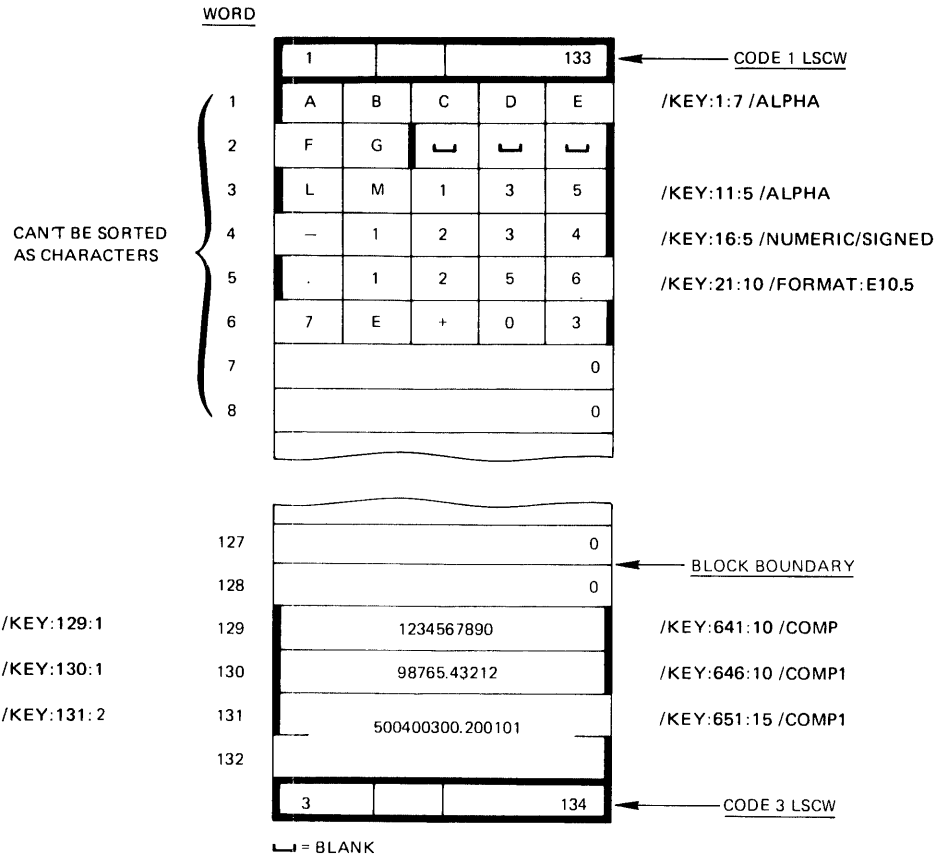
```

Figure 4-14 illustrates the record produced by the program shown above:

**Figure 4-14: FORTRAN Standard and Mixed-Mode Random Binary with LSCWs**

STANDARD FORTRAN RANDOM BINARY  
/FORTRAN/RANDOM/BINARY

MIXED-MODE FORTRAN RANDOM BINARY  
/FORTRAN/RANDOM/BINARY/ASCII



MR-S-1741-81

## SORT/MERGE Command Strings:

### Standard Binary:

```
*SORTED,FIL=DATA,FIL/FORTRAN/RANDOM/BINARY/RECORD:132 -(RET)
*/KEY:129:1(RET)
```

### Mixed-Mode Binary:

```
*SORTED,FIL=DATA,FIL/FORTRAN/RANDOM/BINARY/ASCII/RECORD:660 -(RET)
*/KEY:641:10/COMPUTATIONAL(RET)
```

## NOTE

To sort the file on the various fields of the record, replace the shaded portion of the standard binary command string with one or more of the key specifications given on the left-hand side of Figure 4–14. Likewise, replace the shaded portion of the mixed-mode binary command string with one or more of the key specifications given on the right-hand side of Figure 4–14.

FORTRAN users can specify the file formats illustrated in Figure 4–14 as follows:

Standard FORTRAN Random Binary:

1. /FORTRAN/RANDOM/BINARY
2. /FORTRAN/FIXED/BINARY

Mixed-Mode FORTRAN Random Binary:

1. /FORTRAN/RANDOM/BINARY/ASCII
2. /FORTRAN/FIXED/BINARY/ASCII

### 4.6.2.2 FORTRAN Random Binary (without LSCWs) —

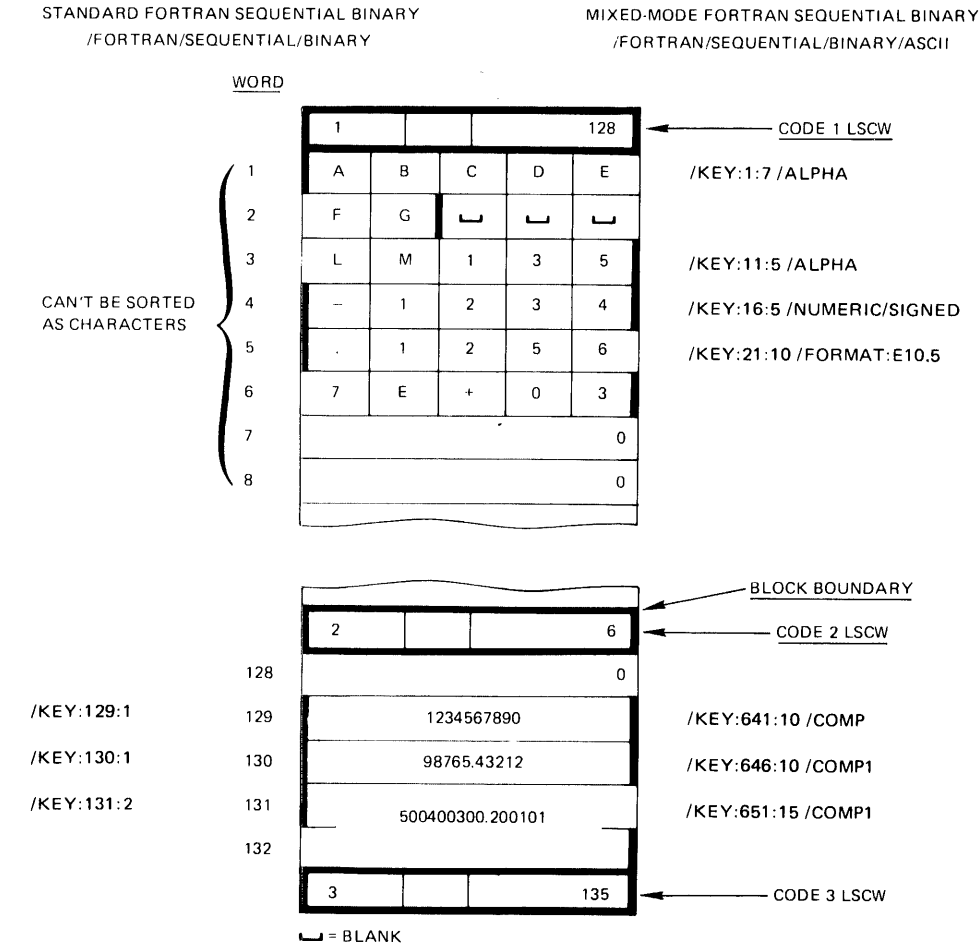
PROGRAM:

```
DIMENSION A(128)
DOUBLE PRECISION Y; REAL X; INTEGER J
A(1)='ABCDE'; A(2)='FG'
A(3)='LM135'
A(4)='-1234'
A(5)='.1256'; A(6)='7E + 03'
J   = 1234567890
X   = 98765.43212
Y   = 500400300.200101
I   = 1
OPEN (UNIT=1,DEVICE='DSK',FILE='DATA.FIL',MODE='IMAGE',
1ACCESS='RANDOM',RECORDSIZE=132)
WRITE (1#I)A,J,X,Y
CLOSE (UNIT=1)
END
```

Figure 4–15 illustrates the record produced by the program shown above:



**Figure 4-15: FORTRAN Standard and Mixed-Mode Random Binary**



**SORT/MERGE Command Strings:**

**Standard Binary:**

```
*SORTED,FIL=DATA,FIL/RANDOM/BINARY/RECORD:132 -(RET)
*/KEY:129:1(RET)
```

**Mixed-Mode Binary:**

```
*SORTED,FIL=DATA,FIL/RANDOM/BINARY/ASCII/RECORD:660 -(RET)
*/KEY:641:10/COMPUTATIONAL(RET)
```

## NOTE

To sort the file on the various fields of the record, replace the shaded portion of the standard binary command string with one or more of the key specifications given on the left-hand side of Figure 4–15. Likewise, replace the shaded portion of the mixed-mode binary command string with one or more of the key specifications given on the right-hand side of Figure 4–15.

FORTRAN users can specify the file formats illustrated in Figure 4–15 as follows:

Standard Random Binary:

1. /RANDOM/BINARY
2. /FIXED/BINARY
3. /BINARY(/FIXED in effect by default)

Mixed-Mode Random Binary:

1. /RANDOM/BINARY/ASCII
2. /FIXED/BINARY/ASCII

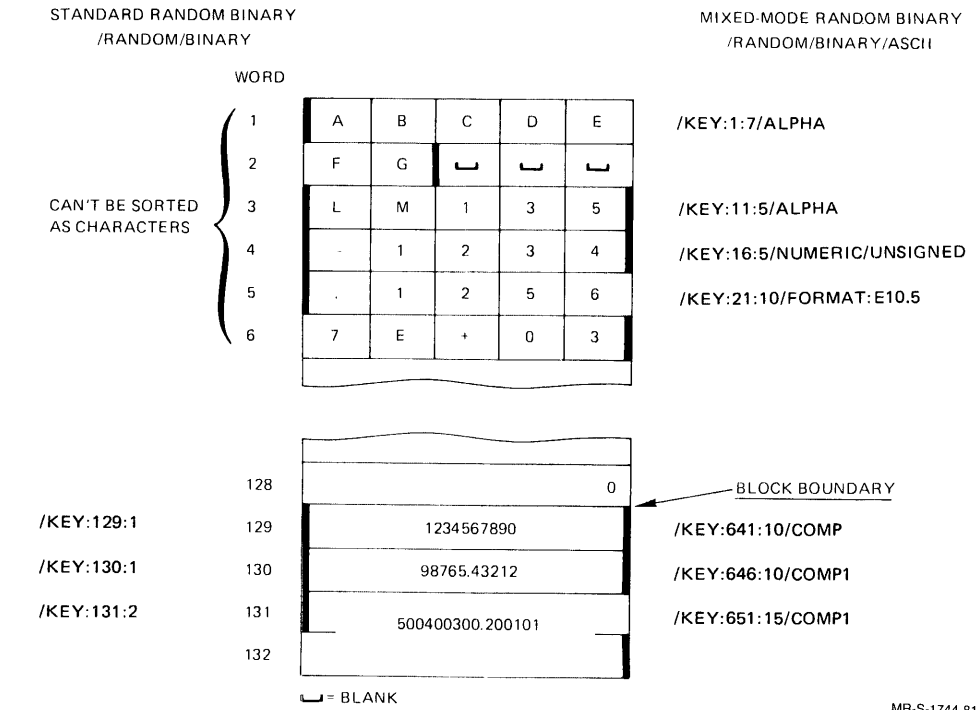
### 4.6.2.3 FORTRAN Sequential Binary (with LSCWs) —

PROGRAM:

```
DIMENSION A(128)
DOUBLE PRECISION Y; REAL X; INTEGER J
A(1)='ABCDE'; A(2)='FG'
A(3)='LM135'
A(4)='-1234'
A(5)='.1256'; A(6)='7E+03'
J   =1234567890
X   =98765.43212
Y   =500400300.200101
OPEN (UNIT=1,DEVICE='DSK',FILE='DATA.FIL',MODE='BINARY',
1ACCESS='SEQOUT')
WRITE (1)A,J,X,Y
CLOSE (UNIT=1)
END
```

Figure 4–16 illustrates the record produced by the program shown above:

**Figure 4-16: FORTRAN Standard/Mixed-Mode Sequential Binary with LSCWs**



**SORT/MERGE Command Strings:**

**Standard Binary:**

```
*SORTED,FIL=DATA,FIL/FORTRAN/SEQUENTIAL/BINARY/RECORD:132 -(RET)
*/KEY:129:1(RET)
```

**Mixed-Mode Binary:**

```
*SORTED,FIL=DATA,FIL/FORTRAN/SEQUENTIAL/BINARY/RECORD:660 -(RET)
*/KEY:641:10/COMPUTATIONAL(RET)
```

**NOTE**

To sort the file on the various fields of the record, replace the shaded portion of the standard binary command string with one or more of the key specifications given on the left-hand side of Figure 4-16. Likewise, replace the shaded portion of the mixed-mode binary command string with one or more of the key specifications given on the right-hand side of Figure 4-16.

FORTRAN users can specify the file formats illustrated in Figure 4–16 as follows:

Standard FORTRAN Sequential Binary:

1. /FORTRAN/SEQUENTIAL/BINARY
2. /FORTRAN/VARIABLE/BINARY

Mixed–Mode FORTRAN Sequential Binary:

1. /FORTRAN/SEQUENTIAL/BINARY/ASCII
2. /FORTRAN/VARIABLE/BINARY/ASCII

**4.6.2.4 FORTRAN Sequential Binary (without LSCWs)** — There are no control words or end-of-record control characters in this format. Therefore, if the records are of variable length, it is impossible for SORT/MERGE to extract a record length for each record. Thus, you cannot correctly sort this file format unless you observe the following restrictions:

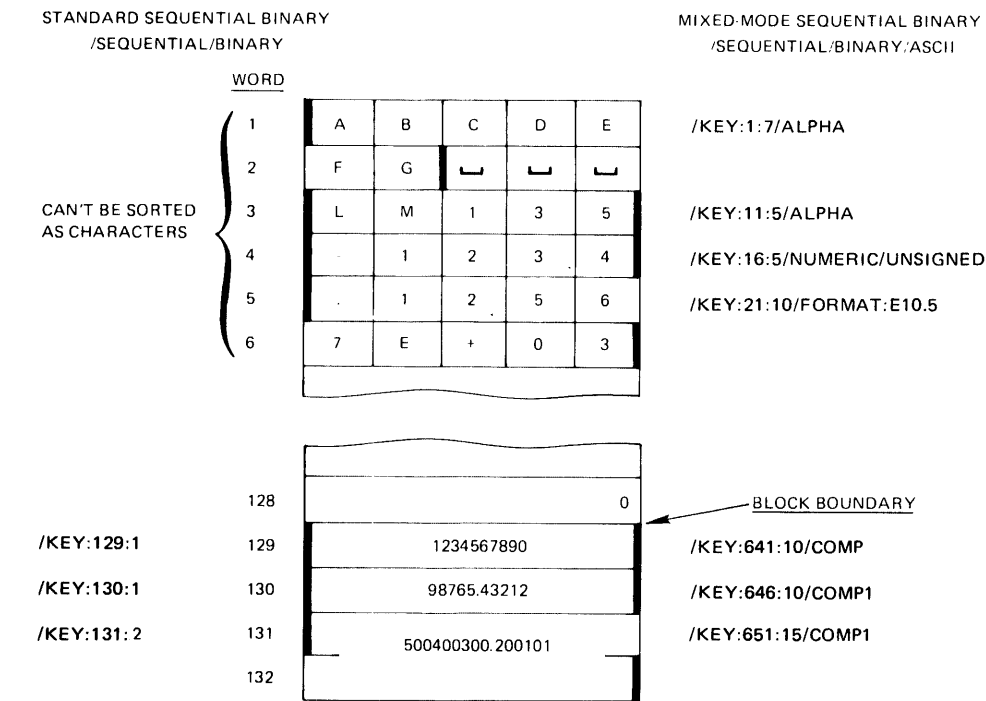
1. You must ensure that all records are actually of fixed length.
2. You must specify /RANDOM (or /FIXED), even though the MODE='SEQOUT' statement is used in the FORTRAN program.

PROGRAM:

```
DIMENSION A(128)
DOUBLE PRECISION Y; REAL X; INTEGER J
A(1)='ABCDE'; A(2)='FG'
A(3)='LM135'
A(4)='-1234'
A(5)='.1256'; A(6)='7E+03'
J   =1234567890
X   =98765.43212
Y   =500400300.200101
OPEN (UNIT=1,DEVICE='DSK',FILE='DATA.FIL',MODE='IMAGE',
1ACCESS='SEQOUT')
WRITE (1)A,J,X,Y
CLOSE (UNIT=1,DEVICE='DSK',FILE='DATA.FIL')
END
```

Figure 4–17 illustrates the record produced by the program shown above:

**Figure 4-17: FORTRAN Standard and Mixed-Mode Sequential Binary**



MR-S-1743-81

**SORT/MERGE Command Strings:**

**Standard Binary:**

```
*SORTED,FIL=DATA,FIL/RANDOM/BINARY/RECORD:132 -(RET)
#/KEY:129:1(RET)
```

**Mixed-Mode Binary:**

```
*SORTED,FIL=DATA,FIL/RANDOM/BINARY/ASCII/RECORD:660 -(RET)
#/KEY:641:10/COMPUTATIONAL(RET)
```

**NOTE**

To sort the file on the various fields of the record, replace the shaded portion of the standard binary command string with one or more of the key specifications given on the left-hand side of Figure 4-17. Likewise, replace the shaded portion of the mixed-mode binary command string with one or more of the key specifications given on the right-hand side of Figure 4-17.

FORTTRAN users can specify the file formats illustrated in Figure 4–17 as follows:

Standard Sequential Binary:

1. /RANDOM/BINARY
2. /FIXED/BINARY
3. /BINARY(/FIXED in effect by default)

Mixed–Mode Sequential Binary:

1. /RANDOM/BINARY/ASCII
2. /FIXED/BINARY/ASCII
3. /BINARY/ASCII(/FIXED in effect by default)



## Chapter 5

# SORT/MERGE Error Messages

### 5.1 Message Format

SORT/MERGE messages have the following format:

1. ?SRTxxx        text
2. %SRTxxx       text
3. [SRTxxx        text
4. \$SRTxxx        text

where:

?	= fatal error — program must be restarted
%	= warning message — program continues
[	= informational message — program continues
\$	= operator intervention message — program continues when operator has replied
SRT	= SORT/MERGE mnemonic
xxx	= 3-letter code for the message
text	= explanation of the message

The following section contains the messages that SORT/MERGE generates. The messages are in alphabetic order by the error code.



## 5.2 Error Messages

?SRT2N1	TOPS-20 version of SORT/MERGE will not run on TOPS-10.  This version of SORT/MERGE was compiled for a TOPS-20 system. It does not run on a TOPS-10 system.
%SRTALN	ANSI label not written.  No system tape label processor is present and, therefore, ANSI labels have been skipped over, but not checked.
%SRTANL	ANSI label not checked.  No system tape label processor is present and, therefore, no ANSI labels have been checked.
?SRTARL	ASCII record length incorrect.  One of the following occurred: <ol style="list-style-type: none"><li>1. You supplied an incorrect record length by including end-of-line characters or by not including sequence numbers in the record length count.</li><li>2. The record is not ASCII.</li></ol>
%SRTATD	Attempt to use temporary device failed—ignoring it.  SORT/MERGE was unable to write a temporary file on the device you specified by means of the /TEMP switch. SORT/MERGE ignores this device.
?SRTATF	At least two input files required for MERGE.  You specified the /MERGE switch, but only specified one input file. A merge is performed on two or more input files.
?SRTBNV	BINARY mode does not support variable length records.  Variable-length records are not allowed in binary files, except for FORTRAN binary.
?SRTCFE	Collating sequence input file error.  SORT/MERGE encountered an error while trying to read your collating sequence file. Try rerunning the sort.
?SRTCFS	Collating sequence file specification in error.  You gave an incorrect file specification for the /COLLATE file. Verify that your specification is correct.

%SRTCLC	Cannot lower core after SORT.  In a COBOL sort, someone expanded memory during a sort or merge. SORT/MERGE cannot, in this case, free the memory it was using, but otherwise continues normally.
?SRTCLS	Collating sequence literal specification in error.  Your /COLLATE:/LITERAL: switch argument is incorrect. Check for errors and respecify the command string.
?SRTCND	Collating sequence not defined.  In a FORTRAN-called sort, you specified an argument with the /COLLATE switch that SORT does not recognize.
?SRTCSD	Cannot set density to 'nnn'.  The specified device cannot be set to density 'nnn'.
?SRTCSM	Cannot set hardware data mode on "device".  The specified "device" cannot be set to industrial compatible mode.
?SRTCTL	Command string too long.  The command string to FSORT is longer than (5 * 128 **2) characters.
?SRTCWB	COMP key must be on word boundary.  You specified a key starting position that did not begin on a word boundary. All COMP and COMP1 keys must begin on a word boundary.
?SRTDDV	Double device illegal.  In a FORTRAN-called sort, you specified more than one device name for the same file specification.
?SRTDFK	Data mode switches must follow a /KEY switch.  In a FORTRAN-called sort, the data type switch must follow the key switch.
?SRTDND	Device 'dev' is not a disk. All scratch devices must be disks.  You specified a nondisk temporary device. This is illegal.
?SRTDNE	Device 'dev' does not exist.  You specified a nonexistent device.

?SRTELN	<p>EBCDIC tape labels not supported.</p> <p>No system tape label processor is present, and therefore SORT/MERGE cannot process EBCDIC labels.</p>
?SRTFCE	<p>Fortran command error.</p> <p>This is an internal FSORT error. Please contact your software specialist or send an SPR.</p>
?SRTFCN	<p>Attempt to free an I/O channel not retained or released.</p> <p>This is an internal SORT/MERGE error and is not expected to occur. Please contact your software specialist or send an SPR.</p>
%SRTFCR	<p>CORE MANAGEMENT ERROR AT RELSPC.</p> <p>This is an internal SORT/MERGE error and is not expected to occur. Please contact your software specialist or send an SPR.</p>
?SRTFCW	<p>FORTTRAN binary control word incorrect.</p> <p>You specified /FORTTRAN/BINARY and SORT/MERGE checked for Logical Segment Control Words (LSCWs). However, SORT/MERGE could not find any or all valid LSCWs. Check that you have specified a file that actually contains LSCWs and that you have properly specified /RANDOM or /SEQUENTIAL.</p>
?SRTFEA	<p>Formal argument count exceeds actual argument count.</p> <p>The argument referenced by ^n exceeds the argument supplied by the user.</p>
?SRTFMI	<p>Output switch illegal in input file.</p> <p>You specified an output-only switch such as /ESTIMATE (SCAN switch), on the input side of the command string.</p>
?SRTFMO	<p>File switches illegal in output file.</p> <p>You specified an input-only switch such as /TEMP, on the output side of the command string.</p>
?SRTFMR	<p>Attempt to free more memory than was originally retained.</p> <p>This is an internal SORT/MERGE error and is not expected to occur. Contact your software specialist or send an SPR.</p>

?SRTFNT	<p>Filename may not be specified with /TEMP device.</p> <p>You have specified a file name with a temporary device, such as DSK:FOO.BAR/TEMP. The /TEMP switch can only be specified with a device name.</p>
?SRTFSA	<p>/FORMAT switch argument error.</p> <p>Verify that the argument you specified with the /FORMAT switch is a valid format descriptor accepted by SORT/MERGE.</p>
?SRTFSM	<p>/FORMAT switch must be preceded by /KEY switch.</p> <p>You can only specify the /FORMAT switch after the /KEY switch it is intended to modify.</p>
?SRTFUF	<p>FILOP. function failed for file.</p> <p>This error message is followed by the ?SRTLRE LOOKUP/ENTER error message (see Table 5-1).</p>
%SRTIBL	<p>IBM label not checked.</p> <p>No system tape label processor is present and, therefore, no IBM labels were checked.</p>
?SRTICS	<p>Illegal user supplied collating sequence.</p> <p>SORT/MERGE discovered illegal characters in your collating sequence. Check the sequence for errors.</p>
?SRTIDS	<p>Illegal /DENSITY: value specified.</p> <p>You specified an illegal value with the /DENSITY switch.</p>
?SRTIEC	<p>Input error from indirect command file.</p> <p>SORT/MERGE received an error while trying to read the indirect command file passed to it by a FORTRAN program.</p>
?SRTIIF	<p>Illegal indirect filespec.</p> <p>The indirect command file specification is incorrect. Check it for errors.</p>
?SRTILC	<p>Illegal character x in numeric field.</p> <p>A nonnumeric character 'x' has been found in a field described as numeric.</p>
%SRTILN	<p>IBM label not written.</p> <p>No system tape label processor is present and therefore, IBM labels were skipped over on input and are not written on output.</p>

?SRTINA	<p>KI-10 version of SORT will not run on KA-10.</p> <p>This version of SORT/MERGE was compiled for a KI-10. You cannot use it on a KA-10.</p>
?SRTINS	<p>Input file not specified.</p> <p>You neglected to specify an input file.</p>
?SRTIRE	<p>Input read error, status 'xxxxxx'.</p> <p>SORT/MERGE got error code 'xxxxxx' from the monitor GETSTS UUO while trying to read a file. See the <i>TOPS-10 Monitor Calls Manual</i> for the meaning of this code.</p>
?SRTJAL	<p>Junk in ASCII line.</p> <p>In a fixed-length ASCII file, the characters after the record were not CR, LF, FF, or VT. You may have specified /FIXED for a variable-length file, or you may have specified the wrong record size.</p>
?SRTKAI	<p>Key argument incorrect.</p> <p>You specified an argument other than ASCENDING or DESCENDING with the /KEY switch.</p>
?SRTKCB	<p>Key comparison code too big.</p> <p>The 2000 words allocated for key extraction and/or key comparison code has been exceeded. Rebuild SORT with a larger FTZXSZ internal parameter.</p>
?SRTKEB	<p>Key extraction code too big.</p> <p>Same as error message ?SRTKCB.</p>
?SRTKLR	<p>Key length required.</p> <p>You must specify both key length and key starting position with the /KEY switch.</p>
?SRTKNR	<p>Key not fully contained in record.</p> <p>You have given a key starting position and length to a variable length file such that a record does not fully contain the key.</p>
?SRTKOR	<p>Key outside of record.</p> <p>You specified a key that extends outside the record (that is, extends past the value set with the /RECORD switch). If the data type is ALPHA and your file is variable-length, the key you specified can extend past the smaller records, but in no case can a key extend past the /RECORD value.</p>

?SRTLNC	<p>LABEL not correct for 'filespec'.</p> <p>The label on the tape does not correspond to the file specification given. Verify that you mounted the correct tape.</p>
?SRTLNI	<p>KL-10 version of SORT/MERGE will not run on KI-10 or KA-10.</p> <p>This version of SORT/MERGE was compiled for a KL-10. You cannot run it on a KI-10 or a KA-10.</p>
?SRTLRE	<p>ENTER error 'n' 'filespec'</p> <p>LOOKUP</p> <p>RENAME</p> <p>DELETE</p> <p>An ENTER, LOOKUP, RENAME, or DELETE error occurred for file 'filespec'. The error code 'n' can be found in Table 5-1, Error Codes, located at the end of this chapter.</p>
\$SRTLRI	<p>Load reel 'n' of output file 'filespec' type CONTINUE when ready.</p> <p>SORT/MERGE is ready to read the records from the next reel of a multireel input file. After the tape is properly mounted, type CONTINUE, and SORT/MERGE resumes reading the file.</p>
\$SRTLRO	<p>Load reel 'n' of output file 'filespec' type CONTINUE when ready.</p> <p>SORT/MERGE requires an additional magnetic tape to continue writing the output file. When the new tape is properly mounted, type CONTINUE, and SORT/MERGE resumes writing the file.</p>
?SRTMCS	<p>Multiple collating sequences not allowed.</p> <p>You can only specify one collating sequence.</p>
?SRTMGF	<p>Monitor GETTAB failed nnnnnn.</p> <p>This is an internal software error and is not expected to occur. Contact your software specialist or send an SPR.</p>
?SRTMOI	<p>Multiple output specs are illegal.</p> <p>You can specify more than one output file specification only if you are doing output to a magnetic tape.</p>
?SRTMOM	<p>Multiple output specs only on magtapes.</p> <p>You can specify more than one output file specification only if you are doing output to a magnetic tape.</p>

%SRTMRS	MERGE record 'n' not in sequence for 'filespec'.  SORT/MERGE has detected an out-of-sequence record in an input file during execution of the /MERGE switch when /CHECK was specified.
?SRTMSC	Mode switch conflict.  You may have specified two contradictory recording mode switches, such as /ASCII/EBCDIC. You can only pair recording mode switches for mixed-mode binary /BINARY/ASCII, /BINARY/SIXBIT, or /BINARY/EBCDIC. Or, you may have specified a data type that conflicts with the recording mode switch, such as using /COMP3 (or /PACKED) with /ASCII.
?SRTMSD	Multireel tape files with other than STANDARD or DEC labels not supported.  SORT/MERGE has filled up your output tape and needs another, but only tapes with STANDARD or DEC labels can be multireel. Use a larger tape, or a tape with one of these label types.
?SRTMTE	Max Temp Files must be in the range 3 to 26.  The /MAXTEMP switch must have a value between 3 and 26.
?SRTMUF	Monitor UVO failed xxxxxx.  This is an internal SORT/MERGE error and is not expected to occur. Contact your software specialist or send an SPR.
%SRTNCS	Not enough core specified.  The amount of memory that you specified with the /CORE switch was insufficient. SORT/MERGE uses the minimum amount of memory for the sort or merge.
?SRTNDV	Null device illegal.  The null device is illegal in a /COLLATE: switch.
?SRTNEC	Not enough core.  SORT/MERGE did not have enough memory to perform the sort. Try specifying more memory with the /CORE switch. Also, you may need to have your core limits increased by the system administrator.

?SRTNEH	<p>Not enough I/O channels for SORT/MERGE.</p> <p>SORT/MERGE cannot get enough I/O channels from the OTS. Check your FORTRAN or COBOL program and close some files before calling SORT/MERGE.</p>
?SRTNFS	<p>No filename specified for labeled tape 'filespec'.</p> <p>If you specify the /LABEL switch for a magnetic tape, then you must include a file specification which agrees with the file specification on the tape label.</p>
%SRTNLN	<p>Non-standard label not written.</p> <p>When you specify /LABEL:NONSTANDARD, SORT/MERGE skips over nonstandard labels on input and does not write them on output.</p>
%SRTNLS	<p>Not enough leaves specified.</p> <p>You specified a value with the /LEAVES switch that was too small. SORT/MERGE uses the minimum tree size of 16 leaves.</p>
?SRTNRL	<p>Name required with labeled magtape.</p> <p>If you specify the /LABEL switch for a magnetic tape, then you must include a file specification which agrees with the file specification on the tape label.</p>
%SRTNSL	<p>Non-standard label not checked.</p> <p>SORT/MERGE does not support nonstandard tape labels.</p>
?SRTNSW	<p>No temporary device is writeable.</p> <p>None of the temporary devices you specified is available.</p>
?SRTOFF	<p>OPEN failed for 'filespec'.</p> <p>SORT/MERGE could not open the specified file. Verify that the file you specified actually exists.</p>
?SRTOKR	<p>At least one key is required.</p> <p>You neglected to specify at least one key in your command string.</p>
?SRTONS	<p>Output file not specified.</p> <p>You neglected to specify an output file.</p>
?SRTOOF	<p>Only one /FORMAT switch per /KEY switch.</p> <p>You specified more than one /FORMAT switch with the same /KEY switch. This is illegal.</p>



?SRTOPF	<p>OPEN or LOOKUP failure for indirect command file.</p> <p>Verify that the command file you specified actually exists.</p>
?SRTOWE	<p>Output write error, status xxxxxx.</p> <p>SORT/MERGE got error code 'xxxxxx' from the monitor GETSTS UUO while trying to read a file. See the <i>TOPS-10 Monitor Calls Manual</i> for the meaning of this code.</p>
?SRTPRI	<p>Priority must be in range -3 to +3.</p> <p>The argument of the /PRIORITY: switch must be in the range -3 to +3.</p>
?SRTRIE	<p>Record incomplete at E-O-F.</p> <p>An end-of-file was encountered in the middle of a record. The input file is probably damaged or in the wrong format.</p>
?SRTRLO	<p>RELEASE called out of sequence. SORT not active.</p> <p>A COBOL program executed a RELEASE verb when a sort or merge was not in progress.</p>
%SRTRNI	<p>Record number inconsistent, 'n' read, 'm' written.</p> <p>This is an internal SORT/MERGE error and is not expected to occur. Contact your software specialist or send an SPR.</p>
?SRTROS	<p>Reel no. 'n' out of sequence for 'filespec'.</p> <p>The operator mounted the wrong reel of a multireel file.</p>
?SRTRSR	<p>Record size required.</p> <p>You neglected to specify the record size.</p>
%SRTRTI	<p>Record truncation on input.</p> <p>One or more variable-length records were longer than the record size specified with the /RECORD switch. The extra data is lost.</p>
?SRTRTO	<p>RETURN called out of sequence. SORT not active.</p> <p>A COBOL program executed a RETURN verb when a sort or merge was not in progress.</p>
?SRTSAT	<p>Standard ASCII requires TU70 drive.</p> <p>To read/write standard ASCII mode on tape, SORT/MERGE requires a TU70 tape drive. Other tape drives do not have the hardware capability for standard ASCII mode.</p>

?SRTSFD	SFD depth greater than 5. The SFD nesting is greater than that allowed.
?SRTSFF	Switches must follow filespecs. In a FORTRAN-called sort, switches must follow the file specification. They cannot precede them.
?SRTSRM	SORT/MERGE will not run on this machine. This version of SORT/MERGE was compiled for a machine other than the one you are using.
?SRTSRS	Spanned records not supported. The Record Descriptor Words (RDWs) in your variable-length EBCDIC file are nonzero. This indicates spanned variable-length EBCDIC records, a file format that SORT/MERGE does not support.
?SRTSSE	Switch scanning error. This is an internal SORT/MERGE error and is not expected to occur. Contact your software specialist or send an SPR.
?SRTSVR	Switch value required. You neglected to specify an argument for a switch that requires one.
?SRTSWP	Temporary structure 'dev' is write-locked. You specified a write-locked device with the /TEMP switch, and SORT/MERGE could not use it.
?SRTTMD	Too many digits in key. You specified a COMP or COMP1 key with more digits than would fit in two words (for double precision) of storage.
%SRTTMT	Too many temporary structures specified. You have specified more temporary devices than SORT/MERGE is able to use. The extra areas are ignored.
?SRTUDL	Unknown delimiter. An illegal character was found in the command string.
?SRTUKS	Unknown switch '/switch'. You specified a switch that SORT/MERGE does not recognize.

?SRTUSV	Unknown switch value.  You specified a switch value or argument that SORT/MERGE does not recognize.
SRTXPN	Expanding to nK   For stand-alone sorts, this indicates how much memory is being used for the sort. For COBOL or FORTRAN-called sorts, this is an internal SORT/MERGE debugging message which only occurs if the feature test switch FTDEBUG is nonzero.

### 5.3 Error Codes

The error codes in Table 5-1 are returned in the AC on RUN and GETSEG monitor calls, in the right half of location E + 1 on 4-word argument blocks of LOOKUP, ENTER, and RENAME monitor calls, and in the right half of location E + 3 on extended LOOKUP, ENTER, and RENAME monitor calls.

**Table 5-1: Error Codes**

Symbol	Code	Explanation
ERFNF%	0	File not found, illegal filename (0,*) file names do not match (UPDATE), or RENAME after a LOOKUP failed. On a FILOP., this error is given if the specified device cannot perform I/O in the direction indicated.
ERIPP%	1	UFD does not exist on specified file structures. (Incorrect project-programmer number.)
ERPRT%	2	Protection failure or directory full on DTA.
ERFBM%	3	File being modified (ENTER, RENAME).
ERAEF%	4	Already existing filename (RENAME or FILOP.), different filename (ENTER after LOOKUP) or supersede (on a non-superseding ENTER). Two LOOKUPs or two ENTERs were performed.
ERISU%	5	Illegal sequence of monitor calls (RENAME with neither LOOKUP nor ENTER, or LOOKUP after ENTER).
ERTRN%	6	One of the following: <ul style="list-style-type: none"> <li>1. Transmission, device, or data error (RUN, GETSEG only).</li> <li>2. Hardware-detected device or data error detected while reading the UFD RIB or UFD data block.</li> <li>3. Software-detected data inconsistency error detected while reading the UFD RIB or file RIB.</li> </ul>
ERNSF%	7	Not a saved file (RUN, GETSEG only).
ERNEC%	10	Not enough core (RUN, GETSEG only).

**Table 5–1 (Cont.): Error Codes**

Symbol	Code	Explanation
ERDNA%	11	Device not available (RUN, GETSEG only).
ERNSD%	12	No such device (RUN, GETSEG only).
ERILU%	13	Illegal monitor call (GETSEG only). No 2-register relocation capability.
ERNRM%	14	No room on this file structure or quota exceeded (overdrawn quota not considered).
ERWLK%	15	Write-lock error. Cannot write on file structure.
ERNET%	16	Not enough table space in free core of monitor.
ERPOA%	17	Partial allocation only.
ERBNF%	20	Block not free on allocated position.
ERCSD%	21	Cannot supersede an existing directory (ENTER).
ERDNE%	22	Cannot delete a nonempty directory (RENAME).
ERSNF%	23	Subdirectory not found (some SFD in the specified path was not found).
ERSLE%	24	Search list empty (LOOKUP or ENTER was performed on generic device DSK and the search list is empty).
ERLVL%	25	Cannot create an SFD nested deeper than the maximum allowed level of nesting.
ERNCE%	26	No file structure in the job's search list has both the no-create bit and the write-lock bit equal to zero, and has the UFD or SFD specified by the default or explicit path (ENTER on generic device DSK: only).
ERSNS%	27	GETSEG from a locked low segment to a high segment which is not a dormant, active, or idle segment. (Segment not on the swapping space.)
ERFCU%	30	The file cannot be updated.
ERLOH%	31	The low segment overlaps the high segment (GETSEG).
ERNLI%	32	The user is not logged in (RUN).
ERENQ%	33	The file still has outstanding locks set.
ERBED%	34	The file has a bad .EXE file directory (GETSEG, RUN).
ERBEE%	35	The file has a bad extension for an .EXE file (GETSEG, RUN).
ERDTB%	36	The file's .EXE directory is too big (GETSEG, RUN).
ERENC%	37	Network capacity has been exceeded, not enough space for connect message (LOOKUP/ENTER).
ERTNA%	40	Task not available (LOOKUP/ENTER/RENAME).
ERUNN%	41	Unknown network node; node went down during connect (ENTER).



## Chapter 6

# **SORT/MERGE Performance Considerations**

This chapter gives a general description of the SORT/MERGE binary tree and describes how the operation of SORT/MERGE is affected by tree size, buffer size, and number of merge passes. This is not a complete description of SORT/MERGE internals. This chapter simply gives you enough information to modify SORT/MERGE's performance by use of the /LEAVES and /CORE switches.

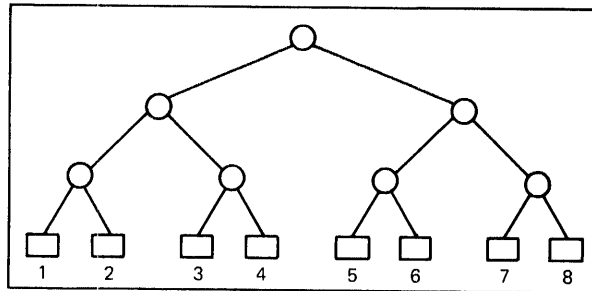
### **6.1 Performance Overview**

SORT/MERGE consists of two major sets of routines, I/O routines and sorting routines. The I/O routines strip off header words, control words, and control characters on input and restore them on output. This is necessary as only pure data can be passed to the sorting routines. There is nothing that you can do to improve the performance of the I/O routines. However, if the file format is not correctly specified, the data is not correctly extracted from the file; header words, control words, or control characters are erroneously passed to the sorting routines. Thus, the results are unpredictable.

The sorting modules perform two functions: sorting and merging data. These two functions are described more fully in the following sections. Since you cannot actively modify the operation of the merge phase, most of the discussion is concerned with the sort phase. You can, however, optimize the number of merge passes that occur. That technique is described in this chapter.

SORT/MERGE uses a variable-size binary tree to sort and merge input files. Figure 6-1 illustrates a simple binary tree:

**Figure 6-1: Binary Tree**



MR-S-1720-81

The tree size is measured by the number of its leaves and corresponds to the number of records the tree holds. For example, the tree shown above has eight leaves, and thus eight records.

#### NOTE

Although SORT/MERGE's minimum tree size is 16 leaves, this chapter uses fewer leaves in the examples for readability.

If you do not specify the size of the tree (with the /LEAVES switch), then SORT/MERGE picks the largest tree greater than 16 leaves that fits in your available memory space.

### 6.1.1 The Sort Phase

The following sections should be read in conjunction with Figure 6-2. This figure illustrates how a binary tree is used to sort and merge data. To keep the figure as simple as possible, the following conditions and conventions have been used:

1. The sorting tree is limited to four leaves (versus a minimum of 16 leaves for SORT/MERGE).
2. Letters are used to represent records (not characters) of arbitrary size. The key field is assumed to be as long as the record.
3. For the purposes of this chapter, "smallest" refers to an ordering that fully takes into account multiple keys with possibly differing ASCENDING or DESCENDING key arguments.
4. The leaves (terminal nodes) of the tree are numbered, and these numbers represent the "address" of the leaves.
5. The upper, or nonterminal, nodes contain pointers to the LOSING key of any two keys that are compared. An additional 'node', shown outside the tree, points to the WINNING key in the tree. For example, if the

records are being sorted in ascending order, then the object is to find the smallest key in the tree at that time. The nonterminal nodes point to the larger of any two keys compared, while the external 'node' points to the smallest key in the tree.

6. The sort shown in the figure is limited to a maximum of two temporary files (rather than the maximum of 26 temporary files for SORT/MERGE).

The sorting process works as follows: SORT/MERGE fills the tree to its capacity with records from the input file. (Multiple input files are treated as one file.) All records in the tree are compared. Then the smallest record is ejected from the tree and written to a temporary file. The ejected record's position in the tree is then filled with the next input record, and the process is repeated. In the special case where the number of records initially in the tree is less than or equal to the capacity of the tree, ejected records are written directly to the output file.

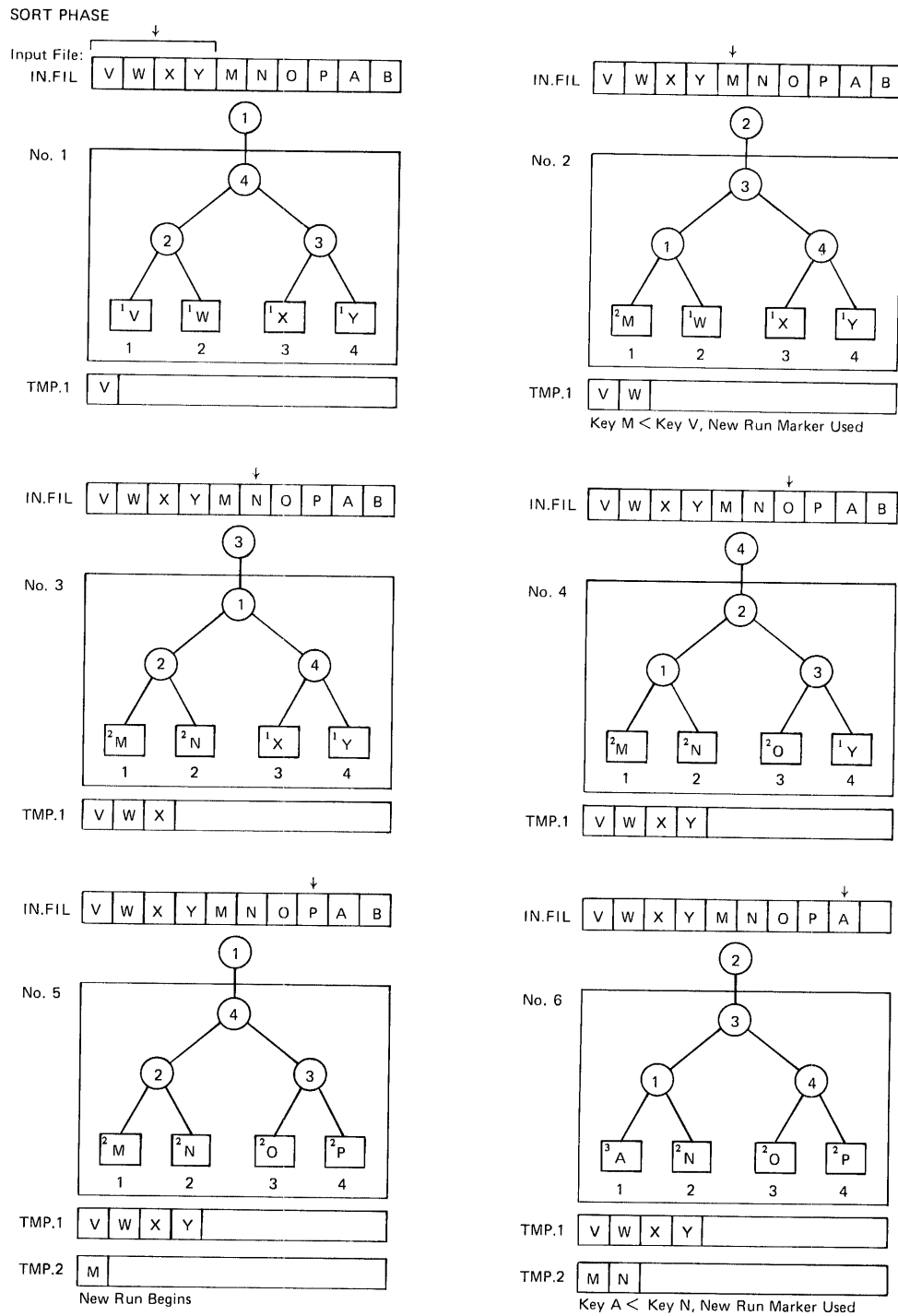
SORT/MERGE continues reading records until one is read that is less than the record just output. This record and all subsequent ones less than the record currently being ejected from the tree are marked to be sent to a new run. When one of the records marked for the new run reaches the top of the tree, the collection of records currently in the temporary file constitutes the longest sequence of ordered records SORT/MERGE could generate, given the size of the tree. SORT/MERGE now creates a new run, beginning with the currently ejected record. The new run is appended to the end of the next temporary file, after any existing runs. SORT/MERGE begins each run, except the first in a file, with a run marker containing the run number of the new run. SORT/MERGE continues sorting records and writing runs until all records have been read. At this point, the sort phase is over.

### 6.1.2 The Merge Phase

For the merge phase, SORT/MERGE reinitializes the tree to contain as many leaves as there are temporary files, and then fills the tree with the first record from each temporary file. The smallest record is then written to a new temporary file, and another record is read into the tree from the same file that the currently ejected record was read from. New merge passes are initiated as long as more than one temporary file remains. When only one temporary file remains, this file is renamed or copied to the output file.

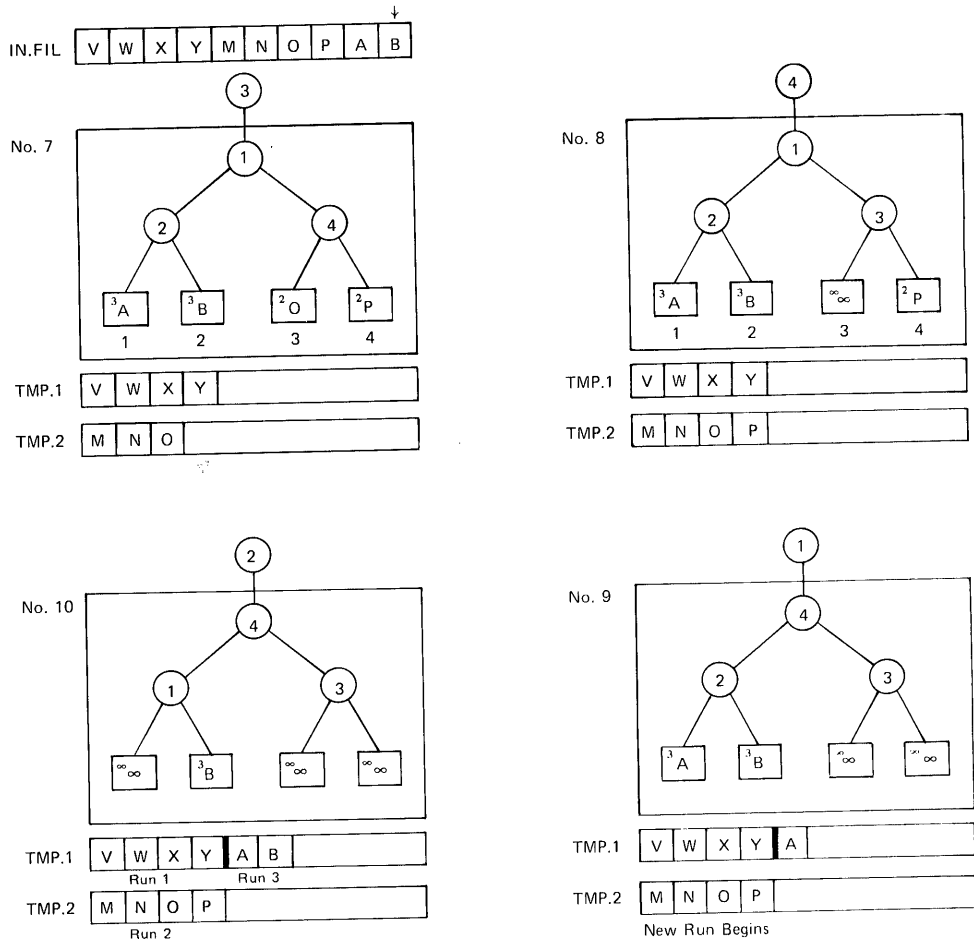


**Figure 6-2: The Operation of the SORT/MERGE Binary Tree**



MR-S-1721-81

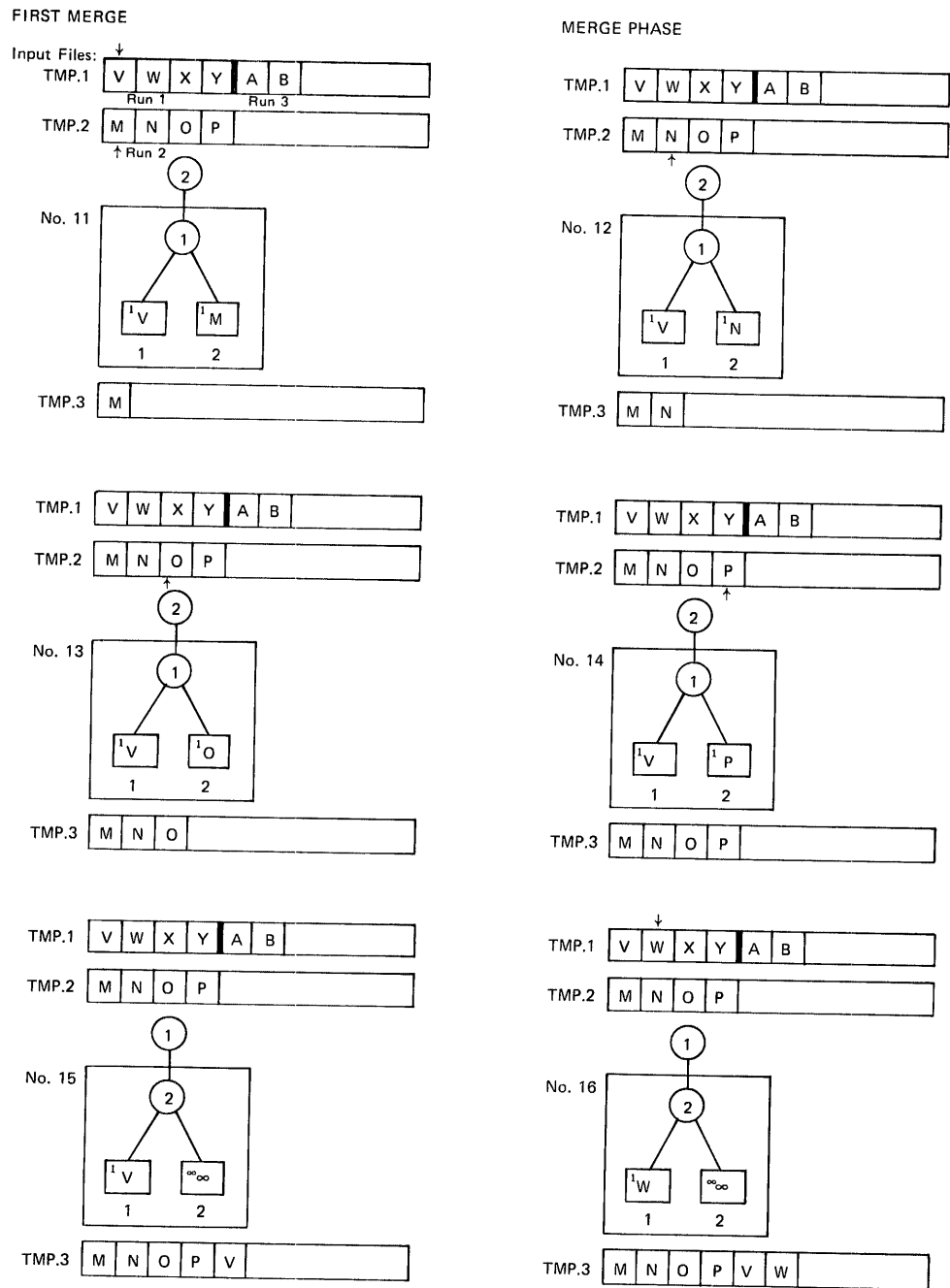
**Figure 6-2 (Cont.): The Operation of the SORT/MERGE Binary Tree**



NOTE: Heavy vertical bars indicate run markers.  
 Arrows point to the record(s) currently being read into the tree.  
 Superscripts preceding each letter are run markers.  
 ∞ = Special record used by SORT/MERGE to flush the tree of records.  
 ∞ = Special run marker.

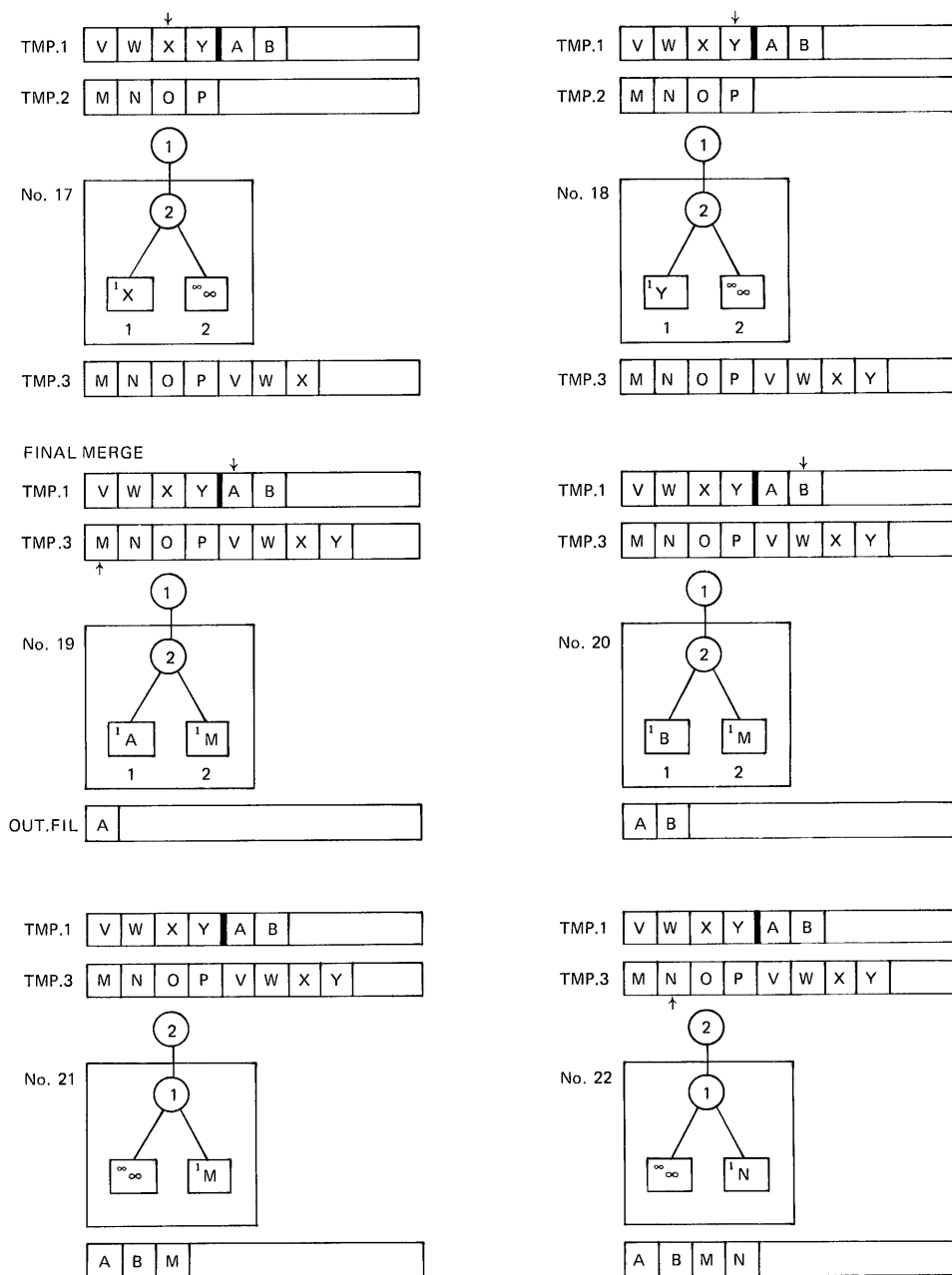
MR-S-1722-81

Figure 6-2 (Cont.): The Operation of the SORT/MERGE Binary Tree



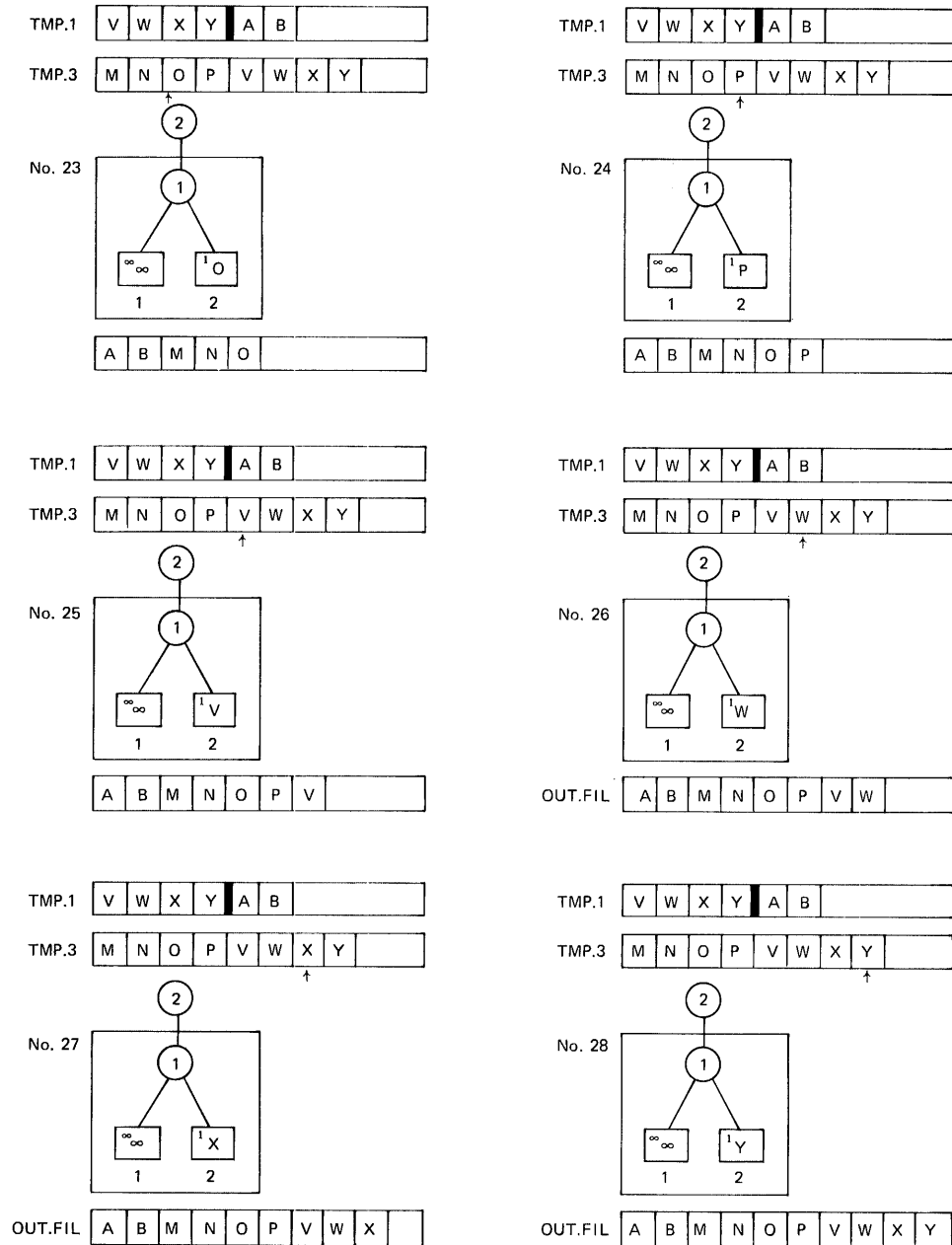
MR-S-1723-81

**Figure 6-2 (Cont.): The Operation of the SORT/MERGE Binary Tree**



MR-S-1724-81

**Figure 6-2 (Cont.): The Operation of the SORT/MERGE Binary Tree**



MR-S-1725-B1

## 6.2 Performance Considerations

There are three major areas of concern in tuning the performance of SORT/MERGE. They are:

1. Tree size
2. Memory size (low segment size)
3. Number of merge passes

The following sections describe all of these areas and give general guidelines for adjusting them to improve the performance of SORT/MERGE. Variables such as installation memory size, system load, file size, file order, key type and length, data type, and recording mode make it impossible to give a simple, well-defined formula for improving the performance of a particular sort. If you are concerned with performance, you should study the guidelines given below and experiment with them.

If you are doing stand-alone sorts, then SORT/MERGE, after completion of the sort, prints statistics concerning the sort. (COBOL and FORTRAN-called sorts do not print these statistics.) Included in these statistics are the tree size in leaves, the low segment size, and the number of runs. Thus, you can study these values and decide which must be altered for performance improvement. This is especially important in sorts that are run using default values for tree size and memory size. The following illustrates the display of sort statistics:

```
[SRTXPN Expanding to 46P]  
Sorted 7 records  
12 KEY comparisons, 1.71 per record  
40 record leaves in memory  
0 runs  
0:00:00 CPU time, 16.43 MS per record  
0:00:42 elapsed time
```

### 6.2.1 Tree Size

During the sort phase, as the tree size increases, fewer runs and temporary files occur, requiring fewer merge passes later. However, a tree that is too large places a heavy burden on system resources and causes a subsequent performance degradation of the sort. A tree that is too small causes excessive merge passes and increases the run time of the sort. The file itself further affects performance. For a given tree size, a well-ordered file results in fewer (and longer) runs and therefore, fewer merge passes. A backwards-sorted file results in many runs and merge passes.

The effect of the tree size on runs is as follows:

If N is the number of leaves in the tree, then:

1. Worst case file — each run except the last has N records
2. Random order file — each run contains approximately  $2^*N$  records

3. Best case file — produces one run equal to the size of the entire input file

The bias is the ratio of the run to the tree size, and is a useful index of the randomness of a file. The bias is printed only if at least one temporary file is written. The following illustrates how to interpret the bias:

bias =	1	Descending file sorted in ascending order (or vice versa)
bias =	2	Randomly ordered files
bias =	3	60% of the records are random, 40% are in order
bias =	7	20% of the records are random, 80% are in order

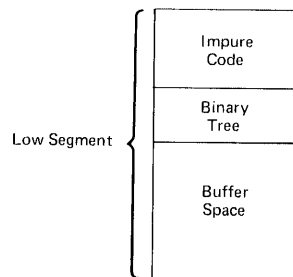
For example, a bias of 7 indicates that 80% of the records in the file are in order and each run contains approximately seven times the number of records which would fit in the tree.

SORT/MERGE's default memory algorithm uses the largest tree size larger than 16 leaves that fits in the memory available.

You can specify the tree size yourself with the /LEAVES switch. However, you should read Section 6.2.3 before using this switch.

## 6.2.2 Memory Size

SORT/MERGE's low segment appears as follows during the sort or merge phase:



MR-S-1727-81

Assuming that you have adjusted the tree size to an optimum value, you should determine how many buffers SORT/MERGE generates for the sort. A large number of I/O buffers per file causes a memory image bigger than TOPS-10 can handle. Then the sort places a heavy burden on the system and causes a subsequent performance degradation. An insufficient number of I/O buffers causes inefficient I/O data transfer. As SORT/MERGE's default algorithm cannot always use the optimum memory size, the efficiency of I/O buffer transfer can be improved if you specify a memory size that causes SORT/MERGE to create a number of buffers more appropriate for your job.

The minimum number of buffers is usually two buffers for each input and output file. Two buffers allow double buffering; the monitor reads the second buffer while SORT/MERGE is processing the first buffer. For disks, a good number of buffers is the cluster size of the disk plus one (five is usually adequate). Buffer lengths are as follows:

- disk files – buffer length is 131 words
- tape files – buffer length is equal to the size of a physical record plus three

See the *TOPS-10 Monitor Calls Manual* for other devices.

Some calculation is involved before you can determine the approximate number of buffers used in your sort. First, examine the following terms that are used in the calculations. Unless otherwise noted, all values are DECIMAL words.

1. *n* – number of leaves in the tree
2. *r* – maximum record length in words
3. *c* – 7680 words (15 pages). This is the length of SORT/MERGE impure code for stand-alone sorts.
4. *e* – word length of extracted keys (see below)
5. *p* – 4 words (3 words of tree pointers plus 1 internal count word)
6. *k* – key length in bytes
7. *lw* – length of low segment in words
8. *lp* – length of low segment in pages
9. *s* – length of impure code plus length of binary tree
10. *b* – length of buffer space
11. *t* – size of binary tree in words
12. *cl* – disk cluster size plus 1 for nondisk devices,  $cl = 2$
13. *m* – maximum of 131 (size of a temporary file buffer) and the maximum buffer length for any input device

The following example explains the calculations you must perform. The following assumptions are made:

1. The sort uses only disk files having a cluster size of 1.
2. There are no extracted keys.
3. The sort does not use the /COLLATE switch.
4. SORT/MERGE statistics indicate that the tree size is 100 leaves and the low segment size is 100 pages.



5. The tree size parameter has already been examined and determined to be optimum.
6. The record size is eight words.

The first step is to convert the reported low segment size to words (in algebraic form):

$$\begin{aligned} lw &= (lp * 512) \\ lw &= (100 * 512) \\ lw &= 51200 \end{aligned}$$

The second step is to find the value of  $t$ , the size of the binary tree (in algebraic form):

$$\begin{aligned} t &= (n * (r | + e | + p)) \\ t &= (100 * (8 | + 0 | + 4)) \\ t &= (100 * (12)) \\ t &= 1200 \end{aligned}$$

The third step is to find the value of  $s$ , the tree size plus impure code (in algebraic form):

$$\begin{aligned} s &= t + c \\ s &= 1200 + 7680 \\ s &= 8880 \end{aligned}$$

The fourth step is to find the value of  $b$ , the buffer space (in algebraic form):

$$\begin{aligned} b &= lw - s \\ b &= 51200 - 8880 \\ b &= 42320 \end{aligned}$$

The fifth step is to divide the buffer space by 2. This is because, during the sort phase, SORT/MERGE has exactly two files open: the current input file and a temporary file:

$$42320/2 = 21160$$

The sixth step is to divide the buffer size by  $m$ . In this case,  $m = 131$ .

$$21160/131 = 161.5 \text{ or } 161 \text{ buffers (rounded downward)}$$

Thus, SORT/MERGE used 161 buffers per file. This figure is high, as the optimum number of buffers is usually two per file, or the cluster size plus one for disk. The following computation (in algebraic form) gives you the required buffer space, assuming two buffers per file:

$$\begin{aligned} b &= cl * m * 2 \\ b &= 2 * 2 * 131 \\ b &= 524 \text{ words} \end{aligned}$$

If you add this value to the value you previously calculated for  $s$  (impure code plus tree), you get:

$$8880 + 524 = 9404 \text{ words}$$

Thus, lw (length of low segment in words) = 9404.

To calculate lp (low segment in pages):

$$lp = lw / 512$$

$$lp = 9404 / 512$$

$$lp = 18.3 \text{ rounded up to } 19$$

Thus, you would specify the following switch combination to SORT/MERGE:

/LEAVES:100/CORE:19P

#### NOTE

If, after attempting to run the sort with the new /CORE value, you receive a message indicating that SORT/MERGE did not have enough core, rerun the sort after adding one or two pages to the /CORE value. Rounding errors can sometimes cause you to calculate a value that is slightly less than the required value.

In the example above, you assumed no extracted keys. Thus, the value of e was zero. If the data type switch is /NUMERIC, /FORMAT, /COMPU, or /COMP1, then e = 1 for single precision keys; and e = 2 for double precision keys.

If you used the /COLLATE: switch in the command string, then e is calculated by a different method. (Note that k = key length.)

If you used the /COLLATE:ASCII switch, or the /COLLATE:FILE: switch, or the /COLLATE:/LITERAL: switch, where the file or the literal contained an alternative ASCII collating sequence, then:

$$e = (k + 4)/5$$

If you used the /COLLATE:EBCDIC switch, or the /COLLATE:FILE: switch, or the /COLLATE:/LITERAL: switch, where the file or the literal contained an alternative EBCDIC collating sequence, then:

$$e = (k + 3)/4$$

In summary, the various formulas that you used are as follows:

1. Convert n pages to words:

$$n / 512$$

2. Convert n words to pages:

$$n * 512$$

3. Calculate the number of buffers per file for a given low segment value:

$$\frac{|lw - ((n * (r | + e| + p)) + c)|}{(cl * m)}$$

4. Calculate an optimized number of buffers:

$$cl * m * 2$$

### 6.2.3 Number of Merge Passes

The largest number of runs that SORT/MERGE can maintain before doing a merge pass is an exponential function of the number of temporary files available. For stand-alone sorts, the number of temporary files is fixed at 26. As merge passes are very expensive in terms of run time, it is important to ensure that no more merge passes occur than are absolutely necessary. You should observe the following figures:

<u>RUNS</u>	<u>MERGE PASSES</u>
0	0 (sort contained entirely in memory)
1 – 26 ( $26^{**1}$ )	1
26 – 676 ( $26^{**2}$ )	2
676 – 17576 ( $26^{**3}$ )	3
17576 – 456976 ( $26^{**4}$ )	4

The values shown above that are exponential powers of 26 are the merge threshold values. If the number of runs exceeds any given threshold limit, then an additional merge pass is necessary. This is particularly unfortunate in cases where the number of runs is only 10 to 20% greater than the threshold limit. As an example, consider the case of a sort that uses 29 runs. This is only 3 runs above the closest threshold value (26). For the sake of 3 runs, a very substantial performance degradation is incurred. Thus, a slight increase in the size of the sort tree might reduce the number of runs to, or below, the 26-run threshold value and eliminates the need for an entire merge pass. However, a slight increase in memory usage gives a substantial reduction in the run time of a sort. Therefore, the tree size is specified with the /LEAVES switch. It is best to increase the size of the tree until the number of runs equals, or drops below, the nearest threshold value.

Alternately, you may be more concerned with memory usage than with run time. If the number of runs is 10 – 20% less than the nearest merge threshold value, you can effect a gain in system performance (at a slight expense in run time for the sort) by specifying a smaller tree. Again, decrease the tree size until the number of runs is equal to or less than the nearest merge threshold value.

#### NOTE

The length of each run is data dependent. What works one time may be incorrect the next time if the data is different.

# Appendix A

## Summary of SORT/MERGE Functions and Switches

This appendix has alphabetic lists of SORT/MERGE functions and SORT/MERGE switches. Each command or switch is described briefly. Complete descriptions of functions can be found in Chapter 2. Complete descriptions of switches can be found in Chapter 3.

Each switch is given a brief description that includes its function, its type, and any defaulting characteristics of the switch.

### A.1 Functions

There are five major functions in SORT/MERGE:

- EXIT – exit from SORT/MERGE and return to TOPS-10 command level.  
Use /EXIT
- HELP – print the text of the help file.  
Use /HELP or /HELP:SWITCHES
- MERGE – merge previously sorted files.  
Use /MERGE with a SORT/MERGE command string
- RUN – run a program.  
Use /RUN:
- SORT – sort the specified files.  
Simply specify a SORT/MERGE command string

## A.2 Switches (in Alphabetic Order)

### /AFTER

Place the output record after the carriage–return/line–feed characters.

Function: File switch

Type: Local

Default: /BEFORE switch

### /ALIGN

Word align all ASCII output records.

Function: File switch

Type: Global

Default: Not word–aligned

Exceptions:

1. /ASCII/FORTRAN is word–aligned by default.
2. Line–sequenced ASCII input files are detected by SORT/MERGE, and are word–aligned on output.

### /ALPHA

The key data type is alphanumeric.

Function: Key data type switch

Type: Local

Default: If /SIGNED, /UNSIGNED, /FORMAT, /NUMERIC, /COMPU, /COMP1, or /COMP3 not specified, this is the default data type.

### /ASCII

The recording mode of the data is ASCII.

Function: Recording–mode switch

Type: Global

Default: See the explanation of this switch in Chapter 3 for a description of the defaults.

### /BEFORE

Place the output record before the carriage–return/line–feed characters.

Function: File switch

Type: Local

Default: This is the default.

## **/BINARY**

The recording mode of the data is binary.

Function: Recording-mode switch

Type: Global

Default: See the explanation of this switch in Chapter 3 for a description of the defaults.

## **/BLOCKED:n**

The n is a decimal number indicating the blocking factor for a COBOL file. Not meaningful for FORTRAN-created files.

Function: File switch

Type: Modified position dependent

Default: Unblocked file

## **/CHECK**

Check the record sequence of files that are being merged and generate an error message if out-of-sequence records are found. This switch doubles the number of comparisons and causes some increase in run time. However, it protects against erroneously merging unsorted files. Valid only for the merge function.

Function: Control switch

Type: Global

Default: Don't check merge input files

## **/COLLATE:n**

The argument to this switch (n) specifies the collating sequence to use when sorting the associated files. The possible arguments are:

1. ASCII
2. EBCDIC
3. FILE:filespec
4. LITERAL:/collating-sequence/
5. ADDRESS:address

Arguments 1 and 2 allow you to sort ASCII data according to the EBCDIC collating sequence or vice versa. Arguments 3 and 4 allow you to specify your own collating sequence. Argument 5 is for FORTRAN users in the format of ADDRESS:ˆn.

Function: Control switch

Type: Global

Default: The collating sequence associated with the recording-mode switch used.

#### /COMPU

The key data type is COMPUTATIONAL.

Function: Key data type switch

Type: Key modifier

Default: If the recording mode is binary, this is the default data type.

#### /COMP1

The key data type is COMP1.

Function: Key data type switch

Type: Key modifier

Default: No default characteristics

#### /COMP3

The key data type is COMP3.

Function: Key data type switch

Type: Key modifier

Default: No default characteristics

#### /CORE:n

The n is the amount of the low segment to be allocated to SORT/MERGE.

Function: Control switch

Type: Global

Default: Determined by SORT/MERGE's default core algorithm

#### /DENSITY:n

The n is a decimal number indicating the recording density to be used for reading or writing a magnetic tape. This switch is ignored for other devices. The possible values of n are:

1. 200
2. 556
3. 800
4. 1600
5. 6250

Function: Tape switch

Type: Position dependent (SCAN switch)

Default: Set by SET DENSITY monitor command

## `/EBCDIC`

The recording mode of the data is EBCDIC.

Function: Recording-mode switch

Type: Global

Default: See the explanation of this switch in Chapter 3 for a description of the defaults.

## `/ERROR:addr`

The `addr` is the absolute octal address that program control transfers to when SORT/MERGE encounters a fatal error. This switch is designed for FORTRAN users, who can specify an absolute address or (indirectly) a symbolic address. See the description of this switch in Chapter 3 for further details.

Function: Control switch

Type: Global

Default: Stop the job on a fatal error

## `/EXIT`

Exit to monitor command level.

Function: Control switch

Type: Global (SCAN switch)

Default: Prompt for another SORT/MERGE command string

## `/FATAL:addr`

The `addr` is the absolute octal address that the SORT/MERGE error code is stored in when SORT/MERGE encounters a fatal error. This switch is designed for FORTRAN users, who can specify an absolute address or (indirectly) a symbolic address. See the description of this switch in Chapter 3 for further details.

Function: Control switch

Type: Global

Default: Do not return a fatal error code to the program

## `/FIXED`

The file contains fixed-length records.

Function: File switch

Type: Global

Default: Depends on the recording-mode switch you specify. See the description of the appropriate recording-mode switch in Chapter 3 for more details.



/FORMAT:nPaw.d

The key data type is FORTRAN ASCII-NUMERIC.

nP =     Scaling factor (positive or negative number)  
a    =     D decimal floating point, double precision  
          E decimal floating point, single precision  
          F decimal floating point, single precision  
          G general  
w    =     key length  
d    =     number of decimal places in key

Function: Key descriptor switch  
Type:     Local  
Default:  No default characteristics

/FORTRAN

The associated files are FORTRAN ASCII or FORTRAN binary files.

Function: File switch  
Type:     Global  
Default:  Not /FORTRAN

/HELP

Print the help file. /HELP:SWITCHES prints switches available to SORT/MERGE users.

Function: Control switch  
Type:     Global (SCAN switch)  
Default:  No default characteristics

/INDUSTRY

Read or write a magnetic tape in industry-compatible mode. This switch is ignored for devices other than tape drives.

Function: Tape switch  
Type:     Modified position dependent (SCAN switch)  
Default:  DEC format

/KEY:n:m:x

The n is the position of the first character of the key. The m is the length of the key. The x is either ASCENDING or DESCENDING.

Function: Key switch (required)  
Type:     Global  
Default:  Default switches are:

1. /ALPHANUMERIC — if /SIGNED, /UNSIGNED, /NUMERIC, /COMPU, /COMP1, /COMP3, or /FORMAT is not specified.

2. /NUMERIC — if /SIGNED or /UNSIGNED is specified and if /ALPHA, /COMPU, /COMP1, or /COMP3 is not specified.
3. /SIGNED — if /COMPUTATIONAL, /COMP1, /COMP3, /NUMERIC, or /FORMAT is specified.

#### /LABEL:n

Specifies whether or not the file contains tape labels, and, if so, what type of tape labels. This switch is ignored for devices other than tape drives. Possible arguments are:

1. ANSI
2. DEC
3. IBM
4. NONSTANDARD
5. OMITTED
6. STANDARD

Function: Tape switch

Type: Modified position dependent (SCAN switch)

Default: /LABEL:STANDARD

#### /LEAVES:n

Specifies the size of the binary tree that SORT/MERGE is to use during the sort phase of a sort. The argument n is a decimal number that indicates how many leaves (or terminal nodes) that the binary tree is to have. This switch does not affect the merge phase of a sort, and is ignored when used with the /MERGE switch.

Function: Control switch

Type: Global

Default: Determined by SORT/MERGE's default memory allocation algorithm. See Chapter 6 for more details.

#### /MAXTEMP:n

Indicates the maximum number (n) of temporary files that can be used during a sort or merge.

Function: Control switch

Type: Global

Default: 26 temporary files

#### /MERGE

Merge the specified input files.

Function: Control switch

Type: Global

Default: No default characteristics

/MESSAGE:arg

Controls how much of an error, informational or warning, message is printed. The possible arguments are:

1. [NO]FIRST — determines whether or not the text portion of the message is printed.
2. [NO]PREFIX — determines whether the error message code (i.e., SRTxxx) is printed with the message.

You can specify the arguments in combination. For example, /MESSAGE:(NOPREFIX,FIRST) omits the error code, but prints the text of the message.

Function: Control switch

Type: Global

Default: /MESSAGE:(PREFIX,FIRST)

/NOCRLF

Indicates that both input and output files are /FIXED /ASCII records containing no carriage control characters.

Function: File switch

Type: Global

Default: Both input and output records have a carriage–return line–feed characters.

/NUMERIC

The key data type is numeric.

Function: Data type switch

Type: Key modifier switch

Default: Causes /SIGNED to be in effect (unless you specify /UNSIGNED).

/OPTION:name

Reads switches from lines beginning with SORT:name in SWITCH.INI.

Function: Control switch

Type: Global (SCAN switch)

Default: Read switches from lines beginning with SORT (unlabeled) in SWITCH.INI.

/PACKED

The key data type is packed decimal (4 bits). This switch is identical to /COMP3.

Function: Key data type switch

Type: Key modifier

Default: No default characteristics

#### /PARITY:n

Specifies the parity (n) to be used when reading or writing a magnetic tape. The possible values for the parity argument (n) are:

1. EVEN
2. ODD

Function: Tape switch

Type: Position dependent (SCAN switch)

Default: System default

#### /PHYSICAL

Suppresses local name assignments; only physical device names are searched to find the data file.

Function: Control switch

Type: Position dependent (SCAN switch)

Default: Search logical names first

#### /POSITION:n

Positions a magnetic tape before a file is read or written. The argument (n) can be positive to skip a number of files or negative to backspace a number of files. This switch is ignored for devices other than magnetic tape drives.

Function: Tape switch

Type: Local

Default: Position at the first file on the tape.

#### /PRIORITY:n

Sets the disk priority for your SORT/MERGE job. The argument (n) can be a value from -3 to 3. An argument of +3 is the highest priority; -3 is the lowest priority.

Function: Control switch

Type: Position dependent

Default: Normally 0. The default can be changed with the DISK. monitor call. (See the *TOPS-10 Monitor Calls Manual*.)

#### /RANDOM

The associated file is a random-access file. This switch is identical to /FIXED.

Function: File switch

Type: Global

Default: Depends on the recording-mode switch you specify. See the description of the appropriate recording-mode switch in Chapter 3 for more details.

#### `/RECORD:n`

The `n` is a decimal integer specifying the record size.

Function: Record length switch (required)

Type: Global

Default: None. This switch is required.

#### `/REWIND`

Rewind the tape before the file is read or written. This switch is ignored for devices other than tape drives.

Function: Tape switch

Type: Local

Default: Don't rewind the tape before use.

#### `/RUN:filespec`

Run the specified program. Three subordinate switches can also be specified with this switch:

1. `/RUNCORE:n` — Run the program with a memory allocation of '`n`'.
2. `/RUNOFFSET:n` — Run the program at the start address + `n`.
3. `/TMPFIL:nam:'string'` — Store the string in TMPCOR core file '`nam`'. The '`nam`' can be a maximum of three characters.

Function: Control switch

Type: Global

Default: No default characteristics

#### `/SEQUENTIAL`

The file is a sequential-access file. This switch is identical to `/VARIABLE`.

Function: File switch

Type: Global

Default: Depends on the recording-mode switch you specify. See the description of the appropriate recording-mode switch in Chapter 3 for more details.

#### `/SIGNED`

The key data is signed.

Function: Key data type switch

Type: Key modifier switch

Default: In effect for `/NUMERIC`, `FORMAT`, `COMPU`, `COMP1`, and `COMP3` unless `/UNSIGNED` is specified.

## **/SIXBIT**

The recording mode of the data is SIXBIT.

Function: Recording-mode switch

Type: Global

Default: See the explanation of this switch in Chapter 3 for a description of the defaults.

## **/STANDARD**

The data should be read from or written to magnetic tape in Standard-ASCII format. This switch is ignored if any other device is used.

Function: Tape switch

Type: Modified position dependent

Default: Core-dump format

## **/SUPPRESS:n**

Specifies which types of messages to suppress. The possible arguments (n) are:

NONE	Suppress no messages.
INFORMATION	Suppress all information messages (those beginning with 'I').
WARNING	Suppress all information messages, and all warning messages (those beginning with 'W').
FATAL	Suppress all information messages, all warning messages, and all fatal error messages (those beginning with 'F').
ALL	Same as FATAL.

### **NOTE**

It is not possible to suppress messages beginning with '\$'.

Function: Control switch

Type: Global

Default: /SUPPRESS:NONE

## **/TEMP**

Allows you to specify the device names for temporary file storage. For stand-alone sorts, the number of device names is fixed at 15.

Function: Control switch

Type: Local

Default: DSK:

## /UNLOAD

Rewind and unload the tape after the file is read or written. For multi-reel files, this switch affects only the disposition of the last tape. All intermediate tapes are UNLOADED regardless of the setting of this switch. This switch is ignored for any device other than a tape drive.

Function: Tape switch

Type: Local

Default: Do not unload the last reel of a magtape file.

## /UNSIGNED

The key is unsigned.

Function: Key data type switch

Type: Local

Default: No default characteristics

## /VARIABLE

The file contains variable-length records.

Function: File switch

Type: Global

Default: Depends on the recording-mode switch you specify. See the description of the appropriate recording-mode switch in Chapter 3 for more details.

## A.1 Switches (by Function)

### Required Switches

/RECORD:  
/KEY:

### Key Data Type Switches

/ALPHA  
/COMPU  
/COMP1  
/COMP3  
/FORMAT:  
/NUMERIC  
/PACKED  
/SIGNED  
/UNSIGNED

### Recording Mode Switches

/ASCII  
/BINARY  
/EBCDIC  
/SIXBIT

### Tape Switches

/DENSITY:  
/INDUSTRY  
/LABEL:  
/PARITY:  
/POSITION:  
/REWIND  
/STANDARD  
/UNLOAD

### File Switches

/AFTER  
/ALIGN  
/BEFORE  
/BLOCKED:  
/FIXED  
/FORTRAN  
/NOCRLF  
/RANDOM  
/SEQUENTIAL  
/VARIABLE

### Control Switches

/CHECK  
/COLLATE:  
/CORE:  
/ERROR:  
/EXIT  
/FATAL:  
/HELP  
/LEAVES:  
/MAXTEMP:  
/MERGE  
/MESSAGE:  
/OPTION:  
/PHYSICAL  
/PRIORITY:  
/RUN:  
/TEMP



## A.2 Switches (by Range)

### Global Switches

/ALIGN  
/ASCII  
/BINARY  
/CHECK  
/COLLATE:  
/CORE:  
/EBCDIC  
/ERROR:  
/EXIT  
/FATAL:  
/FIXED  
/FORTRAN  
/HELP  
/KEY:  
/LEAVES:  
/MAXTEMP  
/MERGE  
/MESSAGE:  
/OPTION:  
/RANDOM  
/RECORD:  
/RUN:  
/SEQUENTIAL  
/SIXBIT  
/SUPPRESS:  
/TEMP  
/VARIABLE

### Local Switches

/AFTER  
/ALPHA  
/BEFORE  
/COMPU  
/COMP1  
/COMP3  
/FORMAT:  
/NOCRLF  
/NUMERIC  
/PACKED  
/POSITION:  
/REWIND  
/SIGNED  
/UNLOAD  
/UNSIGNED

### Modified Position Dependent Switches

/BLOCKED  
/INDUSTRY  
/LABEL:  
/STANDARD

### Position Dependent Switches

/DENSITY:  
/PARITY:  
/PHYSICAL  
/PRIORITY:

## Appendix B

### Collating Sequences and Conversion Tables

Table B-1 shows the ASCII and SIXBIT collating sequence and the conversions from ASCII to EBCDIC, SIXBIT to ASCII, and SIXBIT to EBCDIC. If the ASCII character does not convert to the same character in EBCDIC, the EBCDIC character is shown in parentheses next to the EBCDIC code. Note that the first and last 32 characters do not exist in SIXBIT. Also, the characters in the first column (NULL, SOH, STX, etc.) are control characters, which are nonprinting.

#### NOTE

All codes are specified in octal.

**Table B-1: ASCII and SIXBIT Collating Sequence and Conversion to EBCDIC**

Character	ASCII 7-bit	EBCDIC 9-bit	Character	SIXBIT	ASCII 7-bit	EBCDIC 9-bit
NUL	000	000	Space	00	040	100
SOH	001	001*	!	01	041	132
STX	002	002*	"	02	042	177
ETX	003	003*	#	03	043	173
EOT	004	067	\$	04	044	133
ENQ	005	055*	%	05	045	154
ACK	006	056*	&	06	046	120
BEL	007	057*	'	07	047	175
BS	010	026	(	10	050	115
HT	011	005	)	11	051	135
LF	012	045	*	12	052	134
VT	013	013*	+	13	053	116
FF	014	014*	,	14	054	153
CR	015	025*(NL)	-	15	055	140
SO	016	006*(LC)	.	16	056	113
SI	017	066*(UC)	/	17	057	141
DLE	020	044*(BYP)	0	20	060	360
DC1	021	024*(RES)	1	21	061	361
DC2	022	064*(PN)	2	22	062	362
DC3	023	065*(RS)	3	23	063	363
DC4	024	004*(PF)	4	24	064	364
NAK	025	075*	5	25	065	365
SYN	026	027*(IL)	6	26	066	366
ETB	027	046*(EOB)	7	27	067	367
CAN	030	052*(CM)	8	30	070	370
EM	031	031*	9	31	071	371
SUB	032	032*(CC)	:	32	072	172
ESC	033	047*(PRE)	;	33	073	136
FS	034	023*(TM)	<	34	074	114
GS	035	041*(SOS)	=	35	075	176
RS	036	040*(DS)	>	36	076	156
US	037	042*(FS)	?	37	077	157

**Table B-1 (Cont.): ASCII and SIXBIT Collating Sequence and Conversion to EBCDIC**

Character	SIXBIT	ASCII 7-bit	EBCDIC 9-bit	Character	ASCII 7-bit	EBCDIC 9-bit
@	40	100	174	a	140	171
A	41	101	301	b	141	201
B	42	102	302	c	142	202
C	43	103	303	d	143	203
D	44	104	304	e	144	204
E	45	105	305	f	145	205
F	46	106	306	g	146	206
G	47	107	307	h	147	207
H	50	110	310	i	150	210
I	51	111	311	j	151	211
J	52	112	321	k	152	221
K	53	113	322	l	153	222
L	54	114	323	m	154	223
M	55	115	324	n	155	224
N	56	116	325	o	156	225
O	57	117	326	p	157	226
P	60	120	327	q	160	227
Q	61	121	330	r	161	230
R	62	122	331	s	162	231
S	63	123	342	t	163	242
T	64	124	343	u	164	243
U	65	125	344	v	165	244
V	66	126	345	w	166	245
W	67	127	346	x	167	246
X	70	130	347	y	170	247
Y	71	131	350	z	171	250
Z	72	132	351	{	172	251
[	73	133	255*		173	300*
\	74	134	340	}	174	117
]	75	135	275	~	175	320
^	76	136	137		176	241
—	77	137	155	Delete	177	007

\* These EBCDIC codes either have no equivalent in the ASCII or SIXBIT character sets, or are referred to by difference names. They are converted to the indicated ASCII characters to preserve their uniqueness if the ASCII character is converted back to EBCDIC.

Table B-2 shows the conversion of ASCII code to SIXBIT code. The table does not show ASCII codes 000 through 037 because they all convert to SIXBIT 74 (\), except 11 (TAB) which converts to SIXBIT 00 (space).

**Table B-2: ASCII to SIXBIT Conversion**

Character	ASCII 7-bit	SIXBIT	Character	ASCII 7-bit	SIXBIT
Space	040	00	@	100	40
!	041	01	A	101	41
"	042	02	B	102	42
#	043	03	C	103	43
\$	044	04	D	104	44
%	045	05	E	105	45
&	046	06	F	106	46
'	047	07	G	107	47
(	050	10	H	110	50
)	051	11	I	111	51
*	052	12	J	112	52
+	053	13	K	113	53
,	054	14	L	114	54
-	055	15	M	115	55
.	056	16	N	116	56
/	057	17	O	117	57
0	060	20	P	120	60
1	061	21	Q	121	61
2	062	22	R	122	62
3	063	23	S	123	63
4	064	24	T	124	64
5	065	25	U	125	65
6	066	26	V	126	66
7	067	27	W	127	67
8	070	30	X	130	70
9	071	31	Y	131	71
:	072	32	Z	132	72
;	073	33	[	133	73
<	074	34	\	134	74
=	075	35	]	135	75
>	076	36	^	136	76
?	077	37	—	137	77

**Table B-2 (Cont.): ASCII to SIXBIT Conversion**

<b>ASCII code</b>	<b>ASCII character</b>	<b>SIXBIT code</b>	<b>SIXBIT character</b>
140	`	74	\
141	a	41	A
142	b	42	B
143	c	43	C
144	d	44	D
145	e	45	E
146	f	46	F
147	g	47	G
150	h	50	H
151	i	51	I
152	j	52	J
153	k	53	K
154	l	54	L
155	m	55	M
156	n	56	N
157	o	57	O
160	p	60	P
161	q	61	Q
162	r	62	R
163	s	63	S
164	t	64	T
165	u	65	U
166	v	66	V
167	w	67	W
170	x	70	X
171	y	71	Y
172	z	72	Z
173	{	73	
174		74	\
175	}	75	
176	-	74	\
177	Delete	74	\

Table B-3 shows the EBCDIC collating sequence and the conversion from EBCDIC to ASCII. When conversion is from EBCDIC to SIXBIT, it is as if the code was converted to ASCII and then from ASCII to SIXBIT.

**Table B-3: EBCDIC Collating Sequence and Conversion to ASCII**

EBCDIC code	EBCDIC character	ASCII code	ASCII character	EBCDIC code	EBCDIC character	ASCII code	ASCII character
000	NUL	000	NUL	050		134	\
001	SOH	001	SOH	051		134	\
002	STX	002	STX	052	SM	030	CAN
003	ETX	003	ETX	053	CUZ	134	\
004	PF	024	DC4	054		134	\
005	HT	011	HT	055	ENQ	005	ENQ
006	LC	016	SO	056	ACK	006	ACK
007	Delete	177	Delete	057	BEL	007	BEL
010		134	\	060		134	\
011		134	\	061		134	\
012	SMM	134	\	062		134	\
013	VT	013	VT	063		134	\
014	FF	014	FF	064	PN	022	DC2
015	CR	134	\	065	RS	023	DC3
016	SO	134	\	066	UC	017	SI
017	SI	134	\	067	EOT	004	EOT
020	DLE	134	\	070		134	\
021	DC1	134	\	071		134	\
022	DC2	134	\	072		134	\
023	TM	034	FS	073		134	\
024	RES	021	DC1	074	CU3	134	\
025	NL	015	CR	075	DC4	025	NAK
026	BS	010	BS	076	NAK	134	\
027	IL	026	SYN	077	SUB	134	\
030	CAN	134	\	100	Space	040	Space
031	EM	031	EM	101		134	\
032	CC	032	SUB	102		134	\
033	CU1	134	\	103		134	\
034	IFS	134	\	104		134	\
035	IGS	134	\	105		134	\
036	IRS	134	\	106		134	\
037	IUS	134	\	107		134	\
040	DS	036	RS	110		134	\
041	SOS	035	GS	111		134	\
042	FS	037	US	112	Cent	134	\
043		134	\	113	.	056	.
044	BYP	020	DLE	114	<	074	<
045	LF	012	LF	115	(	050	(
046	ETB	027	ETB	116	+	053	+
047	ESC	033	ESC	117		174	

**Table B-3 (Cont.): EBCDIC Collating Sequence and Conversion to ASCII**

EBCDIC code	EBCDIC character	ASCII code	ASCII character	EBCDIC code	EBCDIC character	ASCII code	ASCII character
120	&	046	&	170		134	\
121		134	\	171		140	`
122		134	\	172		072	:
123		134	\	173		043	#
124		134	\	174		100	(
125		134	\	175		47	'
126		134	\	176		075	=
127		134	\	177		042	"
130		134	\	200		134	\
131		134	\	201		141	a
132		041	!	202		142	b
133		044	\$	203		143	c
134		052	*	204		144	d
135		051	)	205		145	e
136		073	^	206		146	f
137		137	\	207		147	g
140	-	055	-	210	h	150	h
141		057	/	211		151	i
142		134	\	212		134	\
143		134	\	213		134	\
144		134	\	214		134	\
145		134	\	215		134	\
146		134	\	216		134	\
147		134	\	217		134	\
150		134	\	220		134	\
151		134	\	221		152	j
152		134	\	222		153	k
153		054	,	223		154	l
154		045	%	224		155	m
155		137	\	225		156	n
156		076	>	226		157	o
157		077	?	227		160	p
160		134	\	230	q	161	q
161		134	\	231		162	r
162		134	\	232		134	\
163		134	\	233		134	\
164		134	\	234		134	\
165		134	\	235		134	\
166		134	\	236		134	\
167		134	\	237		134	\



**Table B-3 (Cont.): EBCDIC Collating Sequence and Conversion to ASCII**

EBCDIC code	EBCDIC character	ASCII code	ASCII character	EBCDIC code	EBCDIC character	ASCII code	ASCII character
240		134	\	310	H	110	H
241		176		311	I	110	I
242	s	163	s	312		134	\
243	t	164	t	313		134	\
244	u	165	u	314		134	\
245	v	166	v	315		134	\
246	w	167	w	316		134	\
247	x	170	x	317		134	\
250	y	171	y	320		175	
251	z	172	z	321	J	112	J
252		134	\	322	K	113	K
253		134	\	323	L	114	L
254		134	\	324	M	115	M
255		133		325	N	116	N
256		134	\	326	O	117	O
257		134	\	327	P	120	P
260		175		330	Q	121	Q
261		134	\	331	R	122	R
262		134	\	332		134	\
263		134	\	333		134	\
264		134	\	334		134	\
265		134	\	335		134	\
266		134	\	336		134	\
267		134	\	337		134	\
270		134	\	340		134	\
271		134	\	341		134	\
272		134	\	342	S	123	S
273		134	\	343	T	124	T
274		134	\	344	U	125	U
275		135		345	V	126	V
276		134	\	346	W	127	W
277		134	\	347	X	130	X
300		173		350	Y	131	Y
301	A	101	A	351	Z	132	Z
302	B	102	B	352		134	\
303	C	103	C	353		134	\
304	D	104	D	354		134	\
305	E	105	E	355		134	\
306	F	106	F	356		134	\
307	G	107	G	357		134	\

**Table B-3 (Cont.): EBCDIC Collating Sequence and Conversion to ASCII**

<b>EBCDIC code</b>	<b>EBCDIC character</b>	<b>ASCII code</b>	<b>ASCII character</b>	<b>EBCDIC code</b>	<b>EBCDIC character</b>	<b>ASCII code</b>	<b>ASCII character</b>
360	0	060	1	370	8	070	8
361	1	061	1	371	9	071	9
362	2	062	2	372		134	\
363	3	063	3	373		134	\
364	4	064	4	374		134	\
365	5	065	5	375		134	\
366	6	066	6	376		134	\
367	7	067	7	377		134	\



# Glossary

Term	Definition
<b>ASCII</b>	American Standard Code for Information Interchange. A 7-bit code in which textual information is recorded. The ASCII code can represent 128 distinct characters. These characters are uppercase and lowercase letters, numbers, common punctuation marks, and special control characters.
<b>Bias</b>	The ratio of a run to the tree size. The bias should be approximately 2.0 if the file being sorted is truly random. See Run.
<b>Buffer</b>	A device or area, such as external storage devices, used to temporarily hold information being transmitted between two processes.
<b>Data</b>	A general term used to denote any or all information (facts, numbers, letters, and symbols) that refers to or describes an object, idea, condition, or situation. It represents basic elements of information that can be processed by a computer.
<b>DSK</b>	The generic device name for disk devices. It is translated into the user's default path if a directory is not supplied, or into the structure on which the user's default path resides if a directory is supplied.

<b>EBCDIC</b>	Extended Binary Coded Decimal Interchange Code. An 8-bit translation code that can represent 256 distinct characters. Because TOPS-20 is a 36-bit system, EBCDIC disk files contain 9-bit bytes with the 8-bits of data right-justified. Four 9-bit bytes make a word. Standard 8-bit EBCDIC bytes are used when recording industry-compatible tapes. All references to EBCDIC disk files in this manual refer to the TOPS-20 implementation of EBCDIC.
<b>File Format</b>	The header words, control words, and control characters that are used to format the data into separate records.
<b>File Specification</b>	<p>A list of identifiers that uniquely specify a particular file. For example,</p> <pre>dev:&lt;directory&gt;filnam.ext</pre> <p>The 'dev:' specifies the device where the file is stored, the '&lt;directory&gt;' specifies the name of the directory where the file is stored, the 'filnam' is the name of the file, and '.ext' is the file extension.</p>
<b>Key</b>	The part of the record being compared in order to sort the file into the correct sequence.
<b>Label</b>	On magnetic tape, the leader or trailer record used to identify the reel. On random-access devices, the directory block used by the system.
<b>Logical Record</b>	The record that is created by some process, such as a program, but has not yet been written to a storage device, such as disk or magnetic tape. See Physical Record.
<b>Mixed-Mode Binary</b>	A binary file that is interpreted as containing any combination of numeric, alphanumeric, fixed-point binary and floating-point binary data. EBCDIC mixed-mode binary can include packed-decimal values; FORTRAN mixed-mode binary can include FORMAT data.
<b>Physical Record</b>	The record that is written to a storage device such as disk or magnetic tape. Factors such as blocking and default physical record size can generate a physical record consisting of many logical records.
<b>Record</b>	A collection of adjacent data items treated as a unit.

<b>Run</b>	The largest set of records ordered by key value that SORT/MERGE can generate within its memory limits. (As memory increases, fewer, but larger, runs occur.) For ascending (or descending) keys, a new run is created if SORT/MERGE encounters a record whose key is less (or greater) than the key value of the last record written to the temporary file. Runs are always in monotonically increasing (or decreasing) order.
<b>Scratch Device</b>	A device used to store intermediate data during a sort.
<b>Segment</b>	<p>A division or section of the user's virtual address space. There are two types of segments:</p> <p>Low Segment — starts at address zero and stops before address 400000. Program code stored in the low segment is not sharable.</p> <p>High Segment — starts at address 400000 and uses as much of the remaining address space required to hold the program code. This code is sharable.</p> <p>There are two advantages to segmenting the user's virtual address space:</p> <ol style="list-style-type: none"> <li>1. Allows the program code to be shared.</li> <li>2. Allows programs too large for the user's address space to be divided into multiple high segments. When a particular portion of code is required, it can be overlaid on (written over) the code currently stored in the high segment of the user's address space.</li> </ol>
<b>SIXBIT</b>	A 6-bit code in which textual information is recorded. It is a compressed form of the ASCII character set.
<b>Sort</b>	The results of comparisons between groups of records.
<b>Stable Sort</b>	A sort which leaves the relative order of equal keys unchanged.
<b>Standard Binary</b>	A binary file that is interpreted as containing only fixed-point and floating-point binary values.



# Index

- 36-bit, 1-13
- Abbreviation,
  - collate, 3-17
- /AFTER switch, 3-11, A-2
- /ALIGN switch, 3-11, A-2
- /ALPHA switch, A-2
- ALPHANUMERIC data type, 3-5
- /ALPHANUMERIC switch, 1-12, 3-5
- Arguments,
  - /COLLATE switch, 3-16
  - /LABEL switch, 3-24
- ASCENDING order, 1-4
- ASCII, Glossary-1
  - COBOL fixed-length, 4-5
  - COBOL variable-length, 4-9
  - fixed-length, 4-5
  - FORTRAN fixed-length, 4-6
  - FORTRAN variable-length, 4-10
  - variable-length, 4-8
- ASCII file,
  - creating, 1-1
- ASCII recording mode, 4-2
- /ASCII switch, 1-12, 3-9, A-2
- /BEFORE switch, 3-11, A-2
- Bias, Glossary-1
- Binary,
  - COBOL ASCII mixed-mode, 4-25
  - COBOL EBCDIC mixed-mode, 4-28
  - COBOL SIXBIT mixed-mode, 4-27
- Binary file formats, 4-25
- BINARY recording mode, 4-3
- /BINARY switch, 3-10, A-3
- Binary tree,
  - operation of, 6-4
- Binary with LSCW,
  - FORTRAN random, 4-31
  - FORTRAN sequential, 4-35
- Binary without LSCW,
  - FORTRAN random, 4-33
  - FORTRAN sequential, 4-37
- Blocked fixed-length EBCDIC, 4-20
- /BLOCKED switch, 3-12, A-3
- Blocked variable-length EBCDIC, 4-22
- Blocking files, 3-12
- Buffer, Glossary-1
- Calculating memory size, 6-11
- Categories,
  - switch, 3-1
- Character data,
  - sorting, 2-4
- Characters in file,
  - tab, 2-3
- /CHECK switch, 3-15, A-3
- COBOL,
  - field descriptors for, 3-7
  - using SORT/MERGE from, 2-6
- COBOL ASCII mixed-mode binary, 4-25
- COBOL binary file formats, 4-25
- COBOL EBCDIC mixed-mode binary, 4-28
- COBOL fixed-length ASCII, 4-5
- COBOL fixed-length EBCDIC, 4-17
- COBOL fixed-length SIXBIT, 4-13
- COBOL SIXBIT mixed-mode binary, 4-27
- COBOL variable-length ASCII, 4-9
- COBOL variable-length EBCDIC, 4-18
- COBOL variable-length SIXBIT, 4-15
- Codes,
  - error, 5-12
- Collate abbreviation, 3-17
- Collate completion, 3-17
- Collate equivalence, 3-16
- Collate functions, 3-16
- /COLLATE switch, 3-16, A-3
- /COLLATE switch arguments, 3-16



- Collating order,
  - key, 3-5
- Collating sequences, B-1
- Command files,
  - using, 2-5
- Command formats, 2-4
- COMP1 data type, 3-6, 3-8
- /COMP1 switch, 3-6, A-4
- COMP3 data type, 3-6, 3-9
- /COMP3 switch, 3-6, A-4
- Completion,
  - collate, 3-17
- /COMPU switch, A-4
- COMPUTATIONAL data type, 3-6, 3-8
- /COMPUTATIONAL switch, 3-6
- Considerations,
  - performance, 6-9
  - SORT/MERGE performance, 6-1
- Control switches, 3-15
- Control words in file, 2-3
- Conversion tables, B-1
- /CORE switch, 3-18, A-4
- Creating ASCII file, 1-1
- Creating multifield file, 1-1
- Data, Glossary-1
  - sorting character, 2-4
  - sorting nonalphanumeric, 2-4
  - sorting nonnumeric, 2-4
- Data type, 1-13
  - ALPHANUMERIC, 3-5
  - COMP1, 3-6, 3-8
  - COMP3, 3-6, 3-9
  - COMPUTATIONAL, 3-6, 3-8
  - FORMAT, 3-6, 3-8
  - NUMERIC, 3-5, 3-8
  - PACKED, 3-6, 3-9
- Data types,
  - key, 3-5
- /DENSITY switch, 3-23, A-4
- DESCENDING order, 1-4
- Descriptors for COBOL,
  - field, 3-7
- Descriptors for FORTRAN,
  - field, 3-7
- Determining key length, 1-2
- Determining key position, 1-2
- Determining record length, 1-2
- DSK:, Glossary-1
- EBCDIC, Glossary-2
  - blocked fixed-length, 4-20
  - blocked variable-length, 4-22
  - COBOL fixed-length, 4-17
  - COBOL variable-length, 4-18
- EBCDIC file formats, 4-17
- EBCDIC recording mode, 4-2
- /EBCDIC switch, 3-10, A-5
- Equivalence,
  - collate, 3-16
- Error codes, 5-12
- Error messages, 5-2
  - SORT/MERGE, 5-1
- /ERROR switch, 2-8, 3-19, A-5
- EXIT function, 2-4
- /EXIT switch, 2-4, 3-19, A-5
- /FATAL switch, 2-8, 3-19, A-5
- Field descriptors for COBOL, 3-7
- Field descriptors for FORTRAN, 3-7
- File,
  - control words in, 2-3
  - creating ASCII, 1-1
  - creating multifield, 1-1
  - line-sequence numbers in, 2-3
  - sorting line-sequenced, 1-6
  - sorting multifield, 1-3
  - sorting multiline, 1-11
  - tab characters in, 2-3
- File format, Glossary-2
- File formats, 4-1, 4-3
- File formats,
  - binary, 4-25
  - COBOL binary, 4-25
  - EBCDIC, 4-17
  - FORTRAN binary, 4-30
- File specification, Glossary-2
- File specifications, 2-5
- File switches, 3-11
- Files,
  - blocking, 3-12
  - merging, 2-5
  - sorting nontext, 1-12
  - using command, 2-5
- Files containing tabs,
  - sorting, 1-8
- /FIXED switch, 3-13, A-5
- Fixed-length ASCII, 4-5
  - COBOL, 4-5
  - FORTRAN, 4-6
- Fixed-length EBCDIC,
  - blocked, 4-20
  - COBOL, 4-17
- Fixed-length SIXBIT, 4-12
  - COBOL, 4-13
- Format,
  - /KEY switch, 3-4
  - message, 5-1
- FORMAT data type, 3-6, 3-8
- /FORMAT switch, 3-6, A-6

- Formats,
  - binary file, 4-25
  - COBOL binary file, 4-25
  - command, 2-4
  - EBCDIC file, 4-17
  - file, 4-1, 4-3
  - FORTRAN binary file, 4-30
- FORTRAN,
  - field descriptors for, 3-7
  - using SORT/MERGE from, 2-6
- FORTRAN binary file formats, 4-30
- FORTRAN fixed-length ASCII, 4-6
- FORTRAN random binary with LSCW, 4-31
- FORTRAN random binary without LSCW, 4-33
- FORTRAN sequential binary with LSCW, 4-35
- FORTRAN sequential binary without LSCW, 4-37
- /FORTRAN switch, 3-14, A-6
- FORTRAN variable-length ASCII, 4-10
- FORTRAN-called sort restrictions, 2-7
- FSORT program, 2-7
- Function,
  - EXIT, 2-4
  - HELP, 2-4
  - MERGE, 2-4
  - RUN, 2-4
  - SORT, 2-4
  - switches by, A-13
- Function summary,
  - SORT/MERGE, A-1
- Functions, A-1
  - collate, 3-16
- Getting started with SORT/MERGE, 1-1
- Global switches, 3-1
- HELP function, 2-4
- /HELP switch, 2-4, A-6
- How to use SORT/MERGE, 2-1
- /INDUSTRY switch, 3-24, A-6
- Information needed to sort files, 2-2
- Introduction, 2-1
- Key, Glossary-2
- Key collating order, 3-5
- Key data types, 3-5
- Key length, 3-5
  - determining, 1-2
- Key position,
  - determining, 1-2
- Key sign status, 3-8
- /KEY starting position, 3-4
- /KEY switch, 3-3, A-6
- /KEY switch format, 3-4
- Keys,
  - sorting on two, 1-6
- Label, Glossary-2
- /LABEL switch, 3-24, A-7
- /LABEL switch arguments, 3-24
- /LEAVES switch, 3-20, A-7
- Length,
  - determining key, 1-2
  - determining record, 1-2
  - key, 3-5
- Line-sequence numbers in file, 2-3
- Line-sequenced file,
  - sorting, 1-6
- Local switches, 3-1
- Logical record, Glossary-2
- Lowercase text,
  - sorting, 1-11
- LSCW,
  - FORTRAN random binary with, 4-31
  - FORTRAN random binary without, 4-33
  - FORTRAN sequential binary with, 4-35
  - FORTRAN sequential binary without, 4-37
- /MAXTEMP switch, 3-20, A-7
- Memory required, 2-2
- Memory size, 6-10
  - calculating, 6-11
- MERGE function, 2-4
- Merge passes,
  - number of, 6-14
- Merge phase, 2-1, 6-3
- /MERGE switch, 2-4, A-7
- Merging files, 2-5
- Message format, 5-1
- /MESSAGE switch, 3-21, A-8
- Messages,
  - error, 5-2
  - SORT/MERGE error, 5-1
- Mixed-mode binary, Glossary-2
  - COBOL ASCII, 4-25
  - COBOL EBCDIC, 4-28
  - COBOL SIXBIT, 4-27
- Mode,
  - ASCII recording, 4-2
  - BINARY recording, 4-3
  - EBCDIC recording, 4-2
  - SIXBIT recording, 4-2
- Mode switches,
  - recording, 3-9

- Modes,
  - recording, 1-14, 4-1
- Modified position dependent switches, 3-2
- Multifield file,
  - creating, 1-1
  - sorting, 1-3
- Multiline file,
  - sorting, 1-11
- /NOCRLF switch, 3-14, A-8
- Nonalphanumeric data,
  - sorting, 2-4
- Nonnumeric data,
  - sorting, 2-4
- Nontext files,
  - sorting, 1-12
- Number of merge passes, 6-14
- Numbers in file,
  - line-sequence, 2-3
- NUMERIC data type, 3-5, 3-8
- /NUMERIC switch, 3-5, A-8
- Operation of binary tree, 6-4
- /OPTION switch, 3-21, A-8
- Order,
  - ASCENDING, 1-4
  - DESCENDING, 1-4
  - key collating, 3-5
- Overview,
  - performance, 6-1
- PACKED data type, 3-6, 3-9
- /PACKED switch, 3-6, A-8
- /PARITY switch, 3-25, A-9
- Passes,
  - number of merge, 6-14
- Performance considerations, 6-9
  - SORT/MERGE, 6-1
- Performance overview, 6-1
- Phase,
  - merge, 2-1, 6-3
  - sort, 2-1, 6-2
- Physical record, Glossary-2
- /PHYSICAL switch, 3-21, A-9
- Position,
  - determining key, 1-2
  - /KEY starting, 3-4
- Position dependent switches, 3-2
  - modified, 3-2
- /POSITION switch, 3-25, A-9
- /PRIORITY switch, 3-21, A-9
- Program,
  - FSORT, 2-7
  - Random binary with LSCW,
    - FORTTRAN, 4-31
  - Random binary without LSCW,
    - FORTTRAN, 4-33
  - /RANDOM switch, 3-14, A-9
  - Range,
    - switches by, A-14
  - Record, Glossary-2
  - Record length,
    - determining, 1-2
  - /RECORD switch, 3-3, A-10
  - Recording mode switches, 3-9
  - Recording modes, 1-14, 4-1
  - Records,
    - sorting variable-length, 1-10
  - Required,
    - memory, 2-2
  - Required switches, 3-3
  - Restrictions,
    - FORTTRAN-called sort, 2-7
  - /REWIND switch, 3-26, A-10
  - Run, Glossary-3
  - RUN function, 2-4
  - /RUN switch, 2-4, 3-22, A-10
  - Running SORT/MERGE, 2-2
  - SCAN, 2-4
  - Scratch device, Glossary-3
  - Segment, Glossary-3
  - Sequences,
    - collating, B-1
  - Sequential binary with LSCW,
    - FORTTRAN, 4-35
  - Sequential binary without LSCW,
    - FORTTRAN, 4-37
  - /SEQUENTIAL switch, 3-15, A-10
  - Sign status,
    - key, 3-8
  - /SIGNED switch, A-10
  - SIXBIT, Glossary-3
    - COBOL fixed-length, 4-13
    - COBOL variable-length, 4-15
    - fixed-length, 4-12
    - variable-length, 4-14
  - SIXBIT recording mode, 4-2
  - /SIXBIT switch, 3-10, A-11
  - Size,
    - calculating memory, 6-11
    - memory, 6-10
    - tree, 6-9
  - Sort, Glossary-3
  - Sort files,
    - information needed to, 2-2

- SORT function, 2-4
- Sort phase, 2-1, 6-2
- Sort restrictions,
  - FORTTRAN-called, 2-7
- SORT/MERGE,
  - getting started with, 1-1
  - how to use, 2-1
  - running, 2-2
  - starting, 1-3
- SORT/MERGE error messages, 5-1
- SORT/MERGE from COBOL,
  - using, 2-6
- SORT/MERGE from FORTRAN,
  - using, 2-6
- SORT/MERGE function summary, A-1
- SORT/MERGE performance considerations, 6-1
- SORT/MERGE switch summary, A-1
- SORT/MERGE switches, 3-1
- Sorting character data, 2-4
- Sorting files containing tabs, 1-8
- Sorting line-sequenced file, 1-6
- Sorting lowercase text, 1-11
- Sorting multifield file, 1-3
- Sorting multiline file, 1-11
- Sorting nonalphanumeric data, 2-4
- Sorting nonnumeric data, 2-4
- Sorting nontext files, 1-12
- Sorting on two keys, 1-6
- Sorting uppercase text, 1-11
- Sorting variable-length records, 1-10
- Specifications,
  - file, 2-5
- Stable sort, Glossary-3
- Standard binary, Glossary-3
- /STANDARD switch, 3-26, A-11
- Started with SORT/MERGE,
  - getting, 1-1
- Starting position,
  - /KEY, 3-4
- Starting SORT/MERGE, 1-3
- Status,
  - key sign, 3-8
- Summary,
  - SORT/MERGE function, A-1
  - SORT/MERGE switch, A-1
- /SUPPRESS switch, 3-22, A-11
- Switch,
  - /AFTER, 3-11, A-2
  - /ALIGN, 3-11, A-2
  - /ALPHA, A-2
  - /ALPHANUMERIC, 1-12, 3-5
  - /ASCII, 1-12, 3-9, A-2
  - /BEFORE, 3-11, A-2
- Switch (Cont.),
  - /BINARY, 3-10, A-3
  - /BLOCKED, 3-12, A-3
  - /CHECK, 3-15, A-3
  - /COLLATE, 3-16, A-3
  - /COMP1, 3-6, A-4
  - /COMP3, 3-6, A-4
  - /COMPU, A-4
  - /COMPUTATIONAL, 3-6
  - /CORE, 3-18, A-4
  - /DENSITY, 3-23, A-4
  - /EBCDIC, 3-10, A-5
  - /ERROR, 2-8, 3-19, A-5
  - /EXIT, 2-4, 3-19, A-5
  - /FATAL, 2-8, 3-19, A-5
  - /FIXED, 3-13, A-5
  - /FORMAT, 3-6, A-6
  - /FORTRAN, 3-14, A-6
  - /HELP, 2-4, A-6
  - /INDUSTRY, 3-24, A-6
  - /KEY, 3-3, A-6
  - /LABEL, 3-24, A-7
  - /LEAVES, 3-20, A-7
  - /MAXTEMP, 3-20, A-7
  - /MERGE, 2-4, A-7
  - /MESSAGE, 3-21, A-8
  - /NOCRLF, 3-14, A-8
  - /NUMERIC, 3-5, A-8
  - /OPTION, 3-21, A-8
  - /PACKED, 3-6, A-8
  - /PARITY, 3-25, A-9
  - /PHYSICAL, 3-21, A-9
  - /POSITION, 3-25, A-9
  - /PRIORITY, 3-21, A-9
  - /RANDOM, 3-14, A-9
  - /RECORD, 3-3, A-10
  - /REWIND, 3-26, A-10
  - /RUN, 2-4, 3-22, A-10
  - /SEQUENTIAL, 3-15, A-10
  - /SIGNED, A-10
  - /SIXBIT, 3-10, A-11
  - /STANDARD, 3-26, A-11
  - /SUPPRESS, 3-22, A-11
  - /TEMP, 3-23, A-11
  - /UNLOAD, 3-26, A-12
  - /UNSIGNED, A-12
  - /VARIABLE, 3-15, A-12
- Switch arguments,
  - /COLLATE, 3-16
  - /LABEL, 3-24
- Switch categories, 3-1
- Switch format,
  - /KEY, 3-4

- Switch summary,
  - SORT/MERGE, A-1
- Switches, A-2
  - control, 3-15
  - file, 3-11
  - global, 3-1
  - local, 3-1
  - modified position dependent, 3-2
  - position dependent, 3-2
  - recording mode, 3-9
  - required, 3-3
  - SORT/MERGE, 3-1
  - tape, 3-23
- Switches by function, A-13
- Switches by range, A-14
- Tab characters in file, 2-3
- Tables,
  - conversion, B-1
- Tabs,
  - sorting files containing, 1-8
- Tape switches, 3-23
- /TEMP switch, 3-23, A-11
- Text,
  - sorting lowercase, 1-11
  - sorting uppercase, 1-11
- Tree,
  - operation of binary, 6-4
- Tree size, 6-9
- Two keys,
  - sorting on, 1-6
- Type,
  - data, 1-13
- /UNLOAD switch, 3-26, A-12
- /UNSIGNED switch, A-12
- Uppercase text,
  - sorting, 1-11
- Use SORT/MERGE,
  - how to, 2-1
- Using command files, 2-5
- Using SORT/MERGE from COBOL, 2-6
- Using SORT/MERGE from FORTRAN, 2-6
- /VARIABLE switch, 3-15, A-12
- Variable-length ASCII, 4-8
  - COBOL, 4-9
  - FORTRAN, 4-10
- Variable-length EBCDIC,
  - blocked, 4-22
  - COBOL, 4-18
- Variable-length records,
  - sorting, 1-10
- Variable-length SIXBIT, 4-14
  - COBOL, 4-15
- Words in file,
  - control, 2-3

### READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

---

---

---

---

---

---

---

---

Did you find errors in this manual? If so, specify the error and the page number.

---

---

---

---

---

---

---

---

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) \_\_\_\_\_

Name \_\_\_\_\_ Date \_\_\_\_\_

Organization \_\_\_\_\_ Telephone \_\_\_\_\_

Street \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip Code \_\_\_\_\_  
or Country

----- Do Not Tear — Fold Here and Tape -----

**digital**



No Postage  
Necessary  
if Mailed in the  
United States

**BUSINESS REPLY MAIL**

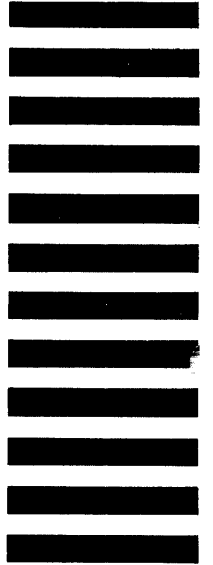
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

**SOFTWARE PUBLICATIONS**

200 FOREST STREET MR1-2/L12

MARLBOROUGH, MASSACHUSETTS 01752



----- Do Not Tear — Fold Here and Tape -----

Cut Along Dotted Line