VMS

digital

VMS Sort/Merge Utility Manual

Order Number AA–LA09A–TE

VMS Sort/Merge Utility Manual

Order Number: AA–LA09A–TE

This document describes the VMS Sort/Merge Utility.

Revision/Update Information:	This manual supersedes the *VAX/VMS Sort/Merge Utility Reference Manual,* Version 4.0.
Software Version:	VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

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

ZK4558

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*	**CANADA**	**INTERNATIONAL**
Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario K2K 2A6 Attn: Direct Order Desk	Digital Equipment Corporation PSG Business Manager c/o Digital's local subsidiary or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

PREFACE ix

NEW AND CHANGED FEATURES xi

SORT/MERGE Description SORT–1

1 SORTING RECORDS SORT–1
1.1 Multiple Keys SORT–2
1.2 Equal Keys SORT–2
1.3 Output File Organization SORT–3
1.4 Sorting Process SORT–3

2 MERGING FILES SORT–3

3 COLLATING SEQUENCE SORT–4

4 SORT/MERGE SPECIFICATION FILE SORT–5
4.1 Converting Version 3.0 Specification Files SORT–6
4.2 Entering Terminal Input SORT–6
4.3 Batch SORT/MERGE Job SORT–6

5 OPTIMIZATION SORT–7
5.1 User Optimization Options SORT–7
5.2 Assigning Work Files To Different Devices SORT–7
5.3 Working Set Extent SORT–7
5.4 SORT Optimization Qualifiers SORT–8
5.5 System Manager Optimization Options SORT–8

6 SUMMARY OF SORT/MERGE COMMANDS SORT–9

SORT/MERGE Usage Summary SORT–10

SORT/MERGE Qualifiers		SORT–11
	/CHECK_SEQUENCE	SORT–12
	/COLLATING_SEQUENCE	SORT–13
	/DUPLICATES	SORT–25
	/KEY	SORT–26
	/PROCESS	SORT–29
	/SPECIFICATION	SORT–31
	/STABLE	SORT–32
	/STATISTICS	SORT–33
	/WORK_FILES	SORT–35
Input File Qualifiers		SORT–36
	/FORMAT	SORT–37
Output File Qualifiers		SORT–39
	/ALLOCATION	SORT–40
	/BUCKET_SIZE	SORT–41
	/CONTIGUOUS	SORT–42
	/FORMAT	SORT–43
	/INDEXED_SEQUENTIAL	SORT–45
	/OVERLAY	SORT–46
	/RELATIVE	SORT–47
	/SEQUENTIAL	SORT–48
Specification File Qualifiers		SORT–49
	/CDD_PATH_NAME	SORT–50
	/CHECK_SEQUENCE	SORT–51
	/COLLATING_SEQUENCE	SORT–52
	/CONDITION	SORT–57
	/DATA	SORT–60
	/FIELD	SORT–62
	/INCLUDE	SORT–64
	/KEY	SORT–66
	/OMIT	SORT–68
	/PAD	SORT–69
	/PROCESS	SORT–70
	/STABLE	SORT–71
	/WORK_FILES	SORT–72

INDEX

TABLES

SORT–1 ASCII Collating Sequence SORT–14

SORT–2 DEC Multinational Collating Sequence SORT–17

Preface

Intended Audience

This manual is intended for all system users.

Document Structure

This document consists of the following sections:

- Description—Provides a full description of the Sort/Merge Utility (SORT).
- Usage Summary—Outlines the following SORT information:
  - Invoking the utility
  - Exiting from the utility
  - Directing output
  - Restrictions or privileges required
- Qualifiers—Describes SORT qualifiers, including format, parameters, and examples.
- Commands—Describes SORT commands, including format, parameters, and examples.

Associated Documents

To use the Sort/Merge Utility, you should also be familiar with the following manuals:

- *Introduction to VMS*
- *VMS DCL Dictionary*

Conventions

Convention	Meaning
$\boxed{\text{RET}}$	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

New and Changed Features

The following enhancements have been made to VMS Sort/Merge Utility for Version 5.0:

- Faster performance overall for all types of files; however, the biggest gain in performance is for sequential output files with fixed or variable record formats.
- SORT takes advantage of larger working sets.
- The /COLLATING_SEQUENCE=cs-name qualifier supports the National Character Set (NCS). You can specify the name of an NCS collating sequence.

SORT/MERGE Description

The VMS Sort/Merge Utility can perform two different functions:

1 Sort records from one or more input files according to the fields you select and generate one reordered output file.

2 Merge up to ten input files that have been previously sorted according to the same key fields and generate one output file.

1 Sorting Records

The Sort/Merge Utility arranges records from input files according to the keys you specify. The records can be arranged in ascending or descending alphabetic or numeric order depending on how the key is described. For example, each record in the file NAMES.DAT consists of two fields — a customer name field and an account number field. The customer name field begins in the first byte of the record, is 13 letters long, and contains character data; the account number field begins in position 16 of the record, is 5 digits long, and contains decimal data.

```
          NAMES.DAT
        Customer Name     Account Number
           (1-13)            (16-20)

            .--------------------.
            |McKee Michael  64388|
            |Erickson Sam   72931|
            |Wilson Brent   16639|
            |Coolidge Sue   98034|
            |Rice Anne      75373|
            .--------------------.
```

To create a new file, BILLING.LIS, with the records from NAMES.DAT arranged by account number, you must decribe the account number key field with the /KEY qualifier.

The /KEY qualifier specifies the following information:

- Position of the key in a record
- Size of the key
- Data type of the key
- Order of the sort operation

In addition to describing the key field, you must also pass SORT the input file specification, followed by the output file specification. The following SORT command sorts the input file, NAMES.DAT, according to the account number key field and creates the output file, BILLING.LIS:

```
$ SORT /KEY=(POSITION:16,SIZE:5,DECIMAL) NAMES.DAT BILLING.LIS
```

```
      NAMES.DAT                              BILLING.LIS

.--------------------.              .--------------------.
|McKee Michael  64388|              |Wilson Brent   16639|
|Erickson Sam   72931|  ======>     |McKee Michael  64388|
|Wilson Brent   16639|              |Erickson Sam   72931|
|Coolidge Sue   98034|              |Rice Anne      75373|
|Rice Anne      75373|              |Coolidge Sue   98034|
.--------------------.              .--------------------.
```

By default SORT assumes a key field that

- Begins in the first position of a record
- Includes the entire record
- Contains character data
- Is sorted in ascending order

To create an output file, BILLING.LIS, with the records ordered alphabetically by customer name, you would use the customer name field as the key. Because entire records will be sorted and the customer name key contains character data that begins in the first position of the record, you do not need to describe the key. You need only pass the input and output file specifications, as in the following command:

```
$ SORT NAMES.DAT BILLING.LIS
```

Note that, whenever you sort on a key field that is different from the default, you must describe that key field with the /KEY qualifier. For more details concerning keys, see the description of the /KEY qualifier.

1.1 Multiple Keys

You can sort with more than one key. Multiple keys must be specified in the order of their priority — the primary key specified first, the secondary key next, and so on. For example, NAMES.DAT is sorted first on the account number key and then by the customer name key with the following command:

```
$  SORT /KEY=(POSITION:16,SIZE:5,DECIMAL) -
_$ /KEY=(POSITION:1,SIZE:16,CHARACTER) NAMES.DAT BILLING.LIS
```

1.2 Equal Keys

If two or more records have equal key fields, the order of those records in the output file is unpredictable. By specifying the /STABLE qualifier, you can maintain the input order of the records. If two or more records have equal fields and if you want to retain only one of the records, specify the /NODUPLICATES qualifier. /STABLE and /NODUPLICATES cannot be specified on the same command line. For more details, see the description of the /STABLE and /NODUPLICATES qualifiers.

1.3 Output File Organization

By default, the output file has the same organization as that of the first input file. If the organization of the output file is to differ from that of the input file, you must specify the appropriate qualifier after the output file specification on the command line.

Qualifier	Purpose
/FORMAT	Specifies record format
/INDEXED_SEQUENTIAL	Specifies indexed-sequential file organization
/RELATIVE	Specifies relative file organization
/SEQUENTIAL	Specifies sequential file organization

If the organization of the output file is indexed-sequential, the file must already exist and must be empty. You must also qualify the output file parameter with /OVERLAY.

For more details, see the descriptions of these qualifiers.

1.4 Sorting Process

SORT arranges files by the following processes:

- Record
- Tag
- Address
- Indexed

By default, SORT arranges files by a record process that keeps records intact when sorting them and produces an output file of complete records. To specify a sorting process, use the /PROCESS qualifier. For a comparison of the record, tag, address, and indexed processes, see the /PROCESS description in the Command Qualifiers section.

The /PROCESS qualifier can be used only with SORT; you cannot specify a process for MERGE.

2 Merging Files

The Sort/Merge Utility combines up to ten sorted files into one ordered output file. All the input files must have the same format and must have been sorted on the same key fields.

For example, BILLING1.LIS and BILLING2.LIS, sorted on the same key, are merged to create one output file MAILING.LIS, which contains all the records from both files. The following MERGE command, containing the input and output file specifications, creates the output file MAILING.LIS:

```
$ MERGE BILLING1.LIS,BILLING2.LIS MAILING.LIS
```

```
 BILLING1.LIS
.--------------------.
|Coolidge Sue   98034|
|Erickson Sam   72931|              MAILING.LIS
|McKee Michael  64388|
.--------------------.            .--------------------.
                                  |Brown Thomas   23581|
                                  |Coolidge Sue   98034|
  BILLING2.LIS        ========>   |Erickson Sam   72931|
.--------------------.            |McKee Michael  64388|
|Brown Thomas   23581|            |Waters Mary    44567|
|Waters Mary    44567|            |Woo Lee        99807|
|Woo Lee        99807|            .--------------------.
.--------------------.
```

A merge operation is specified in the same way as a sort operation with two exceptions:

- You cannot specify a process for a merge operation.
- The /CHECK_SEQUENCE qualifier is used only for a merge operation.

By default, MERGE checks the order of the records. If any records are out of order, a diagnostic message will be issued and processing will continue.

3 Collating Sequence

SORT/MERGE arranges character data according to the following collating sequences:

- ASCII
- EBCDIC
- MULTINATIONAL

By default, SORT arranges character data according to the ASCII collating sequence. When SORT collates data according to the EBCDIC sequence, the output file is arranged according to the EBCDIC sequence, but the data remains in ASCII representation. The MULTINATIONAL sequence collates the international character set and treats uppercase and lowercase letters in a different manner than the ASCII collating sequence. Table SORT–2 lists the MULTINATIONAL collating sequences.

See the /COLLATING_SEQUENCE description in the Command Qualifiers Section for more information on the ASCII, EBCDIC, and MULTINATIONAL collating sequences.

You can also define your own collating sequence or modify any of the three collating sequences through the use of a specification file. See the /COLLATING_SEQUENCE description in the Specification File Qualifiers section.

4 SORT/MERGE Specification File

A SORT/MERGE specification file allows you to specify instructions that accomplish the following:

- Select records to be included in the sort/merge operation
- Reformat the records in the output file
- Use conditional keys or data
- Specify multiple record formats
- Create or modify a collating sequence
- Reassign work files
- Use frequently used sort/merge operations

You use a text editor to create a specification file. Many of the qualifiers used in the specification file are similar to the DCL qualifiers used in the command line. Note, however, that the syntax of these qualifiers can be different. For example, the /KEY qualifier at DCL level has different syntax than the /KEY qualifier in the specification file . Each of the qualifiers is explained in detail in the Specification File Qualifiers section.

Any DCL command qualifiers that you specify on the command line override corresponding entries in the specification file. For example, if you specify the /KEY qualifier in the DCL command line, SORT/MERGE ignores the /KEY clause in the specification file.

Generally, there is no required order in which you must specify the qualifiers in a specification file. The order does become significant, however, in the following cases:

- sorting on more than one key field
- describing the output format
- defining multiple record types

When you specify the FOLD, MODIFICATION, and IGNORE keywords with the /COLLATING_SEQUENCE, you should specify all MODIFICATION and IGNORE clauses before any FOLD clauses. See the /COLLATING_SEQUENCE description in the Specification File Qualifiers section for more details.

You can include comments in your specification file by beginning each comment line with an exclamation point. Unlike DCL command lines, specification files do not need hyphens to continue the line.

After you create the text of the specification file, pass the file name with the /SPECIFICATION qualifier. The default file type for a specification file is SRT.

SORT/MERGE Description

4.1 Converting Version 3.0 Specification Files

You can convert specification files that used the format of previous versions of SORT/MERGE by using the sort translation utility. First, define the symbol SORTTRANS with the following:

```
SORTTRANS :== "$SYS$SYSTEM:SRTTRN"
```

Then specify the following command line:

```
$ SORTTRANS input-file/Snn output-file
```

The input-file parameter specifies the name of the specification file from a previous version (Version 3.0 or earlier). The /Snn qualifier specifies which type of specification file is being converted. Old SORT–32 specification files are specified as /S32; old PDP–11 specification files are specified as /S11. By default, SORTTRANS assumes the /S11 qualifier.

The output-file parameter specifies the name of the specification file that the translator utility created with the new format. There is no default file type for the input file; the default for the new output specification file is SRT.

The translator utility includes the specifications from the old format in the file with the new format, setting them off with comment markers. Any errors found during the conversion of the old specification file are displayed on SYS$OUTPUT.

4.2 Entering Terminal Input

If you want to provide input to SORT interactively at a terminal, specify SYS$INPUT as the input file, qualifying it with the size of the longest record (in bytes) and the approximate size of the input file (in blocks). After you enter the command, enter the input records. Terminate each record by pressing RETURN, and terminate the file by pressing CTRL/Z. For example:

```
$  SORT /KEY=(POSITION:8,SIZE:20)_
_$ SYS$INPUT /FORMAT=(RECORD_SIZE=27, FILE_SIZE=10) NAME.LST
B 2376 FRENCH MARILYN
V 3899 ROSE JACK
     .
     .
     .
*EXIT*
```

This sequence of commands creates the output file NAME.LST, which contains the sorted records.

4.3 Batch SORT/MERGE Job

You can run a sort or merge operation as a batch job, thus freeing your terminal for other work. Create a DCL command procedure containing the sort or merge commands and execute the procedure by using the SUBMIT command. See the *VMS DCL Dictionary* for more information concerning batch jobs.

5 Optimization

If you are dissatisfied with the efficiency of your sort or merge operation, you can investigate a number of possibilities to increase efficiency. First, you should evaluate the variables in your sorting environment by using the /STATISTICS qualifier. After you examine the statistics display, consider any of the optimization options presented in the following subsections.

5.1 User Optimization Options

You can improve SORT efficiency by using a specification file. You can use /CONDITION, /INCLUDE, and /OMIT clauses to include only those records needed in the output file. Specification file clauses can also be used to reformat records to omit unnecessary fields from the output records.

See the Specification File Qualifiers section for more details.

5.2 Assigning Work Files To Different Devices

You can also increase SORT efficiency by assigning the location of the work files to random-access, mass-storage devices such as disks. Normally, SORT places work files on the device SYS$SCRATCH. However, placing work files on separate disk-structured devices permits overlap of SORT's read/write cycle. For greatest sorting efficiency, place work files on devices having the following attributes:

- The fastest devices available
- The devices having the least activity
- The devices having the most space available
- Separate devices

At DCL level, you can select a different device for any work file by specifying:

```
ASSIGN device: SORTWORKn
```

SORTWORKn is a logical name for the work file, where *n* indicates the number of the work file. Acceptable values are 0 through 9. For example, when you specify /WORK_FILES=3, use SORTWORK0, SORTWORK1, and SORTWORK2 in assigning these three work files to other devices.

You can also use the /WORK_FILES clause of the specification file to assign work files.

5.3 Working Set Extent

For the sort phase, SORT allocates enough memory to hold the input file, but not more than the working set extent. During the sort phase, records are read into this memory until it is full. If the memory cannot hold all the records, SORT transfers sorted strings to a work file on a temporary storage device.

When a sort operation uses work files, as in most sorts of large files, a larger working set increases sort efficiency. The larger the sort data structure, the greater chance SORT has of creating longer and fewer sorted strings in the work file. Fewer strings mean fewer merge passes and a faster sort.

Sometimes less memory is desirable. If the system is heavily used, the VMS operating system may be unable to allocate all the pages in the working set extent to your process. To avoid excessive paging in a heavily used system, you could make the working set extent lower. A better method is to misstate the allocated size of the input file with the /FORMAT=(FILE_SIZE:n) qualifier to a value slightly less than the working set extent that you expect your process will receive. This method causes SORT to underestimate its memory needs, use less memory, and take fewer page faults on a heavily used system.

5.4 SORT Optimization Qualifiers

The Sort/Merge Utility provides the following qualifiers that enable you to optimize your sort operation:

/STATISTICS	Displays a statistical summary
/ALLOCATION	Allocates space for output file
/CONTIGUOUS	Allocates contiguous space for output file when used with /ALLOCATION
/BUCKET_SIZE	Specifies bucket size
/FORMAT	Specifies input file size or output file format
/WORK_FILES	Sets the number of work files
/PROCESS	Specifies a different sorting process

Each of these qualifiers is explained in the Command Qualifiers section.

5.5 System Manager Optimization Options

One way the system manager can improve SORT's efficiency is to designate one batch queue for sorting jobs and to give this queue a very large working set quota.

In addition, job process parameters can be adjusted for greatest SORT efficiency. The values recommended below are based solely on SORT considerations; the system manager should integrate other system considerations with these to determine appropriate values.

- Working set extent per process (WSEXTENT and WSAUTHEXT) — For maximum sorting efficiency, this value should be set to the largest value any sorting job would ever need. Generally, the smaller the working set, the slower the sort. For very large files, working sets of 4000 or more pages are not at all unreasonable, provided the system has enough physical memory to support them. To prevent users from monopolizing real memory, the system manager can set individual maximums on a per-user basis through the authorization file.
- Virtual page count per process (VIRTUALPAGECNT) — For maximum sorting efficency, this parameter should be set to at least 1000 pages more than the maximum working set extent. Although SORT limits the size of the sort data structure to the process's working set extent, if work files are required, SORT can use the extra memory for I/O buffers. If this parameter is too low relative to the working set extent, SORT may be unable to create buffers for the work files when the files cannot be sorted in memory.

6 Summary of SORT/MERGE Commands

SORT Command Qualifiers	Defaults
/COLLATING_SEQUENCE=sequence	*/COLLATING_SEQUENCE=ASCII*
/DUPLICATES	*/NODUPLICATES*
/KEY=(field [,...])	*See text.*
/PROCESS=type	*/PROCESS=RECORD*
/SPECIFICATION=filespec	*See text.*
/STABLE	*/NOSTABLE*
/STATISTICS	*/NOSTATISTICS*
/WORK_FILES[=n]	*/WORK_FILES=2*

Input-file Qualifier	Default
/FORMAT=(file-attribute [,...])	*None.*

Output-file Qualifiers	Defaults
/ALLOCATION=n	*None.*
/BUCKET_SIZE=n	*None.*
/CONTIGUOUS	*None.*
/FORMAT=(record-format [,...])	*None.*
/INDEXED_SEQUENTIAL	*None.*
/OVERLAY	*None.*
/RELATIVE	*None.*
/SEQUENTIAL	*None.*

MERGE Command Qualifiers	Defaults
/CHECK_SEQUENCE	*/NOCHECK_SEQUENCE*
/COLLATING_SEQUENCE=sequence	*/COLLATING_SEQUENCE=ASCII*
/DUPLICATES	*/NODUPLICATES*
/KEY=(field [,...])	*See text.*
/SPECIFICATION=filespec	*See text.*
/STABLE	*/NOSTABLE*
/STATISTICS	*/NOSTATISTICS*

Input File Qualifiers	Default
/FORMAT=(file-attribute [,...])	*None.*

Output File Qualifiers	Defaults
/ALLOCATION=n	*None.*
/BUCKET_SIZE=n	*None.*
/CONTIGUOUS	*None.*
/FORMAT=(record-format [,...])	*None.*
/INDEXED_SEQUENTIAL	*None.*
/OVERLAY	*None.*
/RELATIVE	*None.*
/SEQUENTIAL	*None.*

SORT/MERGE Usage Summary

The VMS Sort/Merge Utility sorts records or merges input files. To sort one or more input files, specify the SORT command. These files are sorted according to the fields you select and one reordered output file is generated. To merge up to 10 input files that have previously been sorted according to the same key fields, specify the MERGE command. One output file is generated.

FORMAT

SORT *input-files output-file*

PARAMETERS

input-files
Specifies the files to be sorted. Up to 10 input files can be sorted to create one output file. Multiple input file specifications must be separated by commas. The default file type is DAT.

output-file
Specifies the file to be created. Only one file can be specified. If you omit a file type in the file specification, SORT defaults to the file type of the first input file.

FORMAT

MERGE *input-files output-file*

PARAMETERS

input-files
Specifies the files to be merged. Up to ten files, presorted by the same keys, can be specified. Multiple file specifications must be separated by commas. The default file type is DAT.

output-file
Specifies the merged file to be created. Only one output file can be specified. If you omit a file type in the file specification, MERGE defaults to the file type of the first input file.

usage summary

Both the SORT and MERGE commands invoke the VMS Sort/Merge Utility. After completing an operation, the Sort/Merge Utility exits and returns the user to DCL command level. You can specify where you want the results of a SORT/MERGE operation with the output file parameter.

SORT/MERGE QUALIFIERS

This section describes the qualifiers to the SORT and MERGE commands. These qualifiers enable you to define your key fields, to describe the data in those fields, and to specify various SORT options.

/CHECK_SEQUENCE

Verifies the sequence of the records in MERGE input files. By default, the sequence of records is checked.

FORMAT

/CHECK_SEQUENCE
/NOCHECK_SEQUENCE

DESCRIPTION

The /CHECK_SEQUENCE qualifier is unique to MERGE. By default, MERGE does sequence checking to ensure that the input files have been sorted on the same key.

You can also use the /CHECK_SEQUENCE qualifier to check whether the records of one or more files (up to 10) have been sorted. (The records will still be directed to an output file, which you must specify.) If you are checking whether records are sorted on a key field other than the entire record, you must specify key information, along with requesting sequence.

EXAMPLES

1
```
$ MERGE/KEY=(SIZE:4,POSITION:3)/NOCHECK_SEQUENCE  PRICE1.DAT-
_$ PRICE2.DAT PRICE.LIS
```

The /NOCHECK_SEQUENCE qualifier specifies that sequence of the input files, PRICE1.DAT and PRICE2.DAT, need not be checked because the records in those files were sorted on the same key and the sequence of records is correct.

2
```
$ MERGE/SPECIFICATION=PAYROLL.SRT/CHECK_SEQUENCE -
_$ MAY3.DAT,MAY10.DAT,MAY17.DAT,MAY24.DAT TOTAL.LIS
```

Because the specification file, PAYROLL.SRT, specifies no sequence checking, the /CHECK_SEQUENCE qualifier on the MERGE command line is necessary to override that instruction. The sequence of records in the four input files are to be checked.

/COLLATING_SEQUENCE

Selects one of three predefined collating orders for character key fields, or specifies the name of a National Character Set (NCS) collating sequence to be used in comparing character keys. SORT arranges characters in ASCII sequence by default; the EBCDIC and MULTINATIONAL sequences can also be used.

FORMAT

/COLLATING_SEQUENCE=type
/COLLATING_SEQUENCE=cs-name

PARAMETERS

ASCII

Arranges characters according to ASCII sequence (see Table SORT–1). ASCII is the default sequence and need not be specified.

EBCDIC

Arranges characters according to EBCDIC sequence. The characters remain in ASCII representation; only the order is changed.

MULTINATIONAL

Arranges characters according to MULTINATIONAL sequence (see Table SORT–2), which collates the international character set. When you use the MULTINATIONAL sequence, characters are ordered according to the following rules:

- All diacritical forms of a character are given the collating value of the character (A′, A″, A` collate as A).
- Lowercase characters are given the collating value of their uppercase equivalents (a collates as A, a″ collates as A″).
- If two strings compare as equal, tie-breaking is performed. The strings are compared to detect differences due to diacritical marks, ignored characters, or characters that collate as equal although they are actually different. If the strings still compare as equal, another comparison is done based on the numeric codes of the characters. In this final comparison, lowercase characters are ordered before uppercase.

Care should be taken when sorting or merging files for further processing using the MULTINATIONAL sequence. Sequence checking procedures in most programming languages compare numeric characters. Because MULTINATIONAL is based on actual graphic characters and not on the codes representing those characters, normal sequence checking does not work.

Note that, some languages do not support MULTINATIONAL comparisons and instead can use the LIB$COMPARE_MULTI routine.

CS-NAME

Arranges character keys according to the named sequence, which must be a collating sequence defined in a VAX NCS library. See *VMS National Character Set Utility Manual* for information about the VAX NCS Utility and NCS libraries.

DESCRIPTION

By default, SORT/MERGE arranges records according to ASCII sequence. However, it can also arrange records according to EBCDIC and MULTINATIONAL sequence.

These three collating sequences can be modified to meet your particular needs through the use of a specification file. You can also define your own collating sequence by using a specification file if one of the three collating sequences does not suit your needs. See the description of the specification file for more detail.

Table SORT–1 contains the ASCII collating sequence.

Table SORT–1 ASCII Collating Sequence

ASCII	Hex	Octal	Decimal
NUL	00	000	0
SOH	01	001	1
STX	02	002	2
ETX	03	003	3
EOT	04	004	4
ENQ	05	005	5
ACK	06	006	6
BEL	07	007	7
BS	08	010	8
HT	09	011	9
LF	0A	012	10
VT	0B	013	11
FF	0C	014	12
CR	0D	015	13
SO	0E	016	14
SI	0F	017	15
DLE	10	020	16
DC1	11	021	17
DC2	12	022	18
DC3	13	023	19
DC4	14	024	20
NAK	15	025	21
SYN	16	026	22
ETB	17	027	23
CAN	18	030	24
EM	19	031	25
SUB	1A	032	26
ESC	1B	033	27
FS	1C	034	28

Table SORT–1 (Cont.) ASCII Collating Sequence

ASCII	Hex	Octal	Decimal
GS	1D	035	29
RS	1E	036	30
US	1F	037	31
SP	20	040	32
!	21	041	33
″	22	042	34
#	23	043	35
$	24	044	36
%	25	045	37
&	26	046	38
'	27	047	39
(	28	050	40
)	29	051	41
*	2A	052	42
+	2B	053	43
,	2C	054	44
_	2D	055	45
#.	2E	056	46
/	2F	057	47
0	30	060	48
1	31	061	49
2	32	062	50
3	33	063	51
4	34	064	52
5	35	065	53
6	36	066	54
7	37	067	55
8	38	070	56
9	39	071	57
:	3A	072	58
;	3B	073	59
<	3C	074	60
=	3D	075	61
>	3E	076	62
?	3F	077	63
@	40	100	64
A	41	101	65
B	42	102	66

Table SORT–1 (Cont.) ASCII Collating Sequence

ASCII	Hex	Octal	Decimal
C	43	103	67
D	44	104	68
E	45	105	69
F	46	106	70
G	47	107	71
H	48	110	72
I	49	111	73
J	4A	112	74
K	4B	113	75
L	4C	114	76
M	4D	115	77
N	4E	116	78
O	4F	117	79
P	50	120	80
Q	51	121	81
R	52	122	82
S	53	123	83
T	54	124	84
U	55	125	85
V	56	126	86
W	57	127	87
X	58	130	88
Y	59	131	89
Z	5A	132	90
[	5B	133	91
\	5C	134	92
]	5D	135	93
^	5E	136	94
-	5F	137	95
`	60	140	96
a	61	141	97
b	62	142	98
c	63	143	99
d	64	144	100
e	65	145	101
f	66	146	102
g	67	147	103
h	68	150	104

Table SORT–1 (Cont.) ASCII Collating Sequence

ASCII	Hex	Octal	Decimal
i	69	151	105
j	6A	152	106
k	6B	153	107
l	6C	154	108
m	6D	155	109
n	6E	156	110
o	6F	157	111
p	70	160	112
q	71	161	113
r	72	162	114
s	73	163	115
t	74	164	116
u	75	165	117
v	76	166	118
w	77	167	119
x	78	170	120
y	79	171	121
z	7A	172	122
{	7B	173	123
|	7C	174	124
}	7D	175	125
~	7E	176	126
DEL	7F	177	127

Table SORT–2 contains the DEC Multinational Collating Sequence.

Table SORT–2 DEC Multinational Collating Sequence

HEX Code	Octal Code	Decimal Code	Char or Abbrev.	Description
00	000	000	NUL	Null character
01	001	001	SOH	Start of heading
02	002	002	STX	Start of text
03	003	003	ETX	End of text
04	004	004	EOT	End of transmission
05	005	005	ENQ	Enquiry
06	006	006	ACK	Acknowledge
07	007	007	BEL	Bell
08	010	008	BS	Backspace

Table SORT–2 (Cont.) DEC Multinational Collating Sequence

HEX Code	Octal Code	Decimal Code	Char or Abbrev.	Description
09	011	009	HT	Horizontal tabulation
0A	012	010	LF	Line feed
0B	013	011	VT	Vertical tabulation
0C	014	012	FF	Form feed
0D	015	013	CR	Carriage return
0E	016	014	SO	Shift out
0F	017	015	SI	Shift in
10	020	016	DLE	Data link escape
11	021	017	DC1	Device control 1
12	022	018	DC2	Device control 2
13	023	019	DC3	Device control 3
14	024	020	DC4	Device control 4
15	025	021	NAK	Negative acknowledge
16	026	022	SYN	Synchronous idle
17	027	023	ETB	End of transmission block
18	030	024	CAN	Cancel
19	031	025	EM	End of medium
1A	032	026	SUB	Substitute
1B	033	027	ESC	Escape
1C	034	028	FS	File separator
1D	035	029	GS	Group separator
1E	036	030	RS	Record separator
1F	037	031	US	Unit separator
20	040	032	SP	Space
21	041	033	!	Exclamation point
22	042	034	"	Quotation marks (double quote)
23	043	035	#	Number sign
24	044	036	$	Dollar sign
25	045	037	%	Percent sign
26	046	038	&	Ampersand
27	047	039	'	Apostrophe (single quote)
28	050	040	(	Opening parenthesis
29	051	041	)	Closing parenthesis
2A	052	042	*	Asterisk
2B	053	043	+	Plus
2C	054	044	,	Comma
2D	055	045	-	Hyphen or minus

Table SORT–2 (Cont.) DEC Multinational Collating Sequence

HEX Code	Octal Code	Decimal Code	Char or Abbrev.	Description
2E	056	046	_.	Period or decimal point
2F	057	047	/	Slash
30	060	048	0	Zero
31	061	049	1	One
32	062	050	2	Two
33	063	051	3	Three
34	064	052	4	Four
35	065	053	5	Five
36	066	054	6	Six
37	067	055	7	Seven
38	070	056	8	Eight
39	071	057	9	Nine
3A	072	058	:	Colon
3B	073	059	;	Semicolon
3C	074	060	<	Less than
3D	075	061	=	Equals
3E	076	062	>	Greater than
3F	077	063	?	Question mark
40	100	064	@	Commercial at
61	141	097	a	Lowercase a
41	101	065	A	Uppercase A
E0	340	224	à	Lowercase a with grave accent
C0	300	192	À	Uppercase A with grave accent
E1	341	225	á	Lowercase a with acute acent
C1	301	193	Á	Uppercase A with acute accent
E2	342	226	â	Lowercase a with circumflex
C2	302	194	Â	Uppercase A with circumflex
E3	343	227	ã	Lowercase a with tilde
C3	303	195	Ã	Uppercase A with tilde
E4	344	228	ä	Lowercase a with umlaut,(diaeresis)
C4	304	196	Ä	Uppercase A with umlaut,(diaeresis)
62	142	098	b	Lowercase b
42	102	066	B	Uppercase B
63	143	099	c	Lowercase c
43	103	067	C	Uppercase C
E7	347	231	ç	Lowercase c with cedilla
C7	307	199	Ç	Uppercase C with cedilla

Table SORT–2 (Cont.) DEC Multinational Collating Sequence

HEX Code	Octal Code	Decimal Code	Char or Abbrev.	Description
64	144	100	d	Lowercase d
44	104	068	D	Uppercase D
65	145	101	e	Lowercase e
45	105	069	E	Uppercase E
E8	350	232	è	Lowercase e with grave accent
C8	310	200	È	Uppercase E with grave accent
E9	351	233	é	Lowercase e with acute accent
C9	311	201	É	Uppercase E with acute accent
EA	352	234	ê	Lowercase e with circumflex
CA	312	202	Ê	Uppercase E with circumflex
EB	353	235	ë	Lowercase e with umlaut,(diaeresis)
CB	313	203	Ë	Uppercase E with umlaut,(diaeresis)
66	146	102	f	Lowercase f
46	106	070	F	Uppercase F
67	147	103	g	Lowercase g
47	107	071	G	Uppercase G
68	150	104	h	Lowercase h
48	110	072	H	Uppercase H
69	151	105	i	Lowercase i
49	111	073	I	Uppercase I
EC	354	236	ì	Lowercase i with grave accent
CC	314	204	Ì	Uppercase I with grave accent
ED	355	237	í	Lowercase i with acute accent
CD	315	205	Í	Uppercase I with acute accent
EE	356	238	î	Lowercase i with circumflex
CE	316	206	Î	Uppercase I with circumflex
EF	357	239	ï	Lowercase i with umlaut,(diaeresis)
CF	317	207	Ï	Uppercase I with umlaut,(diaeresis)
6A	152	106	j	Lowercase j
4A	112	074	J	Uppercase J
6B	153	107	k	Lowercase k
4B	113	075	K	Uppercase K
6C	154	108	l	Lowercase l
4C	114	076	L	Uppercase L
6D	155	109	m	Lowercase m
4D	115	077	M	Uppercase M
6E	156	110	n	Lowercase n

Table SORT–2 (Cont.) DEC Multinational Collating Sequence

HEX Code	Octal Code	Decimal Code	Char or Abbrev.	Description
4E	116	078	N	Uppercase N
F1	361	241	ñ	Lowercase n with tilde
D1	321	209	Ñ	Uppercase N with tilde
6F	157	111	o	Lowercase o
4F	117	079	O	Uppercase O
F2	362	242	ò	Lowercase o with grave acent
D2	322	210	Ò	Uppercase O with grave accent
F3	363	243	ó	Lowercase o with acute accent
D3	323	211	Ó	Uppercase O with acute accent
F4	364	244	ô	Lowercase o with circumflex
D4	324	212	Ô	Uppercase O with circumflex
F5	365	245	õ	Lowercase o with tilde
D5	325	213	Õ	Uppercase O with tilde
F6	366	246	ö	Lowercase o with umlaut,(diaeresis)
D6	326	214	Ö	Uppercase O with umlaut,(diaeresis)
F7	367	247	œ	Lowercase oe ligature
D7	327	215	Œ	Uppercase OE ligature
70	160	112	p	Lowercase p
50	120	080	P	Uppercase P
71	161	113	q	Lowercase q
51	121	081	Q	Uppercase Q
72	162	114	r	Lowercase r
52	122	082	R	Uppercase R
73	163	115	s	Lowercase s
53	123	083	S	Uppercase S
DF	337	223	ß	German lowercase sharp s
74	164	116	t	Lowercase t
54	124	084	T	Uppercase T
75	165	117	u	Lowercase u
55	125	085	U	Uppercase U
F9	371	249	ù	Lowercase u with grave accent
D9	331	217	Ù	Uppercase U with grave accent
FA	372	250	ú	Lowercase u with acute accent
DA	332	218	Ú	Uppercase U with acute accent
FB	373	251	û	Lowercase u with circumflex
DB	333	219	Û	Uppercase U with circumflex
FC	374	252	ü	Lowercase u with umlaut,(diaeresis)

Table SORT–2 (Cont.) DEC Multinational Collating Sequence

HEX Code	Octal Code	Decimal Code	Char or Abbrev.	Description
DC	334	220	Ü	Uppercase U with umlaut,(diaeresis)
76	166	118	v	Lowercase v
56	126	086	V	Uppercase V
77	167	119	w	Lowercase w
57	127	087	W	Uppercase W
78	170	120	x	Lowercase x
58	130	088	X	Uppercase X
79	171	121	y	Lowercase y
59	131	089	Y	Uppercase Y
FD	375	253	ÿ	Lowercase y with umlaut,(diaeresis)
DD	335	221	Ÿ	Uppercase Y with umlaut,(diaeresis)
7A	172	122	z	Lowercase z
5A	132	090	Z	Uppercase Z
E6	346	230	æ	Lowercase ae diphthong
C6	306	198	Æ	Uppercase AE with diphthong
F8	370	248	ø	Lowercase o with slash
D8	330	216	Ø	Uppercase O with slash
E5	345	229	å	Lowercase a with ring
C5	305	197	Å	Uppercase A with ring
5B	133	091	[	Opening bracket
5C	134	092	\	Backslash
5D	135	093	]	Closing bracket
5E	136	094	^	Circumflex
5F	137	095	_	Underline (underscore)
60	140	096	`	Grave accent
7B	173	123	{	Opening brace
7C	174	124	|	Vertical line
7D	175	125	}	Closing brace
7E	176	126	~	Tilde
7F	177	127	DEL	Delete, rubout
84	204	132	IND	Index
85	205	133	NEL	Next line
86	206	134	SSA	Start of selected area
87	207	135	ESA	End of selected area
88	210	136	HTS	Horizontal tab set
89	211	137	HTJ	Horizontal tab set with justification
8A	212	138	VTS	Vertical tab set

Table SORT–2 (Cont.) DEC Multinational Collating Sequence

HEX Code	Octal Code	Decimal Code	Char or Abbrev.	Description
8B	213	139	PLD	Partial line down
8C	214	140	PLU	Partial line up
8D	215	141	RI	Reverse index
8E	216	142	SS2	Single shift 2
8F	217	143	SS3	Single shift 3
90	220	144	DCS	Device control string
91	221	145	PU1	Private use 1
92	222	146	PU2	Private use 2
93	223	147	STS	Set transmit state
94	224	148	CCH	Cancel character
95	225	149	MW	Message waiting
96	226	150	SPA	Start of protected area
97	227	151	EPA	End of protected area
9B	233	155	CSI	Control sequence introducer
9C	234	156	ST	String terminator
9D	235	157	OSC	Operating system command
9E	236	158	PM	Privacy message
9F	237	159	APC	Application
A1	241	161	¡	Inverted exclamation mark
A2	242	162	¢	Cent sign
A3	243	163	£	Pound sign
A5	245	165	¥	Yen sign
A7	247	167	§	Section sign
A8	250	168	¤	General currency sign
A9	251	169	©	Copyright sign
AA	252	170	ª	Feminine ordinal indicator
AB	253	171	«	Angle quotation mark left
B0	260	176	°	Degree sign
B1	261	177	±	Plus/minus sign
B2	262	178	^{2}	Superscript 2
B3	263	179	^{3}	Superscript 3
B5	265	181	μ	Micro sign
B6	266	182	¶	Paragraph sign, pilcrow
B7	267	183	•	Middle dot
B9	271	185	^{1}	Superscript 1

Table SORT–2 (Cont.) DEC Multinational Collating Sequence

HEX Code	Octal Code	Decimal Code	Char or Abbrev.	Description
BA	272	186	º	Masculine ordinal indicator
BB	273	187	»	Angle quotation mark right
BC	274	188	¼	Fraction one quarter
BD	275	189	½	Fraction one half
BF	277	191	¿	Inverted question mark

EXAMPLE

```
$ SORT/COLLATING_SEQUENCE=MULTINATIONAL -
_$ NAMES.DAT,NOM.DAT LIST.LIS
```

This SORT command arranges the input files NAMES.DAT and NOM.DAT according to the MULTINATIONAL collating sequence to create the output file LIST.LIS.

/DUPLICATES

Eliminates all but one of multiple records with duplicate keys. By default, SORT retains multiple records with duplicate keys.

FORMAT

/DUPLICATES
/NODUPLICATES

DESCRIPTION

By default, SORT/MERGE retains records with equal keys. The /NODUPLICATES qualifier eliminates all but one record with equal keys. The retained record is unpredictable. If you want to specify which duplicate record to keep, invoke SORT at the program level and specify an equal-key routine. See *VMS Utility Routines Manual* for more detail.

The /STABLE and the /NODUPLICATES qualifiers are mutually exclusive.

EXAMPLE

```
$ SORT/KEY=(POSITION:3,SIZE:5,DECIMAL)/NODUPLICATES -
_$ ACCT1,ACCT2 ACCT.LIS
```

This SORT command arranges the two input files according to the key supplied and eliminates all but one of multiple records with equal keys.

/KEY

Describes key fields, including the position, size, sorting order, and data type. By default, SORT reorders a file by sorting entire records with character data in ascending order. Any other type of key field must be specified. When specifying multiple keys, use a separate /KEY qualifier for each key.

FORMAT

/KEY=(field [,...])

QUALIFIER VALUES

POSITION:n
Specifies the position of the first byte in the key field. A value of 1 to 32,767 may be specified. The first byte in a record is considered position 1. Both the position and the size of the key field must be specified. If a decimal sign is stored in a separate byte in the key field, that byte is counted when you determine position.

SIZE:n
Specifies the length of the key field. Both the position and size of the key field must be specified. The total composite size of all keys and the original input record length must be less than 32,767 bytes. If the decimal sign is stored in a separate byte in the key field, that byte is not counted toward the size of the data.

The data type of the key determines what values are acceptable when specifying size:

- 1 to 32,767 characters for character data
- 1, 2, 4, 8, or 16 bytes for binary data
- 1 to 31 digits for decimal data
- No value is necessary for floating point data

ASCENDING
Orders the sorting operation in ascending alphabetical or numerical order. ASCENDING is the default order.

DESCENDING
Orders the sorting operation in descending alphabetical or numerical order.

CHARACTER
Specifies character data in the key field. CHARACTER is the default data type.

BINARY
Specifies binary data in the key field.

SIGNED
Specifies signed binary or decimal data in key field. SIGNED is the default for binary and decimal data.

UNSIGNED
Specifies unsigned binary or decimal data in the key field.

F_FLOATING
Specifies F_FLOATING format data in the key field.

D_FLOATING
Specifies D_FLOATING format data in the key field.

G_FLOATING
Specifies G_FLOATING format data in the key field.

H_FLOATING
Specifies H_FLOATING format data in the key field.

DECIMAL
Specifies decimal data in the key field.

TRAILING_SIGN
Specifies trailing sign decimal data in the key field. TRAILING_SIGN is the default for decimal data.

LEADING_SIGN
Specifies leading sign decimal data in the key field.

OVERPUNCHED_SIGN
Specifies overpunched decimal data in the key field. OVERPUNCHED_SIGN is the default for decimal data.

SEPARATE_SIGN
Specifies separate sign decimal data in the key field.

ZONED
Specifies zoned decimal data in the key field.

PACKED_DECIMAL
Specifies packed decimal data in the key field.

NUMBER:n
Specifies the order of priority of each key if you do not list multiple keys in the order of their priority. A value of 1 to 255 may be specified.

DESCRIPTION

The /KEY qualifier specifies all the necessary information about a key field. If the file is to be sorted using entire records with character data in ascending order, you do not need to specify the key information.

When a key field must be described, you must specify both the position and the size of the key. Also, if the sorting or merging operation is to be done in descending alphabetic or numeric order, specify DESCENDING in the key description.

If the data in the key fields is not character data, you must specify the data type. The following data types are recognized by the Sort/Merge Utility:

CHARACTER
BINARY, [SIGNED]
BINARY, UNSIGNED
F_FLOATING
D_FLOATING
G_FLOATING
H_FLOATING
ZONED
DECIMAL [,SIGNED, TRAILING_SIGN, OVERPUNCHED_SIGN]
DECIMAL, LEADING_SIGN, SEPARATE_SIGN [SIGNED]
DECIMAL, LEADING_SIGN, [OVERPUNCHED_SIGN, SIGNED]
DECIMAL, [TRAILING SIGN], SEPARATE_SIGN, [SIGNED]
DECIMAL, UNSIGNED
PACKED_DECIMAL

The items in brackets are defaults and need not be specified.

Multiple Keys

You can specify up to 255 key fields in a sorting operation. If you do specify multiple keys, decide which is primary, which is secondary, and so on; then, in the command string, list them in the order of their priority.

By default, SORT assigns 1 to the first key specified in the command line, 2 to the second key, and so on. If you do not list the keys in the order of their priority, specify the order of each with the parameter NUMBER:n.

For each sort key, you must use a separate /KEY qualifier. If SORT finds /KEY parameters repeated after a single /KEY qualifier, it does not treat these as specifications for multiple keys; instead, the duplicate parameters override previously specified parameters.

EXAMPLE

```
$ SORT/KEY=(POS:16,SIZ:3)/KEY=(POS:1,SIZ:11) -
_$ /KEY=(POS:40,SIZ:2,DESC) YRENDAVG.DAT YRAVGSRT.LIS
```

This SORT command identifies three key fields. The input file, YRENDAVG, is first sorted by the key beginning in position 16, then by the key beginning in position 1, and finally by the key beginning in position 40. The third key used sorts in descending order.

/PROCESS

Defines the internal sorting process. The /PROCESS qualifier allows you to choose one of four processes: record, tag, address, or index. By default, SORT uses a record sorting process. Use only with the SORT command.

FORMAT

/PROCESS=type

QUALIFIER VALUES

RECORD
Keeps records intact while sorting and produces an output file consisting of complete records. Record is the default sorting process.

TAG
Sorts only the keys and then rereads the input file to produce an output file consisting of complete records.

ADDRESS
Sorts only the keys and produces an output file that is an index of record addresses in binary format. The index must be submitted to a program for further processing.

INDEX
Sorts only the keys and produces an output file that is an index of keys and of record addresses in binary form. The index must be submitted to a program for further processing.

DESCRIPTION

You can choose one of four processing methods for the sorting of your data: record, tag, address, or index sort. Record sort is the default. The four processes result in varied output file formats; they also differ in input and output device requirements and processing methods.

When selecting a sort process, consider the following:

1 How you will use the output file.

- Because record and tag sorts generate files containing entire sorted records, these reordered files are ready to be used.
- Both address- and index-sorted output files can be processed by a program written in native-mode BASIC, MACRO, or BLISS.
- Address sort creates a list of pointers to the records in the input file. This list consists of binary record's file addresses (RFAs), plus a file number when sorting multiple input files. A program accesses the records by use of the pointers.
- Index sort creates an output file containing both RFAs and key fields, plus a file number when sorting multiple files. The format of these key fields is the same as in the input files. If the program needs key field content for a decision during future processing, select index sort rather than address sort.

If you need to reorder records from one file in several ways for different purposes, store several output files from address or index sort and use the files to access the records in the main file in the sorted order you want.

2 The temporary storage space available for the sort.

- Tag sort uses less temporary storage space than record sort. Because record sort keeps the record intact during the sort, it uses much more work space when the files are large.
- Address and index sort use little temporary storage space.

3 The type of input and output device used.

- Record sort is the only process that can accept input from cards, magnetic tape, and disk.
- Output from tag and record sorts can go to any output device; output from address and index sort must go to a device that accepts binary data.

4 Differences in speed.

- If you plan to retrieve the sorted records at some point in the operation, record sort is usually the fastest process.
- Because tag sort moves only keys instead of complete records, it can be faster than record sort when record size is very large and key size is small. Tag sort can also be faster for extremely large files and devices with fast seek times. In most cases, however, the time that tag sort takes to reaccess the input file to create the output file makes it slower than record sort.
- Address and index sort are the fastest processes.

EXAMPLE

```
$ SORT/KEY=(POS:40,SIZ:2,DESC)/PROCESS=TAG YRENDAVG.DAT-
_$ DESCYRAVG.LIS
```

This SORT operation uses a tag sorting process to create the output file, DESCYRAVG.LIS.

/SPECIFICATION

Identifies a SORT or MERGE specification file.

FORMAT

/SPECIFICATION=filespec

QUALIFIER VALUE

filespec
Specifies the SORT/MERGE specification file. The default file type is SRT.

DESCRIPTION

The /SPECIFICATION qualifier identifies the specification file to be used in a SORT or MERGE operation. A specification file allows you to do the following:

- Change the format and length of the records in the output file
- Conditionally alter record order and data fields
- Omit specified records from the process
- Include specified records in the process
- Change the way in which characters are ordered
- Reassign work files
- Define commonly used SORT or MERGE operations

See the Description Section and the Specification File Qualifiers Section for more information about specification files.

EXAMPLE

```
$ SORT/SPECIFICATION=ACCTS.SRT SALES1.DAT,SALES2.DAT MAILING.LIS
```

This SORT command arranges the input files according to the instructions detailed in the specification file, ACCTS.SRT.

/STABLE

Directs records with equal keys to the output file in their input file order. The default condition is /NOSTABLE.

FORMAT

/STABLE
/NOSTABLE

DESCRIPTION

When the input files contain records with equal keys, those records are grouped in an unpredictable order. Specifying the /STABLE qualifier arranges records with equal keys in the order of the input files on output. If you use this qualifier when sorting multiple input files, on output, records with equal keys in the first file precede those from the second file and so on.

The /STABLE and /NODUPLICATES qualifiers are mutually exclusive.

EXAMPLE

```
$ SORT/KEY=(POS:1,SI:5,DECIMAL)/STABLE PRICESA.DAT,PRICESB.DAT -
_$ PRICESC.DAT SUMMARY.LIS
```

In this SORT operation, records with equal keys from PRICESA.DAT will be listed first, followed by those from PRICESB.DAT, followed by those from PRICESC.DAT.

/STATISTICS

Displays a statistical summary that can be used for optimization.

FORMAT

/STATISTICS

DESCRIPTION

When the /STATISTICS qualifier is used, SORT/MERGE displays statistics in SYS$OUTPUT. You can use these statistics to judge the efficiency of the ordering operation and to determine adjustments that can improve its performance. To save these statistics in a file, specify the following command:

```
$ DEFINE/USER SYS$ERROR output-file
```

The following statistical display results when you use the /STATISTICS qualifier:

```
                  VMS Sort/Merge Statistics

Records read:        nnn               Input record length:       nnn
Records sorted:      nnn               Internal length:           nnn
Records output:      nnn               Output record length:      nnn
Working set extent:  nnn               Sort tree size:            nnn
Virtual memory:      nnn               Number of initial runs:    nnn
Direct I/O:          nnn               Maximum merge order:       nnn
Buffered I/O:        nnn               Number of merge passes:    nnn
Page faults:         nnn               Work file allocation:      nnn
Elapsed time: nn:nn:nn.nn              Elapsed CPU:       nn:nn:nn.nn
```

Records read is the number of records read by SORT or MERGE.

Records sorted is the number of records sorted. This number could be less than the number of records read if a specification file is used to select only certain records for the sort or merge operation.

Records output is the number of records written to the output file. This number could be less than the number of records sorted if /NODUPLICATES was selected, or if I/O errors occurred when the output records were being written.

Working set extent shows the number of pages in the process working set extent. This value is used as an upper limit on the size of the sort data structure. Adjusting this value is one way to improve the efficiency of a sort operation.

Virtual memory is the number of pages of virtual memory added to the SORT image to hold the data.

The total of the **direct I/O** and **buffered I/O** is the number of I/O movements needed to read and write data. The lower these values are, the more efficient the ordering operation.

The number of **page faults** also indicates how well the data fits into memory: the higher the number of page faults, the less efficient the ordering operation.

Elapsed time is the total wall clock time used by the sort or merge operation, in hours, minutes, seconds, and hundredths of seconds.

The **input record length** value is obtained from Record Management Service (RMS) unless the user supplies it.

Internal length is the size in bytes of an internal format node. This includes any keys, data, a word to store the length, record file addresses (RFAs), and converted keys.

Output record length is the length of the output record. The length is computed from the input record length, the sort process, and the record reformatting requested.

Sort tree size is the number of records that fit in sort's internal sort data structure.

Number of initial runs is one indication of how well the data fits into memory.

The **maximum merge order** is the maximum number of sorted strings that are merged at one time.

The **number of merge passes** is the number of times SORT merges strings until one sorted output string is produced. The number of initial runs and the number of merge passes indicate how well the data fits in memory. The higher these numbers, the further the working set size is from containing the data, and the longer the sort takes.

Work file allocation is the number of blocks used for the work files. When more than one merge pass is needed, this size is approximately twice the size of the input file allocation.

Elapsed CPU is the CPU time used by the ordering operation; it does not include time spent waiting for I/O operations to complete or time spent waiting while another process executes.

EXAMPLE

```
$ SORT /STATISTICS PRICE1.DAT,PRICE2.DAT PRICE.LIS
```

This SORT command results in the following statistical display:

```
          VMS Sort/Merge Statistics

Records read:          793          Input record length:        80
Records sorted:        793          Internal length:            80
Records output:        793          Output record length:       80
Working set extent:    100          Sort tree size:            412
Virtual memory:        433          Number of initial runs:      2
Direct I/O:             22          Maximum merge order:         2
Buffered I/O:            9          Number of merge passes:      1
Page faults:          3418          Work file allocation:      114
Elapsed time: 00:00:05.98           Elapsed CPU:       00:00:03.63
```

In the sample statistics display, the sort data structure size is limited by the small working set extent. By doubling the working set extent you can almost double the sort data structure size, enabling all the records to fit in memory without using work files.

/WORK_FILES

Used for optimization.

FORMAT

/WORK_FILES=n

QUALIFIER VALUE

n
Specifies the number of work files requested; 0 to 10 files may be specified.

DESCRIPTION

SORT does not create work files until it needs them. If SORT needs work files, it creates two by default (SORTWORK0, SORTWORK1), which are placed in SYS$SCRATCH. Usually, there is no advantage to requesting more than one work file. However, if the available disks are too small or too full for SORT work files, you can increase the number of work files by any number from 1 through 10 to make each work file smaller.

EXAMPLE

```
$ ASSIGN DRA5: SORTWORK0
$ ASSIGN DB0: SORTWORK1
$ ASSIGN DB1: SORTWORK2
$ SORT/KEY=(POS:1,SI:80)/WORK_FILES=3 -
_$ STATS1,STATS2,STATS3,STATS4 SUMMARY.LIS
```

Since the input files in this sort operation are large files, specifying three work files improves the efficiency of the sort operation.

INPUT FILE QUALIFIERS

This section describes the qualifier to the input file. This qualifier should be specified immediately after the input file specification.

/FORMAT

Input File Qualifier

Defines input file characteristics; allows you to specify or override record or file size.

FORMAT

input filespec/**FORMAT**=*(type:n,[...])*

QUALIFIER VALUES

RECORD_SIZE:n

Specifies the file's longest record length (LRL) in bytes.

The maximum longest record length that can be specified depends on the file organization:

Sequential files	32,767
Relative files	16,383
Indexed-sequential files	16,362

These totals include control bytes for variable records with fixed-length control (VFC) format.

FILE_SIZE:n

Specifies input file size in blocks. The maximum file size accepted is 4,294,967,295 blocks.

DESCRIPTION

SORT obtains the LRL and file size from RMS. If you know the LRL that RMS has defined for the input files is incorrect, you can override this value by specifying the record size with RECORD_SIZE. For multiple input files, LRL is the length of the longest record in all files.

If you do not know the LRL value for a file, use the ANALYZE/RMS_FILE command. The LRL value appears in the file attributes section in the statistical report generated for the file that you specify.

SORT uses input file size information to determine the amount of memory needed, as well as the size of the work files for the sort operation. If the file size is unknown, (for example you are sorting files not residing on disk or standard ANSI magnetic tape) SORT assumes a fairly large file size.

If this default is too large, SORT overestimates its memory and work file requirements; the sort operation will be more efficient if you specify a smaller input file size. If the default is too small, SORT underestimates its memory requirements; therefore, you should specify a larger input file size, provided the sort data structure size is not limited by the working set extent.

EXAMPLE

```
$ SORT/KE=(POS:40,SIZ:2,DESC) -
_$ CRAO:YRENDAVG.DAT/FORMAT=(RECORD_SIZE:41,FILE_SIZE:3) DESCYRAVG.LIS
```

Because the input file, YRENDAVG.DAT, does not reside on a disk device or ANSI magnetic tape, file organization must be described by the /FORMAT qualifier.

OUTPUT FILE QUALIFIERS

This section describes the qualifiers to the output file. These qualifiers should be specified immediately after the output file specification.

/ALLOCATION

Output File Qualifier

Used for optimization.

FORMAT

output filespec/**ALLOCATION=n**

QUALIFIER VALUE

n
Specifies the number of blocks to be allocated. A value of 1 to 4,294,967,295 is allowed.

DESCRIPTION

SORT/MERGE preallocates space for the output file based on total input file allocation, thereby avoiding the overhead of extending the file every time another few blocks are written to it.

If you know, however, that the output file allocation will differ substantially from the total input file allocation (because you are reformatting data or omitting records) you can specify the number of blocks to be preallocated for the output file.

The /ALLOCATION qualifier is required if the /CONTIGUOUS qualifier is used.

EXAMPLE

```
$ SORT/KEY=(POS:1,SIZ:80) STATS.DAT -
_$ SUMMARY.LIS/ALLOCATION=1000/CONTIGUOUS
```

This SORT command allocates 1000 contiguous blocks for the output file, SUMMARY.LIS.

/BUCKET_SIZE

Output File Qualifier

Used for optimization.

FORMAT

output filespec/**BUCKET_SIZE=n**

QUALIFIER VALUE

n
Specifies the bucket size. A value of 1 to 32 is allowed.

DESCRIPTION

Use the /BUCKET_SIZE qualifier with relative and indexed-sequential output disk files to specify RMS bucket size (the number of 512-byte blocks per bucket). If the output file organization is the same as for the input files, the default value is the same as the first input file bucket size. If output file organization is different, the default value is 1. The maximum number of blocks per bucket is 32. See the *VMS Record Management Services Manual* for more information.

EXAMPLE

```
$ SORT/KEY=(POS:1,SI:80) STATS1.DAT,STATS2.DAT -
_$ SUMMARY.LIS/BUCKET_SIZE=16/RELATIVE
```

This SORT command results in the output file SUMMARY.LIS that has a bucket size of 16 with relative organization.

/CONTIGUOUS

Output File Qualifier

Used for optimization.

FORMAT

output filespec/**CONTIGUOUS**

DESCRIPTION

By default, SORT/MERGE does not allocate contiguous disk blocks for the output file. You can request, however, that the output file be stored in contiguous disk blocks by specifying the /CONTIGUOUS qualifier, thereby decreasing access time. If you use the /CONTIGUOUS qualifier, you must also specify the /ALLOCATION qualifier because, if the preallocated space is too small, RMS may be unable to extend the file contiguously.

EXAMPLE

```
$ SORT/KEY=(POS:1,SIZ:80) STATS.DAT -
_$ SUMMARY.LIS/ALLOCATION=1000/CONTIGUOUS
```

This SORT command allocates 1000 contiguous blocks for the output file, SUMMARY.LIS.

/FORMAT

Output File Qualifier

Specifies the output file record format if it differs from the input file format.

FORMAT

output filespec/**FORMAT**=*(type:n ...)*

QUALIFIER VALUES

FIXED:n

Specifies fixed-length records in the output file.

VARIABLE:n

Specifies variable-length records in the output file.

CONTROLLED:n

Specifies variable with fixed-length control (VFC) records in the output file.

n

Optionally indicates the maximum record size (in bytes) of the output records. The maximum record size allowed depends on the file organization.

Sequential files	32,767
Relative files	16,383
Indexed-sequential files	16,362

These totals include control bytes. If you do not specify the maximum record size, the default is a length large enough to hold the longest output record.

SIZE:n

Specifies the size, in bytes, of the fixed portion of VFC (CONTROLLED) records, up to a maximum of 255 bytes. If you do not specify SIZE, the default is the size of the fixed portion of the first input file. If you specify this size as 0, RMS defaults the value to 2 bytes.

BLOCK_SIZE:n

Specifies the output file's block size, in bytes, if you have directed the file to magnetic tape. You can also accept the default. If the input file is a tape file, the block size of the output file defaults to that of the input file. Otherwise, the output file block size defaults to the size used when the tape was mounted.

Acceptable values for block size (n) range from 20 to 65,532. To ensure correct data interchange with other DIGITAL systems, however, specify a block size of not more than 512 bytes. For compatibility with most non-DIGITAL systems, the block size should not exceed 2048 bytes.

DESCRIPTION

If the sort operation is a record or tag sort, the default output record format is the same as the first input file record format. If the sort operation is an address or index sort, the default output record format is fixed record format. If the input files have different record formats, SORT provides an output record size that is large enough to contain the largest record in the input files.

In specifying the output record format, you can indicate the maximum record size, in bytes, of the output records. You can specify fixed-length records, variable-length records, or variable with fixed-length control records.

EXAMPLE

```
$ SORT/KEY=(POS:1,SI:80) STATS.DAT SUMMARY.LIS/FORMAT=FIXED:80
```

The input file, STATS.DAT, consists of variable-length records that are 80 bytes in length. The /FORMAT qualifier specifies that the output file, SUMMARY.LIS, consists of fixed-length records.

/INDEXED_SEQUENTIAL

Output File Qualifier

Defines file organization.

FORMAT

output filespec/**INDEXED_SEQUENTIAL**

DESCRIPTION

If the organization of the output file is to be different from that of the input files, then you must specify the new organization. Use the /INDEXED_SEQUENTIAL qualifier to define indexed sequential organization for the output file. Additionally, the output file must exist and be empty, and you must use the /OVERLAY qualifier.

EXAMPLE

```
$ CREATE/FDL=NEW.FDL AVERAGE.DAT
$ SORT/KEY=(POS:1,SI:80) DATA.DAT,STATS.DAT -
_$ AVERAGE.DAT/INDEXED_SEQUENTIAL/OVERLAY
```

The CREATE/FDL command creates an empty file, AVERAGE.DAT. The SORT command specifies that the output file have an indexed-sequential organization and be written to the empty file, AVERAGE.DAT.

/OVERLAY

Output File Qualifier

Specifies that the output file is to be overlaid on, or written to, an existing empty file.

FORMAT

output filespec/**OVERLAY**

DESCRIPTION

If you specify indexed-sequential organization, the output file must already exist and must be empty. You must specify that the empty file is to be overlaid with the sorted records by using the /OVERLAY qualifier.

If the input file organization is sequential or relative and if you create an empty file for the sorted records using an RMS program, use the /OVERLAY qualifier to specify that the output file is to be overlaid.

You can use the Create/FDL Utility to create an empty data file; use the /OVERLAY qualifier to specify that SORT is to write output to that file. Any attributes that you specify when creating the empty file then become attributes of the SORT output file. See the *VMS Record Management Services Manual* for more information. You can also refer to this manual if you want to use the Convert Utility to produce an indexed-sequential file on output.

EXAMPLE

```
$ CREATE/FDL=NEW.FDL AVERAGE.DAT
$ SORT/KEY=(POS:1,SI:80) STATS.DAT AVERAGE.DAT/OVERLAY
```

The FDL file, NEW.FDL, specifies special attributes for the file, AVERAGE.DAT. When SORT writes output to that file, the resulting SORT output file has the attributes specified by the FDL file.

/RELATIVE

Output File Qualifier

Defines output file organization as relative.

FORMAT

output filespec/**RELATIVE**

DESCRIPTION

If the organization of the output file is to be different from that of the input files, then you must specify the new organization. If you do not specify file organization, the default for record and tag sorts is the organization of the first input file. You must use the /RELATIVE qualifier to specify relative output file organization.

EXAMPLE

```
$ SORT/KEY=(POS:1,SI:80) STATS.DAT SUMMARY.LIS/RELATIVE
```

Because the input file, STATS.DAT, is not a relative file and the output file, SUMMARY.LIS, will be, /RELATIVE qualifies the output file specification.

/SEQUENTIAL

Output File Qualifier

Defines output file organization as sequential.

FORMAT

output filespec/**SEQUENTIAL**

DESCRIPTION

If the organization of the output file is to be different from that of the input files, you must specify the new organization. If you do not specify file organization, the default for record and tag sorts is the organization of the first input file. If you do not specify file organization, the default for address and index sorts is sequential organization.

Use the /SEQUENTIAL qualifier when the default is not sequential file organization and you want an output file with sequential file format.

EXAMPLE

```
$ SORT/KEY=(POS:1,SI:80) STATS.DAT SUMMARY.LIS/SEQUENTIAL
```

Because the input file, STATS.DAT, is not a sequential file and the output file, SUMMARY.LIS, will be, /SEQUENTIAL qualifies the output file specification.

SPECIFICATION FILE QUALIFIERS

This section describes the qualifiers you can use in a specification file.

/CDD_PATH_NAME

Specification File Qualifier

Allows use of the Common Data Dictionary record definitions.

FORMAT

/CDD_PATH_NAME="cdd-path-name"

QUALIFIER VALUES

"cdd-path-name"
Specifies the CDD record definition.

DESCRIPTION

You can use the /CDD_PATH_NAME qualifier only if your system has VAX Common Data Dictionary (CDD) installed. The /CDD_PATH_NAME qualifier identifies CDD-defined fields and attributes for SORT. The **cdd-path-name** specifies a record definition within the CDD. Identifying these fields with this qualifier is the same as specifing them with the /FIELD qualifier. /CDD_PATH_NAME can be used in place of or in conjunction with /FIELD statements. Once the fields have been identified, they can then be used in later specification file clauses, such as /KEY, /CONDITION, /INCLUDE, or /OMIT.

EXAMPLE

```
/CDD_PATH_NAME="customer"
```

This /CDD_PATH_NAME qualifier identifies the customer record, which was previously identified in the CDD.

/CHECK_SEQUENCE

Specification File Qualifier

Specifies whether or not the sequence of records is checked.

FORMAT

/CHECK_SEQUENCE
/NOCHECK_SEQUENCE

DESCRIPTION

By default, MERGE checks the sequence of records in the input files. If you want to override that default, specify /NOCHECK_SEQUENCE in your specification file text.

EXAMPLE

```
/NOCHECK_SEQUENCE
```

This /NOCHECK_SEQUENCE clause overrides MERGE's default behavior.

/COLLATING_SEQUENCE

Specification File Qualifier

Specifies one of three predefined collating sequences or a user-defined sequence for character key fields. Allows you to modify any of the predefined collating sequences or any previously defined user-defined sequences.

FORMAT

/COLLATING_SEQUENCE=
(SEQUENCE=sequence_type
[,MODIFICATION=(character operator character)]
[,IGNORE=character or character-range,...]
[,FOLD]
[,[NO]TIE_BREAK])

QUALIFIER VALUES

SEQUENCE=sequence_type

ASCII
Specifies ASCII collating sequence, which is the default sequence.

EBCDIC
Arranges characters according to EBCDIC sequence. The characters remain in ASCII representation; only the order is changed.

MULTINATIONAL
Arranges characters according to MULTINATIONAL sequence, which collates the international character set. When you use the MULTINATIONAL sequence, characters are ordered according to the following rules:

- All diacritical forms of a character are given the collating value of the character (A′,A″,A\` collate as A).
- Lowercase characters are given the collating value of their uppercase equivalents (a collates as A, a″ collates as A″).
- If two strings compare as equal, tie-breaking is performed. The strings are compared to detect differences due to diacritical marks, ignored characters, or characters that collate as equal although they are actually different. If the strings still compare as equal, another comparison is done based on the numeric codes of the characters. In this final comparison, lowercase characters are ordered before uppercase.

Care should be taken when sorting or merging files for further processing using the MULTINATIONAL sequence. Sequence checking procedures in most programming languages compare numeric characters. Because MULTINATIONAL is based on actual graphic characters and not on the codes representing those characters, normal sequence checking does not work.

Note that some languages do not support MULTINATIONAL comparisons and instead can use the LIB$COMPARE_MULTI routine.

user-defined-sequence

Specifies a user-defined collating sequence. Define a collating sequence by specifying a string of single or double characters or ranges of single characters. (A double character is any set of two single characters collated as if they were one character. For example, "CH" can be defined to collate as "C".) This string should be enclosed in parentheses.

You can also represent characters by their corresponding octal, decimal, or hexadecimal values, using the radix operators: %O, %D, %X.

You must observe the following rules when defining your collating sequence:

- Enclose characters in quotation marks.
- Separate each character and character range by commas, and enclose the entire list in parentheses.
- Give all the characters appearing in the character keys in the sort or merge operation a collating value. Any character not given a collating value will be ignored unless the FOLD or MODIFICATION options are specified.
- Do not define a character more than once.
- Do not specify the null character by using quotation marks (""). Instead, use a radix operator, such as %X0.
- Specify quotation marks by enclosing them within another set of quotation marks (""""), or by using a radix operator.

MODIFICATION=(character operator character)

Specifies a change to the collating sequence specified in the SEQUENCE option. You can modify the ASCII, EBCDIC, MULTINATIONAL, or user-defined sequence. The sequence being modified must be specified in the SEQUENCE option of the clause, even if the sequence is the default (ASCII).

character

Specifies a character in the collating sequence. You can specify a single or double character. A double character is any set of two single characters collated as if they were a single character. Enclose the character in quotation marks.

operator

Specifies the operator used to compare the characters. You can specify greater than (>), less than (<), or equal to (=).

The kinds of changes permitted in the MODIFICATION option are listed below:

- A single or double character can be equated to a single character that has already been assigned a collating value ("a"="A").
- A single or double character can collate after a single character that has already been assigned a collating value ("CH"> "C").
- A single or double character can collate before a single character that has already been assigned a collating value ("D" <"A").
- A double character can be equated to a previously defined double character ("CH" = "SH").

- A single character can be equated to a double character sequence ("C" = "CH").

IGNORE
Specifies that SORT/MERGE ignore a character or character range in the collating sequence when making an initial comparison. Note that, when tie-breaking takes place, SORT/MERGE considers the characters specified in the IGNORE clause. Tie-breaking takes place when two or more strings have compared as equal and the MULTINATIONAL sequence is being used or when two or more strings have compared as equal and the TIE_BREAK clause has been specified.

FOLD
Specifies that all lowercase letters be given the collating value of their uppercase equivalents. For ASCII, EBCDIC, and user-defined sequences, the lowercase letters are a through z.

Since the lowercase letters in the MULTINATIONAL sequence already have the collating value of their uppercase equivalents, the use of FOLD is unnecessary.

TIE_BREAK
NOTIE_BREAK
Specifies whether or not SORT/MERGE should use numeric values to break any ties between characters that have equivalent values. By default, tie-breaking occurs with the MULTINATIONAL sequence. Specifying NOTIE_BREAK overrides this default and ensures that no further comparisons are made after the initial comparison.

A TIE_BREAK option must be specified for the ASCII, EBCDIC, and user-defined sequences in order for tie-breaking to occur. TIE_BREAK should be used when specifying FOLD or MODIFICATION for the these sequences.

DESCRIPTION

The /COLLATING_SEQUENCE clause specifies the collating instructions for a sort or merge operation. In a /COLLATING_SEQUENCE clause, you can specify ASCII (the default), EBCDIC, or MULTINATIONAL sequence; you can also define your own sequence.

In addition, the MODIFICATION, IGNORE, FOLD, and [NO]TIE_BREAK options of the /COLLATING_SEQUENCE clause can be used to modify any of these sequences. You can make more than one modification to the collating sequence. If you intend to modify any collating sequence, you must specify the sequence in the SEQUENCE option, even if it is the default sequence (ASCII).

Because the FOLD, MODIFICATION, and IGNORE clauses are processed in the order in which they are specified, care should be taken when specifying the order of those clauses. Normally, FOLD should be specified after all MODIFICATION and IGNORE clauses to ensure that the effects of the MODIFICATION and IGNORE clauses apply to uppercase and lowercase characters.

You can request that SORT/MERGE ignore a character or character range within the given collating sequence by using an IGNORE clause.

By default in the MULTINATIONAL collating sequence, SORT/MERGE folds lowercase letters onto their uppercase equivalents. If you want this folding to occur in the other collating sequences, you must specify a FOLD clause with the instructions for the collating sequence.

Also by default in the MULTINATIONAL collating sequence, SORT/MERGE uses numeric comparisons to break any ties in the collating values. Ties occur when two equal keys collate the same. If you do not want the default when using the MULTINATIONAL collating sequence, specify the keyword NOTIE_BREAK. For tie breaking in the other collating sequences, specify a TIE_BREAK clause.

EXAMPLES

1

```
/COLLATING_SEQUENCE=(SEQUENCE=ASCII,IGNORE=("-"," ")
```

This /COLLATING_SEQUENCE clause with an IGNORE option specified results in the following fields being compared as equal before tie breaking:

```
252-3412
252 3412
2523412
```

2

```
/COLLATING_SEQUENCE=(SEQUENCE=("A"-"L","LL","M"-"R","RR","S"-"Z"))
```

This /COLLATING_SEQUENCE clause defines a sequence in which the double character LL collates as a single character between L and M, and the double character RR collates as a single character between R and S. These double characters would otherwise appear in their normal alphabetical order. By default, this user-defined sequence does not define any other characters, such as lowercase a through z.

3

```
/COLLATING_SEQUENCE=(SEQUENCE=                   ! User-defined sequence
         ("AN","EB","AR","PR","AY","UN","UL",    ! that gives each month
          "UG","EP","CT","OV","EC","0"-"9"),     ! a unique value in its
                                                 ! chronological order
          MODIFICATION=("'"="19"),

          FOLD)
```

This /COLLATING_SEQUENCE clause defines a collating sequence that allows you to order a file SEMINAR.DAT according to the seminar date. The file SEMINAR.DAT is set up as follows:

```
16 NOV 1983   Communication Skills
05 APR 1984   Coping with Alcoholism
11 Jan '84    How to Be Assertive
12 OCT 1983   Improving Productivity
15 MAR 1984   Living with Your Teenager
08 FEB 1984   Single Parenting
07 Dec '83    Stress --- Causes and Cures
14 SEP 1983   Time Management
```

The primary key is the year field; the secondary key is the month field. Because the month field is not numeric and you want the months ordered chronologically, you must define your own collating sequence. You can do this by sorting on the second two letters of each month — in their chronological sequence — giving each month a unique key value.

The MODIFICATION option specifies that the apostrophe (') be equated to 19, thereby allowing a comparision of '83 and 1984. The FOLD option specifies that upper and lower case letters are treated as equal.

The output from this sort operation appears as follows:

```
14 SEP 1983   Time Management
12 OCT 1983   Improving Productivity
16 NOV 1983   Communication Skills
07 Dec '83    Stress --- Causes and Cures
11 Jan '84    How to Be Assertive
08 FEB 1984   Single Parenting
15 MAR 1984   Living with Your Teenager
05 APR 1984   Coping with Alcoholism
```

/CONDITION

Specification File Qualifier

Defines conditions for key and data handling and for record selection.

FORMAT

/CONDITION= *(NAME=condition-name, TEST=(field-name operator test-condition [logical-operator ...]))*

QUALIFIER VALUES

NAME=condition-name

Specifies the name of the condition you are testing. This **condition-name** can be used in /KEY, /DATA, /OMIT, and /INCLUDE clauses after it has been defined in the /CONDITION clause.

TEST=(field-name operator test-condition)

Specifies the conditional test.

field-name

Specifies the name of the field you are testing. The **field-name** must be defined previously in a /FIELD clause.

operator

Specifies the logical or relational operator used in the conditional test. The logical operators that you can use are AND and OR; the relational operators that you can specify are listed below.

Operator	Meaning
EQ	Equal to
NE	Not equal to
GT	Greater than
GE	Greater than or equal to
LT	Less than
LE	Less than or equal to

test-condition

Specifies the constant or **field-name** against which you are testing. A constant is specified with the following syntax:

```
Decimal_digits (default)
%Ddecimal_digits
%Ooctal_digits
%Xhexadecimal_digits
"character"
```

Note: **Normally, you don't need to specify the radix operator (%D); however, *test-condition* will assume the same data type as the *field-name*.**

The **field-name** must be defined in a /FIELD clause.

DESCRIPTION

A specification file can be used to change the relative order of a record or to alter the contents of certain fields of a record. You must first use a /CONDITION clause to define a conditional test. Once you define a test in a /CONDITIONAL clause, you can use that same test in a /KEY or /DATA clause to change the order of record or in a /OMIT or /INCLUDE clause to change the contents of a record.

If you want to change the order of records in the output file, you first specify a condition name in a /CONDITION clause and set up a test for what meets that condition. Then, you would specify the relative order in a /KEY clause of the form:

```
/KEY=(IF condition-name THEN value ELSE value)
```

You can use any values to specify the relative order of the records.

The /CONDITION clause also permits you to change the contents of a field in the output records. First specify a condition name, and then set up a test for what meets the condition. Specify the contents you want in the field in a /DATA clause of the form:

```
/DATA=(IF condition-name THEN "new-contents" ELSE "new-contents")
```

EXAMPLES

1
```
/FIELD=(NAME=AGENT,POSITION:20,SIZE:15)
/CONDITION=(NAME=AGENCY,
            TEST=(AGENT EQ "Real-T Trust"
            OR
            AGENT EQ "Realty Trust"))
   /DATA=(IF AGENCY THEN "Realty Trust" ELSE AGENT)
```

In this example, two real estate files are being sorted. One file refers to an agency as Real-T Trust; the other refers to the same agency as Realty Trust. The /CONDITION and /DATA clauses instruct SORT to list the AGENT field in the sorted output file as Realty Trust.

2
```
/FIELD=(NAME=ZIP,POSITION:60,SIZE:6)
/CONDITION=(NAME=LOCATION,
            TEST=(ZIP EQ "01863"))
/KEY=(IF LOCATION THEN 1
      ELSE 2)
```

In this example, all the records with a zip code of 01863 will appear at the beginning of the sorted output file. The conditional test is on the ZIP field, defined in the /FIELD clause; the condition is named LOCATION. The values of 1 and 2 in this /KEY clause signify a relative order for those records that satisfy the condition and those that do not.

3
```
/FIELD=(NAME=ZIP,POSITION:60,SIZE:6)
/CONDITION=(NAME=LOCATION,
           TEST=(ZIP EQ "01863"))
/DATA=(IF LOCATION THEN "NORTH CHELMSFORD"
       ELSE "Outside district")
```

In this example, the /CONDITION clause tests for the 01863 zip code. The /DATA clause specifies that the name of town field will be added to the output record, depending on the test results.

4
```
/FIELD=(NAME=FFLOAT,POS:1,SIZ:0,F_FLOATING)
/CONDITION=(NAME=CFFLOAT,TEST=(FFLOAT GE 100))
/OMIT=(CONDITION=CFFLOAT)
```

In this example, the number 100 is considered to be a F_floating data type because field FFLOAT is defined as F_FLOATING in the /FIELD clause.

/DATA

Specification File Qualifier

Specifies the fields of a record to be directed to the output file.

FORMAT

/DATA= *field-name*
/DATA= *(IF condition-name THEN "new-contents" ELSE "new-contents")*

QUALIFIER VALUES

field-name
Specifies the name of a field in a record. The **field-name** must be defined previously in a /FIELD clause.

condition-name
Specifies a **condition-name** that has been defined previously in a /CONDITION clause.

new-contents
Specifies how the record is to be altered. The **new-contents** can be a constant or a **field-name** that has been defined in a /FIELD clause.

DESCRIPTION

A /DATA clause must identify every field in the records you are directing to the output file. Specify the data fields in the order you want them to appear in the output record. By default, the record format for an output file is the same as that for the input file. If you want to eliminate or reorder fields from the output record, you can use a /DATA clause, causing only those fields identified in /DATA clauses to be directed to the output file.

You can conditionally change the contents of a field in the output records by first specifying a condition name and then setting up a test for what meets the condition in a /CONDITION clause. You then specify the contents you want in the field in a /DATA clause of the form:

```
/DATA=(IF condition-name THEN "new-contents" ELSE "new-contents")
```

EXAMPLES

1
```
/FIELD=(NAME=AGENT,POSITION:1,SIZE:5)
/FIELD=(NAME=ZIP,POSITION:6,SIZE:3)
/FIELD=(NAME=STYLE,POSITION:10,SIZE:5)
/FIELD=(NAME=CONDITION,POSITION:16,SIZE:9)
/FIELD=(NAME=PRICE,POSITION:26,SIZE:5)
/FIELD=(NAME=TAXES,POSITION:32,SIZE:5)
/DATA=PRICE
/DATA=" "
/DATA=TAXES
/DATA=" "
/DATA=STYLE
/DATA=" "
/DATA=ZIP
/DATA=" "
/DATA=AGENT
```

The /FIELD clauses define the fields in the records from an input file that has the following format:

```
AGENT ZIP STYLE CONDITION PRICE TAXES
```

The /DATA clauses, which use the **field-names** defined in the /FIELD clauses, reformat the records to create output records of the following format:

```
PRICE TAXES STYLE ZIP AGENT
```

2
```
/FIELD=(NAME=AGENT,POSITION:20,SIZE:15)
/CONDITION=(NAME=AGENCY,
            TEST=(AGENT EQ "Real-T Trust"
            OR
            AGENT EQ "Realty Trust"))
   /DATA=(IF AGENCY THEN "Realty Trust" ELSE AGENT)
```

In this example, two real estate files are being sorted. One file refers to an agency as Real-T Trust; the other refers to the same agency as Realty Trust. The /CONDITION and /DATA clauses instruct SORT to list the AGENT field in the sorted output file as Realty Trust.

/FIELD

Specification File Qualifier

Defines the fields in the input files.

FORMAT

**/FIELD=(NAME=field-name,POSITION:n,)
SIZE:n,[DIGITS:n,]data-type**

QUALIFIER VALUES

NAME=field-name

Specifies the name of the field. The **field-name** cannot have any embedded blanks, must begin with an alphabetic character, and can be no longer than 31 characters.

POSITION:n

Specifies the position of the field in the record.

SIZE:n

Specifies the size of a field containing character or binary data. In the specification file, SIZE implies byte lengths. The data type determines what values are acceptable, as follows:

- For character data, the size must not exceed 32,767 characters.
- For binary data, the size specified must be 1, 2, 4, 8, or 16 bytes.
- For floating-point data, no size is specified.

DIGITS:n

Specifies the size of a field containing decimal data. The size of a field containing decimal data must not exceed 31 digits. Note that DIGITS:n is used only when describing a field containing decimal data.

data-type

Specifies the data type of the field. You are not required to specify the **data-type** if it is character; SORT assumes character data type by default. The following data types are recognized by VMS SORT/MERGE:

```
CHARACTER
BINARY[,SIGNED]
BINARY,UNSIGNED
D_FLOATING
F_FLOATING
G_FLOATING
H_FLOATING
ZONED
DECIMAL[,SIGNED,TRAILING_SIGN,OVERPUNCHED_SIGN]
DECIMAL,LEADING_SIGN,SEPARATE_SIGN[,SIGNED]
DECIMAL,LEADING_SIGN,[OVERPUNCHED_SIGN,SIGNED]
DECIMAL,[TRAILING_SIGN],SEPARATE_SIGN[,SIGNED]
DECIMAL,UNSIGNED
PACKED_DECIMAL
```

DESCRIPTION

If you are altering the order or format of output records, you must identify each field of the records. These fields include key fields, fields to be compared, and fields to be directed to the output file. You identify each field by specifying a name, its position and size in the record, and its data type in the /FIELD clause.

Once the **field-name** has been specified in the /FIELD clause, it can be used in the /CONDITION, /KEY, and /DATA clauses.

EXAMPLE

```
/FIELD=(NAME=SALARY,POSITION:10,DIGITS:8,DECIMAL)
```

This /FIELD clause identifies a field in a record by the name SALARY, specifies that it starts in position 10 of the record, is 8 digits long, and consists of decimal data.

/INCLUDE

Specification File Qualifier

Specifies record selection, as well as multiple record formats.

FORMAT

/INCLUDE=(CONDITION=condition-name) [,KEY=...][,DATA=...]

QUALIFIER VALUES

CONDITION=condition-name

Refers to the **condition-name** specified in a previous /CONDITION clause.

KEY=...

Defines a key field because the default record type defined in the /KEY clause is not being used.

DATA=...

Defines a data field because the default record type defined in the /DATA clause is not being used.

DESCRIPTION

You can specify that records are to be conditionally included in an output file. After defining a condition in a /CONDITION clause, specify record selection in an /INCLUDE clause requesting that records satisfying the condition are to be included in the output file.

You can specify multiple /INCLUDE and /OMIT clauses in a specification file. The order you specify them determines the order the input records are tested for inclusion. After the last /INCLUDE clause, all records that have not already been included or explicitly omitted are omitted.

You can unconditionally include any records not previously omitted or included by specifying /INCLUDE without a condition.

When sorting multiple record formats, one /INCLUDE clause should be specified for each different record format among the records to be sorted. If you do not specify a KEY option within the INCLUDE clause, SORT assumes the default key definitions. If the KEY is specified in the /INCLUDE clause, the default key definitions are not used. The order of the KEY fields in the /INCLUDE clause determines how the internal key is built for sorting. The order of the DATA fields in the /INCLUDE clause determines the way the output record is formatted. If you specify a key or data field in an /INCLUDE clause, you must define all other key or data fields in the record.

See the Specification File Example Section for an example of a sort of records with two different formats.

EXAMPLE

```
/FIELD=(NAME=ZIP,POSITION:20,SIZE:6)
/CONDITION=(NAME=LOCATION,
           TEST=(ZIP EQ "01863"))
/INCLUDE=(CONDITION=LOCATION)
```

These /CONDITION and /INCLUDE clauses specify that records with the zip code 01863 will be included in the output file.

/KEY

Specification File Qualifier

Specifies the key fields.

FORMAT

/KEY=field-name
/KEY=(field-name,order)
/KEY=([IF condition-name THEN value ELSE] ...)
value [,order]

QUALIFIER VALUES

field-name
Specifies the name of the key field. The **field-name** has been previously specified in a /FIELD clause.

order
Specifies the order of the sort. The ASCENDING option specifies ascending order for a sort or merge operation. This option is the default. The DESCENDING option specifies descending order for a sort or merge operation.

value
Specifies the key. The value can be a constant or a **field-name** that has been defined in a /FIELD clause.

DESCRIPTION

If you are sorting on the entire record using character data, there is no need to specify your key field. Otherwise, you must specify a /KEY clause for each of the keys, in the order of their priority. You can sort on as many as 255 key fields.

There are three ways to use the /KEY clause. First, you can use the /KEY clause to identify the key field name. Second, you can use the /KEY clause to identify the key field name and to specify sorting order. In this case, you must enclose the field name and the order option in parenthesis. Finally, you can use a conditional /KEY clause to change the order of records in the output file. You first specify a condition name in a /CONDITION clause and then set up a test for what meets that condition. Then, you specify the relative order in a /KEY clause of the form:

```
/KEY=(IF condition-name THEN value ELSE value)
```

You can use any values to specify the relative order of the records.

EXAMPLES

1
```
/FIELD=(NAME=SALARY,POSITION:10,DIGITS:8,DECIMAL)
/KEY=(SALARY,DESCENDING)
```

This /KEY clause specifies that the key field is SALARY and that the sorting order is descending.

2
```
/FIELD=(NAME=ZIP,POSITION:20,SIZE:6)
/CONDITION=(NAME=LOCATION,
            TEST=(ZIP EQ "01863"))
/KEY=(IF LOCATION THEN 1
      ELSE 2)
```

In this example, all the records with a zip code of 01863 are to appear at the beginning of the sorted output file. The conditional test LOCATION (defined in a /CONDITION clause) is on the ZIP field (named in a /FIELD clause). The values of 1 and 2 in this /KEY clause signify a relative order for those records that satisfy the condition and those that do not.

/OMIT

Specification File Qualifier

Specifies record selection as well as multiple record formats.

FORMAT

/OMIT=(CONDITION=condition-name)

QUALIFIER VALUE

CONDITION=condition-name
Refers to the **condition-name** previously specified in a /CONDITION clause.

DESCRIPTION

You can specify that records are to be omitted from the output file by using an /OMIT clause. First, you must define a condition with a /CONDITION clause. Specify your record selection with an /OMIT clause requesting the records satisfying that condition be selected for omission from your sort. By default, SORT/MERGE includes all the other input records in the output file.

You can specify multiple /OMIT and /INCLUDE clauses in your specification file. The order you specify them determines the order the input records are tested for omission. All the records that have not already been included or omitted after the last /OMIT clause are included. You can unconditionally omit any records not previously omitted or included by specifying the /OMIT clause only.

EXAMPLE

```
/FIELD=(NAME=ZIP,POSITION:20,SIZE:6)
/CONDITION=(NAME=LOCATION,
           TEST=(ZIP EQ "01863"))
/OMIT=(CONDITION=LOCATION)
```

These /CONDITION and /OMIT clauses specify that records with the zip code 01863 are to be included in your output file.

/PAD

Specification File Qualifier

Allows you to specify a pad character to use when reformatting records or when comparing strings of unequal length.

FORMAT

/PAD=single-character

QUALIFIER VALUE

single-character

Specifies the character SORT will use to pad a string. Characters, decimal, octal, or hexadecimal digits can be used. The pad character should be specified as follows:

- Use quotation marks for a character—"#"
- Use decimal radix for decimal digits—%D35
- Use octal radix for octal digits—%O043
- Use hexadecimal radix for hexadecimal digits—%X23

DESCRIPTION

Use the /PAD clause to specifiy a pad character when comparing strings of unequal length or when reformatting records. By default, SORT uses the null character for padding, ensuring conformity with the previous versions. Double characters that can be defined as single characters ("ch" > "c") cannot be used as pad characters.

EXAMPLE

```
/PAD="."
```

This /PAD clause specifies that records will be padded with periods.

/PROCESS

Specification File Qualifier

Defines the processing method for the sorting operation (SORT only).

FORMAT

/PROCESS=type

QUALIFIER VALUES

RECORD
Specifies the record sort. This SORT process is the default.

TAG
Specifies the tag sort.

ADDRESS
Specifies the address sort.

INDEX
Specifies the index sort.

DESCRIPTION

By default, SORT uses a record sorting process. You can also specify a tag, address, or index sorting process. If you intend to reformat the output records, you cannot use address or index sort. For a comparison of the four processes, see the description of /PROCESS in the Command Qualifiers Section. Use the /PROCESS clause with SORT only.

EXAMPLE

```
/PROCESS=tag
```

This /PROCESS clause specifies that SORT use a tag sorting process.

/STABLE

Specification File Qualifier

Specifies that records with equal keys are directed to the output file in their input file order. The default condition is /NOSTABLE.

FORMAT

/STABLE
/NOSTABLE

DESCRIPTION

By default, the order of output records with equal keys is unpredictable. Specifying the /STABLE clause in a specification file arranges records with equal keys in the output file in the order of the input files as specified in the command line. If you use this qualifier when sorting multiple input files, on output, records with equal keys in the first file will precede those from the second file and so on.

EXAMPLE

```
/STABLE
```

This /STABLE clause ensures that records with equal keys will have the same order in the input and output files.

/WORK_FILES

Specification File Qualifier

Specifies the reassignment of work files.

FORMAT

/WORK_FILES=(device[,...])

QUALIFIER VALUE

device

Specifies a logical name for the work file. Unlike the DCL qualifier /WORK_FILES=n, the specification file clause /WORK_FILES=(device[,...]) specifies work file assignments, not the number of work files.

DESCRIPTION

You can improve the performance of SORT by placing work files on different disk-structured devices. Using the /WORK_FILES clause in a specification file to reassign work files makes it unnecessary to make logical assignments prior to invoking SORT at the command or program level.

For more information on reassigning work files, see the Description Section.

EXAMPLE

```
/WORK_FILES=("WRKD$:")
```

This /WORK_FILES clause assigns one of SORT's work files to the device WRKD$: because that device has the most space available.

SPECIFICATION FILE EXAMPLE

```
/FIELD=(NAME=RECORD_TYPE,PO:1,SIZ:1)    ! Record's type, one-byte field
/FIELD=(NAME=PRICE,PO:2,SIZ:8)          ! Price field, both files
/FIELD=(NAME=TAXES,PO:10,SIZ:5)         ! Taxes field, both files
/FIELD=(NAME=STYLE_A,PO:15,SIZ:10)      ! Style field, format A file
/FIELD=(NAME=STYLE_B,PO:20,SIZ:10)      ! Style field, format B file
/FIELD=(NAME=ZIP_A,PO:25,SIZ:5)         ! Zip code field, format A file
/FIELD=(NAME=ZIP_B,PO:15,SIZ:5)         ! Zip code field, format B file

/CONDITION=(NAME=FORMAT_A,              ! Condition test, format A file
        TEST=(RECORD_TYPE EQ "A"))
/CONDITION=(NAME=FORMAT_B,              ! Condition test, format B file
        TEST=(RECORD_TYPE EQ "B"))

/INCLUDE=(CONDITION=FORMAT_A,           ! Output format, type-A records
       KEY=ZIP_A,
       DATA=PRICE,
       DATA=TAXES,
       DATA=STYLE_A,
       DATA=ZIP_A)

/INCLUDE=(CONDITION=FORMAT_B,           ! Output format, type-B records
       KEY=ZIP_B,
       DATA=PRICE,
       DATA=TAXES,
       DATA=STYLE_B,
       DATA=ZIP_B)
```

In this example, two input files from two different branches of a real estate agency are sorted according to the instructions specified in a specification file. The records in the first file that begin with an "A" in the first position have this format:

```
|A|PRICE|TAXES|STYLE|ZIP|
 1 2     10    15    25
```

The records in the second file that begin with a "B" in the first position and have the style and zip code fields reversed, as follows:

```
|B|PRICE|TAXES|ZIP|STYLE|
 1 2     10    15  20
```

To sort these two files on the zip code field in the format of record A, you first define the fields in both records with the /FIELD clauses. Then, specify a test to distinguish between the two types of records with the /CONDITION clauses. Finally, the /INCLUDE clauses change the record format of type B to record format of type A on output.

Note that, if you specify either key or data fields in an /INCLUDE clause, you must explicitly specify all the key and data fields for the sort operation in the /INCLUDE clause.

Also note that records that are not type A or type B are omitted from the sort.

Index

A

Address sort
  reasons for selecting • SORT–29
  specifying • SORT–70
/ALLOCATION qualifier • SORT–40
Assigning work files • SORT–7, SORT–72

B

Block allocation
  specifying • SORT–40
Block size (magnetic tape)
  default • SORT–43
  restrictions • SORT–43
/BUCKET_SIZE qualifier • SORT–41

C

/CDD_PATH_NAME qualifier • SORT–50
/CHECK_SEQUENCE qualifier • SORT–12, SORT–51
Collating sequence • SORT–4
  ASCII • SORT–4, SORT–13, SORT–54
    table of • SORT–14
  default • SORT–4
  EBCDIC • SORT–4, SORT–13, SORT–54
  folding • SORT–55
  ignoring characters in • SORT–54
  modifying • SORT–53, SORT–54
  MULTINATIONAL • SORT–4, SORT–13, SORT–54
    table of • SORT–14
  restriction
    in user-defined • SORT–53
  specifying • SORT–54
/COLLATING_SEQUENCE qualifier • SORT–13, SORT–52
Conditional tests • SORT–58
/CONDITION qualifier • SORT–57
/CONTIGUOUS qualifier • SORT–42

D

/DATA qualifier • SORT–60
Data types
  specifying • SORT–62
/DUPLICATES qualifier • SORT–25

E

Equal key
  default order • SORT–71

F

/FIELD qualifier • SORT–62
File attributes
  specifying • SORT–43
Fixed-length records
  specifying • SORT–43
/FORMAT qualifier • SORT–37, SORT–43

I

/INCLUDE qualifier • SORT–64
/INDEXED_SEQUENTIAL qualifier • SORT–45
Index sort
  reasons for selecting • SORT–29
  specifying • SORT–70
Input file
  qualifier • SORT–37
  record size • SORT–37

K

Key
  equal • SORT–2
  multiple • SORT–2
  specifying • SORT–1, SORT–2, SORT–66

Key
  specifying (cont'd.)
    data type • SORT–26
    order • SORT–26
    position • SORT–26
    size • SORT–26
/KEY qualifier • SORT–26, SORT–66

L

Longest record length (LRL) • SORT–37

M

Maximum record size • SORT–43
Multiple keys
  number allowed • SORT–28
  rules for specifying • SORT–28

O

/OMIT qualifier • SORT–68
Operators in condition tests • SORT–57
Optimizing SORT
  system manager options • SORT–8
  user options • SORT–7
Output file
  organization • SORT–3
    defaults • SORT–45
    specifying • SORT–45
  qualifiers • SORT–3
    /ALLOCATION • SORT–40
    /BUCKET_SIZE • SORT–41
    /CONTIGUOUS • SORT–42
    /FORMAT • SORT–43
    /INDEXED_SEQUENTIAL • SORT–45
    /OVERLAY • SORT–46
    /RELATIVE • SORT–47
    /SEQUENTIAL • SORT–48
  reformatting record fields • SORT–61
Output file, preallocation
  for optimization • SORT–40
Output file organization
  defaults • SORT–47, SORT–48
  specifying • SORT–47, SORT–48
/OVERLAY qualifier • SORT–46

P

Pad character • SORT–69
/PAD qualifier • SORT–69
/PROCESS qualifier • SORT–29, SORT–70

R

Record format
  changing • SORT–44
  default for • SORT–44
  reformatting • SORT–61
Record selection
  including records • SORT–64
  omitting records • SORT–68
  order of clauses • SORT–64, SORT–68
Record sort
  reasons for selecting • SORT–29
  specifying • SORT–70
Reformatting the output record • SORT–61
/RELATIVE qualifier • SORT–47

S

Sequence checking
  default • SORT–12
  specifying • SORT–51
/SEQUENTIAL qualifier • SORT–48
SORT
  See Sort/Merge Utility
Sorting process • SORT–3
  address • SORT–29
  default • SORT–29, SORT–70
  index • SORT–29
  record • SORT–29
  selecting • SORT–29
  specifying • SORT–70
  tag • SORT–29
Sort/Merge Utility
  DCL qualifiers • SORT–11 to SORT–35
Sort order
  specifying • SORT–66
Specification file
  comments • SORT–5
  converting previous versions • SORT–6
  creating • SORT–5

Specification file (cont'd.)
  example of • SORT–73
  qualifiers • SORT–49
    /CDD_PATH_NAME • SORT–50
    /CHECK_SEQUENCE • SORT–51
    /COLLLATING_SEQUENCE • SORT–52
    /CONDITION • SORT–57
    /DATA • SORT–60
    /FIELD • SORT–62
    /INCLUDE • SORT–64
    /KEY • SORT–66
    /OMIT • SORT–68
    /PAD • SORT–69
    /PROCESS • SORT–70
    /STABLE • SORT–71
    /WORK_FILES • SORT–72
  special considerations • SORT–5
  uses for • SORT–5, SORT–31
/SPECIFICATION qualifier • SORT–31
/STABLE qualifier • SORT–32, SORT–71
Statistics
  sample display • SORT–33
/STATISTICS qualifier • SORT–33

T

Tag sort
  reasons for selecting • SORT–29
  specifying • SORT–70

U

User-defined sequence
  defining • SORT–53

V

Variable-length records
  specifying • SORT–43
Variable with fixed-length control records
  specifying • SORT–43

W

Work files
  assigning location of • SORT–72
  default number • SORT–35
  specifying number of • SORT–35
Working set extent • SORT–7
  adjusting, for optimization • SORT–7
/WORK_FILES qualifier • SORT–35, SORT–72

Reader's Comments

VMS Sort/Merge Utility Manual
AA–LA09A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ____________________

What I like best about this manual is ____________________

What I like least about this manual is ____________________

I found the following errors in this manual:

Page	Description
______	____________________
______	____________________
______	____________________
______	____________________
______	____________________

Additional comments or suggestions to improve this manual:

____

I am using **Version** ______ of the software this manual describes.

Name/Title ____________________ Dept. __________
Company ____________________ Date __________
Mailing Address ____________________
____________________ Phone __________

-- 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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here

Reader's Comments

VMS Sort/Merge Utility Manual
AA–LA09A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ______________________________

What I like best about this manual is ______________________________

What I like least about this manual is ______________________________

I found the following errors in this manual:

Page Description

Additional comments or suggestions to improve this manual:

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________ Dept. __________
Company ______________________ Date __________
Mailing Address ______________________
______________________ Phone __________

-- 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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here