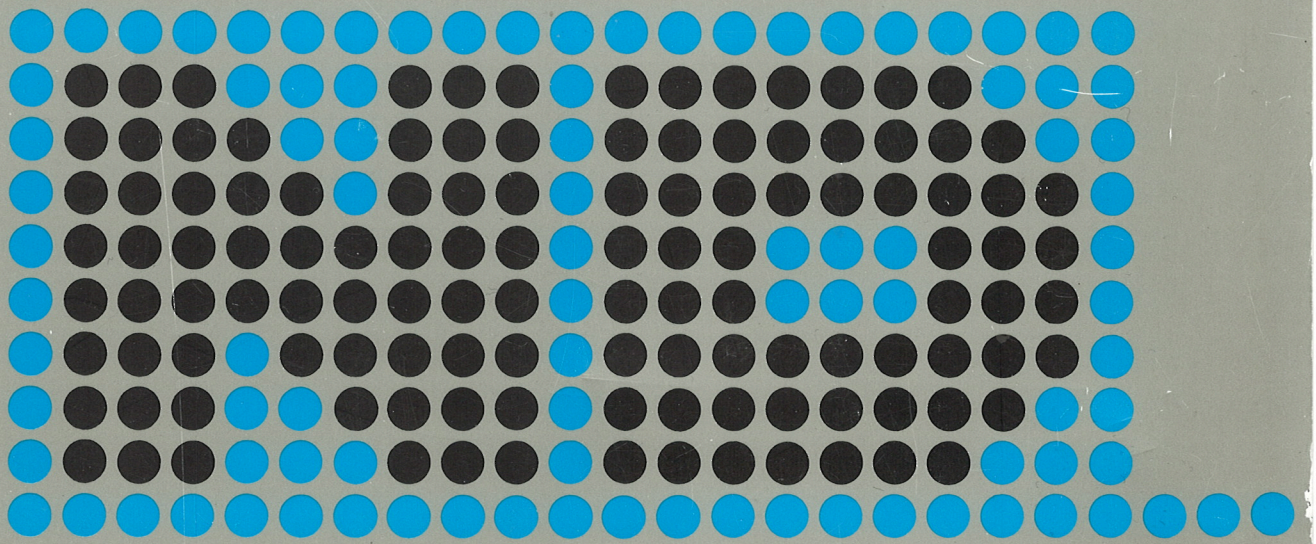
Norsk Data

ND-100/500 SORT-MERGE
User Guide

ND-60.236.1 EN



ND-100/500 SORT-MERGE
User Guide

ND-60.236.1 EN

PREFACE

The product

This manual describes the use of the SORT-MERGE package:

- ND-10179 version E, for ND-100,
- ND-10344 version C, for ND-500.

The package is a general purpose tool for rearranging data on mass storage files: disks, magnetic tapes, and floppy diskettes. The package can be used as a SINTRAN III subsystem, or called as a subroutine in application programs.

The reader

The readers of this manual fall into two categories.

The so-called "end user", that is any person using ND-computers, and needing to create data on files and organize them in different ways. The other category is programmers, or people with a knowledge of writing computer programs, with the need to include a sorting process as part of the program.

Prerequisite knowledge

Both categories of users are expected to have knowledge of running subsystems, and to create files using PED or NOTIS-WP text-editors.

To write a program using the SORT or MERGE subroutines, the reader should also be familiar with compiling and loading programs.

The manual

This is an new edition of the manual, with more emphasis on how to use the program, showing examples throughout.

Chapter 1 gives a short introduction to the SORT-MERGE concept, includes examples of three ways the program may be used: Interactive mode, that is typing commands directly from the terminal; running mode file or batch file containing all necessary commands; or calling the SORT or MERGE subroutines from a user-written application program.

Chapter 2 describes the commands that are mandatory in any sort or merge operation.

Chapter 3 describes the different types of files and data fields the program can handle.

Chapter 4 describes some additional commands: such as help and information, scratch file, use of magnetic tape as input and output, and two special commands for the ND-500 computers.

Chapter 5 describes the SORT and MERGE subroutines and their parameters as called from a user program, handling errors in the program, and handling errors running mode and batch jobs.

Chapter 6 lists all errors that may occur during an interactive session, together with a short explanation.

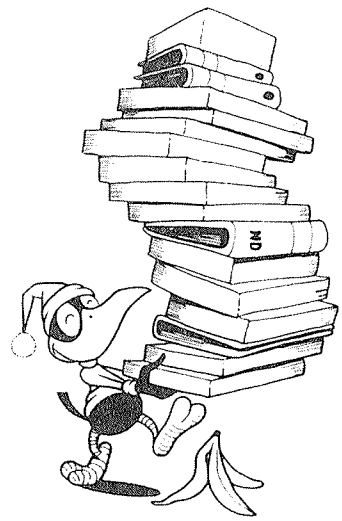
Chapter 7 describes the capacity of the program, the maximum size of the input file and the scratch file.

Chapter 8 explains the methods used for the sort and merge processes.

Related manuals

Most information on running the SORT-MERGE program is contained within the manual, however, programmers may need to refer to the following manuals:

SINTRAN III Reference Manual	ND-60.128
SINTRAN III Timeshar./Batch Guide	ND-60.132
ND FORTRAN Reference Manual	ND-60.145
ND Relocatable Loader	ND-60.066
BRF-LINKER User Guide	ND-60.196
ND-500 LINKAGE-LOADER/MONITOR.	ND-60.136



LIST OF FIGURES

LIST OF TABLES

LIST OF EXAMPLES

CHAPTER 1
HOW TO USE THE SORT-MERGE PROGRAM

- INTERACTIVE USE OF THE PROGRAM
- RUNNING THE PROGRAM FROM MODE FILE
- CALLING SORT-MERGE FROM A USER-WRITTEN PROGRAM

General tool

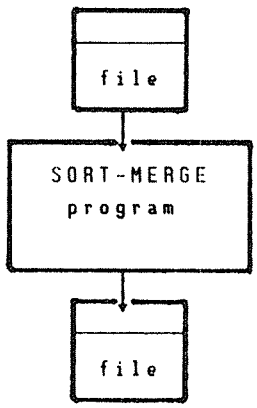
SORT-MERGE is a general tool to put records or lines of a file into ASCENDING (increasing) or DESCENDING (decreasing) order, using one or several fields as the ordering criterion.

Different file types

SORT-MERGE accepts files from many different sources, such as output from programs written in FORTRAN, COBOL, BASIC or other languages, and files made by the editors PED, NOTIS-WP.

What is SORT?

Sort means that all records or lines in one input file are ordered according to the contents of one or several fields, whether text or numeric data, into an ascending or descending sequence.



Input file

You must specify
- file organization,
- record size, number of sort fields
- position of fields in the record,
- size and type of each field

Output file

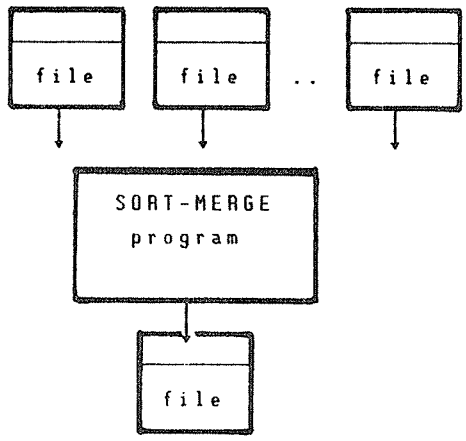
Figure 1. The SORT Operation

The contents of the records or lines are not changed, only the order they appear in the output file may be different.

Normally the input file is not changed, and the records are written to another file. However, it is possible to specify that the output appears on the same file as the input.

What is MERGE?

Merge takes two or more input files, already sorted, and combines records from them according to the contents of specified fields. The result is a single, sorted file.



Two or more input files, maximum 14

You must specify
- file organization,
- record size, number of sort fields,
- position of fields in the record,
- size and type of each field,
- name of input and output files

One single output file

Figure 2. The MERGE Operation

The merge operation is normally used when data is collected at a certain frequency, for example daily, weekly, or monthly, and there is need to combine all the data before further processing can continue.

Assuming that each data set is organized in the same way, and sorted on the same fields, it is faster to merge the presorted files, than to sort all data records once more.

ND-100 vs ND-500

This manual covers both the ND-100 and the ND-500 versions of the SORT-MERGE programs and the subroutine libraries.

The two programs behaves in the same way with a few exceptions.

Different data types

In the ND-100 version the data type INTEGER occupies 2 bytes (16-bits), whereas in the ND-500 it occupies 4 bytes (32-bits). Also, the data type REAL in the ND-100 occupies 6 bytes (48-bit floating point) and in the ND-500 it occupies 4 bytes (32-bit floating point).

This may influence the position of fields specified in the KEY-DESCRIPTION command, as well as the length of data records. However, using ASCII data types, both programs behaves identical.

ND-500 commands

The ND-500 version also includes two extra commands which may be used to improve the speed of the sorting process, these are explained on page 42.

INTERACTIVE USE OF THE PROGRAM

You can run the SORT-MERGE program directly from the terminal, typing commands and parameters in a dialog fashion. Commands and all necessary parameters may be typed on a single line, or if you type carriage-return (↵) after the command-name, the program asks for each parameter separately.

Starting the program

You start the program by typing:

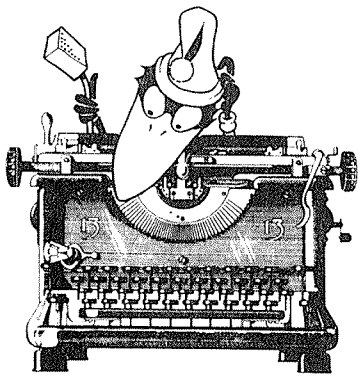
@SORT-MERGE-100↵

or

@SORT-MERGE-500↵

When the prompt character (*) appears you can type a command.

Commands and parameters



If the command you have given is accepted, the program displays a leading text for each parameter, and you are expected to respond with a value that are suited. The command is started when all the parameters have been entered.

If an error is detected, whether it be a unacceptable command or a wrong parameter value, an error message is displayed. You must repeat the command with correct information. The HELP command gives you an explanation of the error message, or a list of all commands and their parameters.

User input

In the examples throughout the manual input typed by the user is underlined, and the carriage-return is marked with ↵.

Note that this last symbol will, however, not appear on the terminal.

Correcting typing errors

Occasionally you may notice a typing error before you have pressed the carriage-return key. You may correct the line using all of the standard line-editing features of the SINTRAN III operating system, e.g. ctrl-A to delete a single character, ctrl-Q to delete a whole line, and so on. On certain terminals, a key marked DEL or ⌀ may also be used to delete single characters.

Do not use NOTIS-WP keys

Beware however, that using any 'NOTIS-WP' keys, such as WORD, SENT, LINE, ERASE, COPY, MOVE, etc, may cause the program to terminate abnormally. The SINTRAN III command

@CONTINUE, restarts the program, but the last command you entered must be repeated.

Example of interactive dialog:

```
@HELP,,"COMMANDS:TEXT"↵

@SORT--100↵   or @SORT--500
*rec-desc↵
record-length:    1:80↵
number of fields: 1↵
type of file:     TEXT↵
*key-desc↵
position:         9↵
length of field:  25↵
sequence:         ASCENDING↵
type of data:     ASCII↵
*sort↵
input-file: COMMANDS:TEXT↵
output-file: ↵

SAME AS INPUT ? (Y/N):y↵

 254 RECORDS SORTED
CPU TIME = 0.8 SECONDS
*exit↵
@
```

Comments:

Write all SINTRAN III commands
on a file called COMMANDS:TEXT.
Start the program.
Type the RECORD-DESCRIPTION command
- length of records in the file
- number of fields used as sort-key
- type of file.

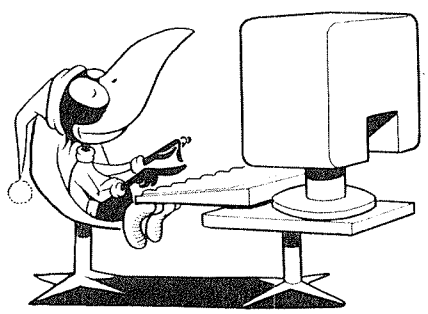
Type the KEY-DESCRIPTION command
- position where the field starts
- number of characters in the field
- ordering sequence
- type of data: alphanumeric

Type the SORT command
- name of input file
- carriage-return only

The program asks for confirmation

The sorted records can be found on
the input file.
Terminate the program

Example 1. Using SORT in Interactive Mode



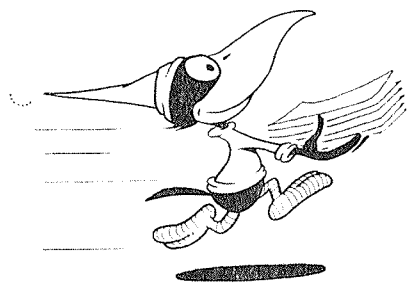
In this particular case, a file is made by using the operating system's HELP command, and then sorting all lines of that file in alphabetic order.

The file is in ASCII characters, all lines varying in size from 1 to 80 characters. The file type is therefore 'TEXT'.

The field we want to use for sorting begins in position 9, and is set to 25 characters length. (The lines are actually longer but this makes no difference to the result.) We want the output to appear in ASCENDING order, and the input file to receive the output of the sort.

Take a look at the file, both before and after the sorting, and see if you can match the various parameters.

RUNNING THE PROGRAM FROM MODE FILE



Instead of typing all the commands on the terminal each time the file is to be sorted, you may use an editor program to write the SORT-MERGE commands on a file.

This file can then be given as input file to the @MODE command which will then read the commands from the file instead of the terminal. The output from the program is written to the terminal, or directed to another output file.

In such a file you must write each command and its parameters on a single line. The dialog mode, or input directly from the terminal is not allowed.

The example below shows such a command file, using the same parameters as for the interactive job in the previous section, except that the output from the SORT command is directed to another file.

Example of MODE file

```
@help,,"commands:text"
@sort-merge-100
rec-desc 80, 1, text
key-desc 9,30,ascending,ascii
sort commands:text,sorted:text
exit
```

The command and its parameters are typed on the same line.

These lines are stored on a file called SORT-JOB:MODE.

Example 2. Using SORT in Mode Job

Assuming that the commands above are stored in the file SORT-JOB:MODE, starting execution is done by:

```
@MODE SORT-JOB:MODE,TERMINAL
```

The "dialog" with the SORT-MERGE program appears on the TERMINAL, or it can be directed to a file. The output from the program is stored on the file SORTED:TEXT.

Example 3. Command to start a Mode Job

@MODE saves typing

The use of command files and the @MODE command may save you a lot of typing. Further information on the use of command files can be found in the manual SINTRAN III Timesharing and Batch Guide - ND-60.132.

CALLING SORT-MERGE FROM A USER-WRITTEN PROGRAM

If the SORT program is used quite frequently, it may be convenient for the user to have a program that for example asks the name of the input file, and then calls the SORT or MERGE routine, executing the operation from within the program.

Ease of use

This way the user may have less to worry about, as all other information to SORT or MERGE is set up by the program.

```
PROGRAM MYSORT
:
:
    CALL SORT(infile,outfile,..
:
END
```

The program calls the subroutine SORT or MERGE with all the parameters necessary for the job.

The source program must be compiled and loaded together with the SORT library, to make an executable program.

```
@MYSORT↵
```

To start the program, simply type the program name.

Example 4. Calling SORT from a User-Written Program

Details on how to write a program to perform a sorting operation is shown on page 45.

CHAPTER 2
THE MOST IMPORTANT COMMANDS

- RECORD-DESCRIPTION
- KEY-DESCRIPTION
- SORT
- MERGE
- EXIT

In this chapter the commands that are needed in all SORT or MERGE operations are explained. Other commands also exist but they are used less often. Their definitions are given in chapter 4.

The two first commands define the type of file and the type of data the program is to work with, and must be given first and in the order shown, followed by either a SORT or a MERGE command specifying the files to be used for input or output.

- RECORD-DESCRIPTION
- KEY-DESCRIPTION
- SORT
- MERGE

  Several SORT or MERGE commands may be given, using the same record and key formats from the current descriptions.

  Finally, the command

- EXIT

terminates the SORT-MERGE program, returning control to the operating system.

RECORD-DESCRIPTION

With this command you must define three parameters: the length of the records, the number of fields the SORT or MERGE is to use in the comparisons, and whether the type of records in the file are of fixed or variable length.

Format:

```
RECORD-DESCRIPTION <record-length> <no-of-fields> <type-of-file>
```

<record length>

This parameter can be given either as a fixed number, e.g. 80, or as two numbers separated by a colon (:), indicating variable length records, e.g. 50:80. In this case the first number is the the minimum number of characters in a record, and the second number is the maximum length.

Note that if the <type-of-file> is FIXED, only the first number is used.

<no-of-fields>

This parameter specifies the number of fields to be used for comparison by the SORT or MERGE operation.

This number controls how many parameters the user may give in the following KEY-DESCRIPTION command.

A maximum of 10 fields can be used if the SORT-MERGE program is run in interactive mode or from a mode file. However, user-written programs calling the SORT-MERGE routines can handle up to 99 fields. See page 45.

Note that only the number of fields to be used in the SORT or MERGE operation should be specified here, not all the different fields in a record.

<type-of-file>

This parameter defines the type of file, and may take one of the following values:

- TEXT The file is written in 7-bit ASCII characters; each record is terminated by a carriage-return and a line-feed character (CR+LF).

This type of file is made by most programming languages and such text-editors as PED and NOTIS-WP, and must contain only printable characters.

For COBOL programs this type of file is written using 'RECORDING MODE T'.

- FIXED All records of the file are of the same length. The file can be in 7-bit ASCII as for 'TEXT' files, but other recording formats may produce fixed formatted records containing binary and/or BCD data.

  For COBOL programs this type of file is written using 'RECORDING MODE F', then there is no carriage-return/line-feed terminating the records.

  Using FIXED length records enables the SORT-MERGE program to work about 7 times faster than for TEXT and VARYING type. Remember to include the carriage-return and line-feed characters in the record length, if such are used to terminate the records.

- VARYING Each record has a "byte-count" indicating the length of the data fields that follow. This counter is automatically added for COBOL programs using 'RECORDING MODE IS V'. This two-byte counter should not be included in the record-length parameter.

ND-100 vs ND-500

Please note the different size of the data types INTEGER and REAL in the ND-100 vs ND-500, since this may influence the positions of fields and the length of the data records. More detailes can be found on page 29ff.

```
@sort-merge-100
RECORD-DESCRIPTION 50:80, 2, TEXT
key-description 10 30 ascending ascii,  1 10 descending ascii
sort indata:data, sorted:data
exit
```

Figure 3. The RECORD-DESCRIPTION Command

KEY-DESCRIPTION

Here you must specify four parameters: position, length, type of data, and sequence of order, for each fields to be used in the SORT or MERGE operation.

All fields to be used for comparison by SORT or MERGE must be defined by a KEY-DESCRIPTION command. The parameters may be separated with blanks or commas.

If you run the SORT-MERGE in interactive mode, the program asks you to enter four values for each of the fields specified in the RECORD-DESCRIPTION command.

You may respond with the values for a single field, one item at a time until all four values are entered, or with all four values for one field, or with all values for all fields at once. The prompt text for each parameter is repeated until all fields are satisfactorily entered.

When running the program as a mode job, however, all field parameters must be given on the same line as the command.

Major and Minor keys

The order the fields are given in determines the major or minor order of fields. In other words, if several records have the same value in a field, the contents of the minor fields determines the final order on the output file.

The first field that is defined, whichever location it has in the record, becomes the major field, then the following fields becomes the minor fields in the order which they are given.

Format:

```
KEY-DESCRIPTION  <position> <length> <sequence> <type> ....
```

<position>

The position in the record where the field begins. The first user data start in position 1, even for files with type VARYING.

<length>

The size of the field in characters (bytes).

If the field is located in the variable part of a TEXT-record, the SORT or MERGE fills up the remaining part up to the maximum length with blanks or zeroes, depending on data type.

For fields of the data type BCD, see below, each character or byte represents two digits.

Note that this parameter cannot exceed the maximum length of the record, and cannot be larger than 255 bytes. The total length of all fields cannot exceed 255 bytes either.

<sequence>

The sequence in which the field is to be ordered:

- ASCENDING means that the field is ordered on increasing values: .e.g. 1,2,3,...9, or A,B,C,...Z
- DESCENDING means that the field is ordered on decreasing values: .e.g. 9,8,7....1, or Z,X,Y,...A

The parameter can be abbreviated to 'A' or 'D' respectively.

<type of data>

The following data types are accepted:

- ASCII, the field is treated according to ASCII alphabet.
- ASCII-UPPER-CASE, all the characters are treated as uppercase letters.
- ALTERNATIVE-ASCII, the field is sorted using alternate code as defined using ALTERNATIVE-COLLATING-SEQUENCE command, see page 39.
- BCD, the field is treated as a set of 4-bit digits, 2 per byte.
- BITSTRING, the field is treated as a series of 8 bit unsigned integers. The leftmost bit contains the sign.
- INTEGER, the field is treated as a binary word, length either 2 or 4 bytes, representing a 16-bit (ND-100) or 32-bit word (nD-500). The leftmost bit contains the sign.

- NUMERIC-UNSIGNED, the field must only contain numeric ASCII characters. No sign or special symbols are allowed, and may disrupt the sort-merge process.
- NUMERIC-LEADING-SEPARATE, the first character must only contain a minus (-) or a plus (+) sign.
- NUMERIC-TRAILING-SEPARATE, the last character must only contain the + or - sign respectively.
- NUMERIC-LEADING-EMBEDDED, the first character may contain the first digit and the sign, using 'multipunch' technique.
- NUMERIC-TRAILING-EMBEDDED, the last character may contain the first digit and the sign, using 'multipunch' technique.
- REAL, the field consists of a floating-point number. The length of the field is either 6 bytes (48-bit ND-100) or 4 bytes (32-bit ND-500), depending on the computer used to create the file.

All parameter names, such as ASCII, NUMERIC-UNSIGNED, INTEGER, etc, may be abbreviated, as long as the names do not become ambiguous.

On page 29ff. the different data types are described in more detailes.

ND-100 vs ND-500

Please note the different size of the data types INTEGER and REAL in the ND-100 vs ND-500, since this may influence the positions of fields and the length of the data records. More detailes can be found on page 29ff.

```
@sort-merge-100
record-description 50:80, 2, text
KEY-DESCRIPTION 11 30 ASCENDING ASCII,  1 10 DESCENDING ASCII
sort indata:data, sorted:data
exit
```

Figure 4. The KEY-DESCRIPTION Command

SORT

This command starts the sorting operation, if the record-description and key-description commands have been given.

Files can be stored on disk, floppy diskettes, and magnetic tapes. Peripheral devices such as line-printers and terminals cannot be used as input or output directly from the SORT-MERGE program.

The input file may be used to receive the result of the operation, but care should be taken, because if the sort fails due to inconsistency with record-lengths, the original input file may be destroyed.

Format:

```
SORT <input-file> <output-file>
```

<input-file>

The name may be any standard SINTRAN III file name, including: remote computer, directory name, user name, file name, and file type. Default file type is :DATA.

<output-file>

The output file follows the same format as the input file,

If the input file is to receive the sorted output, a CR (carriage-return) should be given.

This option must be confirmed with the following request:

```
SORT input-file ↵
SAME AS INPUT (Y/N): y ↵
```

Figure 5. Confirming that the Input File is to be Used as Output File

Any other response than 'Y' or 'y', terminates the command.

The output file can be either INDEXED or CONTIGUOUS, the latter makes the SORT run faster. A contiguous file is made by giving a fixed number of pages when the file is created. An indexed file is made by giving

zero (0) pages when the file is created.

Successful operation

When the sort operation is finished successfully, the following performance report is printed:

```
xxxx RECORDS SORTED
CPU-TIME yy.zz SECONDS
```

Figure 6. Sort Performance Report

Errors detected

If the SORT detects an error, either that a file cannot be read or written, or the record-length specified does not correspond with the number of records in the file, an error message is printed.

If you give a HELP command after such an error message, a short explanation of the error is displayed.

All such messages are listed on page 61, together with how the situation is correct.

```
@sort-merge-100
record-description 50:80, 2, text
key-description 10 30 ascending, ascii  1 10 descending ascii
SORT INDATA:DATA, SORTED:DATA
exit
```

Figure 7. The SORT Command

MERGE

This command allows a set of files, each having the same record-layout and organization, and sorted in the same sequence, to be collected in one output file. This contains all records of the input files ordered according to the fields specified with the KEY-DESCRIPTION command.

Combining sorted files

A typical use of the MERGE command is when data is collected file at certain time intervals, e.g. weekly or monthly, and the user wants to combine these files into one single file. A number of files can be merged into one file in one operation.

Handling large files

You may find the merge operation particularly useful if your data files are extremely large, or if you have limited disk-space for the work file needed by the SORT-MERGE program. (See the SCRATCH command on page 38.)

You may divide the large file into several smaller ones, sort each one separately, and then combine all the files into one with the MERGE command.

The size of the largest file that the SORT procedure can handle, and the space required by the work file, can be found on page 71.

Format:

```
MERGE <number-of-input-files> <names-of-input-files> <output-file>
```

<number-of-input-files>

You must specify the number of files the program is to use for input. The maximum number is 14.

<names-of-input-files>

A set of file names, separated by blanks or commas. The number of names must correspond with the number given in the previous parameter. Default file type is :DATA.

In the interactive mode, the program continues to ask for file names until the required number have been given.

<output-file>

The name of the output file. This file must be large enough to receive all records of the input files together. Default file type is :DATA.

An output file must be specified; it is not possible to use one of the input files as for the SORT command.

The output file can be either INDEXED or CONTIGUOUS, the latter reduces the time of the operation.

Successful termination

When the MERGE operation has been successfully completed, the following performance report is printed:

```
Final pass running
xxxx Records merged
CPU-time= yy.zz Seconds
```

Figure 8. Merge Performance Report

Errors detected

If the MERGE detects an error, either that a file cannot be read or written, or the record-length specified does not correspond to the number of records in a file, the operation is terminated and an error message is printed.

If you type the HELP command after such an error message, a short explanation is displayed on the terminal.

All such error messages are listed on page 61.

```
@sort-merge-100
record-description 50:80, 2, text
key-description 11 30 ascending ascii, 1 10 descending ascii
MERGE 3 ORDERS-JAN,ORDERS-FEB, ORDERS-MAR, Q1-ORDERS
exit
```

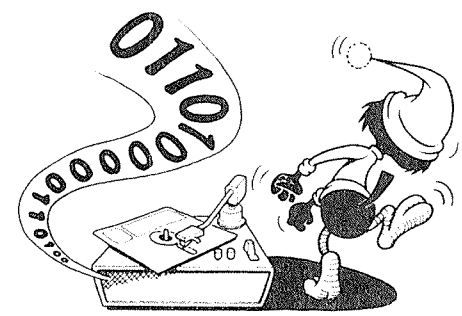
Figure 9. The MERGE Command

EXIT

This command terminates the SORT-MERGE program, and returns control to the operating system.

Format:

```
EXIT  (no parameters)
```



CHAPTER 3
TYPES OF FILES AND FIELDS

- FILE TYPES:
  - FIXED
  - TEXT
  - VARYING

- FIELD TYPES:
  - ASCII
  - ASCII-UPPER-CASE
  - ALTERNATIVE-ASCII
  - NUMERIC-UNSIGNED
  - INTEGER
  - REAL
  - BCD
  - NUMERIC-LEADING-SEPARATE
  - NUMERIC-LEADING-EMBEDDED
  - NUMERIC-TRAILING-SEPARATE
  - NUMERIC-TRAILING-EMBEDDED

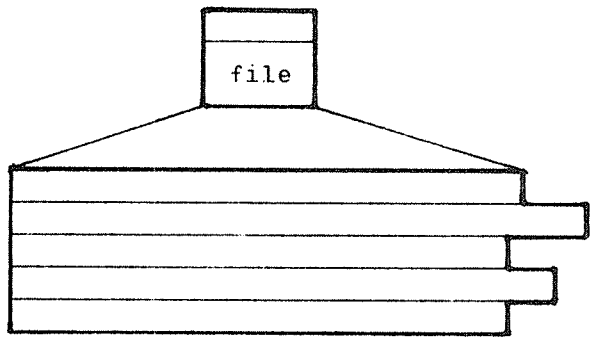
DIFFERENT TYPES OF FILES

What is a FILE?

A file is a collection of records, or lines, containing some relevant information. Examples are names, addresses, and phone numbers of friends and relatives, or transactions to an inventory register, just to mention a few.

For the SORT or MERGE program it is not important what the file will be used for, only which fields in the records you want to use to determine how the records are ordered.

Neither SORT nor MERGE change the contents or the position of the fields within a record.



A file normally contains many different records, each bearing similar information.

The records in the file may be of either fixed size or of variable size.

Figure 10. Illustration of Records in a File

A file may contain records of the same size, or records of different lengths. The SORT-MERGE program must be told how the file is organized; this is done by the RECORD-DESCRIPTION command.

FIXED record size

For this type of file each record in the file is of the same length. The record may consist of different types of fields, a mixture of ASCII characters, BCD, and/or binary fields.

Such files are mostly made by application programs using formatted output statements.

In COBOL programs the file must be declared with 'RECORDING MODE F'.

TEXT records

This type of file is often created by the user with text-editors such as PED or NOTIS-WP, and consists of 7-bit ASCII characters.

In COBOL programs the file must be declared with 'RECORDING MODE T'.

The records may be of the same size (FIXED), or vary in size (TEXT), however, each record must be terminated by a CARRIAGE-RETURN and a LINE-FEED character.

When looking at the file with the PED or NOTIS-WP editors, note that the two terminating characters are not seen. They must however be counted when determining the length of the records.

The SORT-MERGE runs considerably faster if the file is made up of records of fixed length and even number of bytes, than with variable length and odd number of bytes.

VARYING length records

This type of file is written by COBOL programs, using file description with the clause 'RECORDING MODE V'.

Each record, which may vary in length, includes a two-byte count field, indicating the size of the total record. The remaining fields may be a mixture of different data types, such as characters, binary, or BCD numbers.

When specifying the sort-field position, the two-byte count field should not be counted, i.e. the first user field begins at character position 1.

DIFFERENT TYPES OF FIELDS

What is a FIELD ?

Normally the records in a file contain data that is broken into several fields. In our name and address file below, each element, the name, the address, and the phone-number, constitutes a separate piece of information occurring in all records in the file.

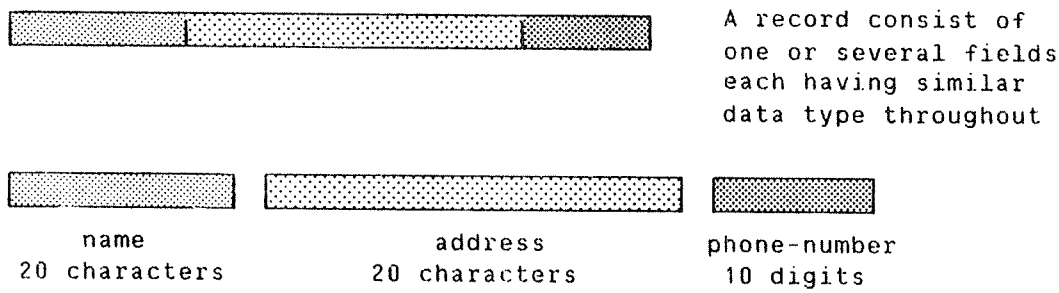


Figure 11. Illustration of Fields in a Record

Each of the fields may contain data with alphabetic or numeric values; they can vary in size, and in the way a data item is organized.

For the purpose of the SORT or MERGE operation, the fields which will be used to determine the sequence of the output file must be specified using the KEY-DESCRIPTION command.

The information about a field consists of four parts: the character position, the length of the field, the order of sequence, and the type of data in the field.

Adjacent fields

In some cases, where the data is of the same type, and the fields are adjacent to each other, as for example the name and the address field in the record above, the user can combine the two fields into one sort-key. This will reduce the time for the SORT-MERGE operation.

The different types of data that can be represented in a field are described below:

Alphanumeric data

Alphabetic information consists mostly of letters from the alphabet, in upper and lowercase types. Often also digits and special symbols appear in alphabetic fields. This is then referred to as alphanumeric data.

Such characters are usually taken from the ASCII character set, each letter using 8-bits representing the position of the letter within the alphabet.

In certain cases the normal character sequence does not suit the order in which the user wants the output to appear. In such a case an "alternate collating sequence" can be specified, where the user can define the order of character sequence in the alphabet.

ASCII The field is sorted or merged based on the ASCII character set.

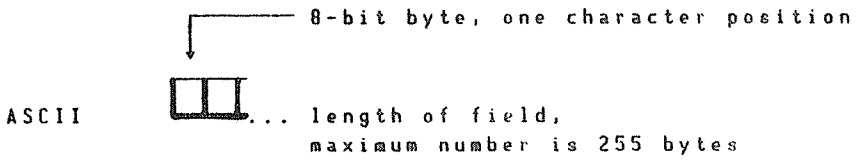


Figure 12. ASCII Data Type

ASCII-UPPER-CASE All letters is treated as if they were in uppercase.

Otherwise the type and organization is like the ASCII characters.

ALTERNATIVE-ASCII The field is sorted according to the alternative collating sequence; the user defines the order of the character set to be used.

The type and organization is like the ASCII characters.

See description of the command ALTERNATIVE-COLLATING-SEQUENCE on page 39.

Numeric data Numeric data can be a little bit more complex. The fields can be made up of BCD characters, using 4-bits per digit, as fixed size byte-strings or bit-strings, many times referring to the computer's "word-length".

The value can be represented in a binary fashion, which is efficient for the computer to handle, but difficult for people. Often, such binary data must be converted to alphanumeric characters before it can be read by people.

The SORT-MERGE program can handle many different numeric data types.

BCD

The field consists of binary coded digits (BCD), as a string of 4-bit numeric data, packed two and two per characters (bytes). This type of data is usually produced by COBOL programs, using the data description PICTURE COMP-3.

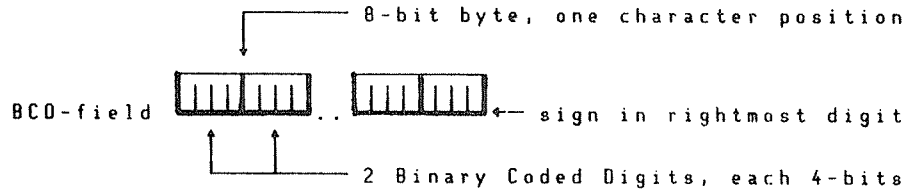


Figure 13. BDC Data Type

The length of this field must be calculated as:

$$\frac{\text{number of digits} + 1}{2} = \text{number of bytes}$$

Maximum length is 18 digits, or 10 bytes including the sign position.

In COBOL, the description PICTURE 9(11) COMP-3, occupies 6 bytes.

BITSTRING

The field is regarded as consisting of 8-bit bytes, and treated as unsigned integer.

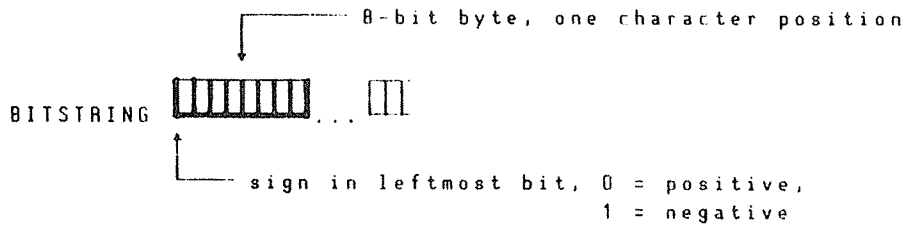


Figure 14. BITSTRING Data Type

INTEGER

The field is regarded as 16-bit (ND-100) or 32-bit (ND-500) binary words, or 2 or 4 bytes, respectively. The leftmost bit is treated as sign-bit, indicating negative value if set to 1.

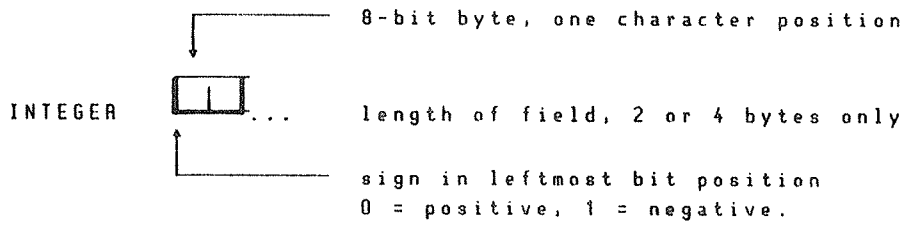


Figure 15. INTEGER Data Type

ND-100 vs ND-500

Please note the different sizes of this data type in the ND-100 and the ND-500, since this may influence the position of the fields and the length of data records.

NUMERIC-UNSIGNED

The field must consist of numeric bytes, characters 0..9, only. No sign symbols are allowed (+ or -), and any other characters in the field may produce unpredictable result.

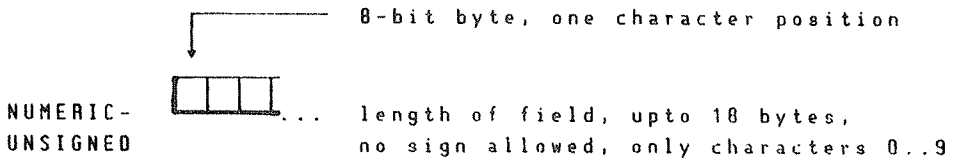


Figure 16. NUMERIC-UNSIGNED Data Type

In COBOL, the description PICTURE 9(5) NUMERIC-UNSIGNED, occupies 5 bytes.

NUMERIC-LEADING-SEPARATE

The field must consist of numeric bytes only, but the leftmost byte may contain a minus sign (-), indicating a negative number.

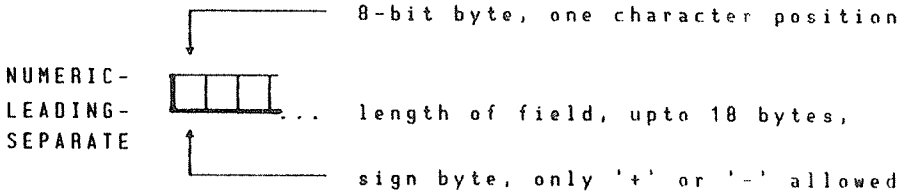


Figure 17. NUMERIC-LEADING-SEPARATE Data Type

In COBOL, the description PICTURE S9(5) NUMERIC-LEADING-SEPARATE, occupies 6 bytes.

NUMERIC-TRAILING-SEPARATE

The field must consist of numeric bytes only, the rightmost byte is reserved for sign representation.

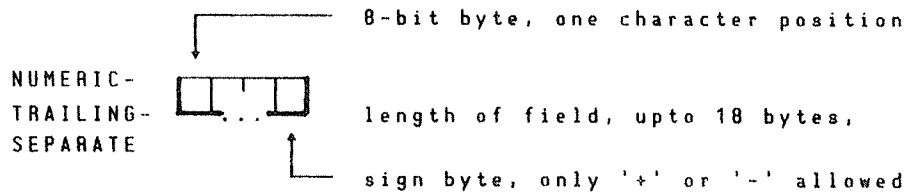


Figure 18. NUMERIC-TRAILING-SEPARATE Data Type

In COBOL, the description PICTURE S9(5) NUMERIC-TRAILING-SEPARATE, occupies 6 bytes.

NUMERIC-LEADING-EMBEDDED (*)

The field must consist of numeric bytes only. The leftmost byte may contain the sign representation using a 'multipunch' feature, an old term from the punch-card era, where a negative digit was indicated with an extra punch in column 12, creating an alphabetic character to represent a negative number.

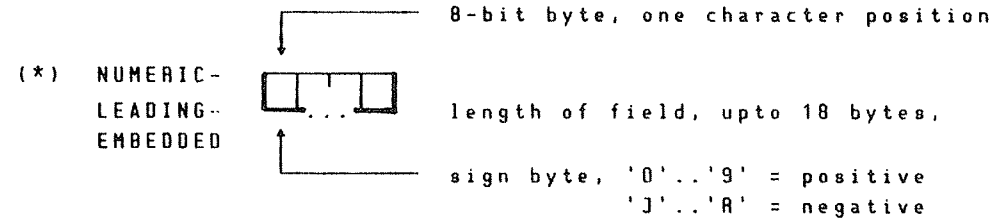


Figure 19. NUMERIC-LEADING-EMBEDDED Data Type

In COBOL, the description PICTURE S9(5) NUMERIC-LEADING-EMBEDDED, occupies 5 bytes.

(*) Note that the TRAILING-EMBEDDED representation is the default sign representation of the ANS COBOL standard. The embedded sign representation of the former ND-COBOL systems, (older than 1980) is not completely compatible with this standard, and may, in certain cases cause unpredictable results.

NUMERIC-TRAILING-EMBEDDED (*) This is similar to the previous type, but the rightmost byte may contain the sign, using the 'multipunch' feature to indicate negative number,

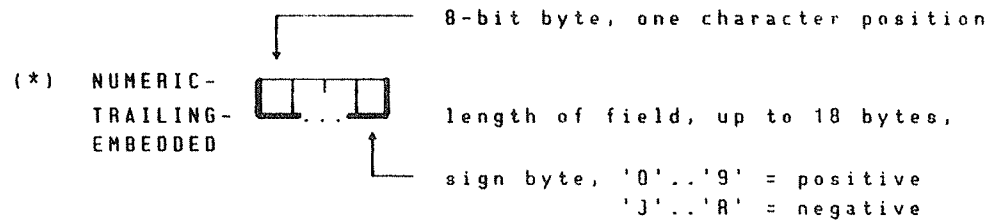


Figure 20. NUMERIC-TRAILING-EMBEDDED Data Type

In COBOL, the description PICTURE S9(5) NUMERIC-TRAILING-EMBEDDED, occupies 5 bytes.

(*) Note that the TRAILING-EMBEDDED representation is the default sign representation of the ANS COBOL standard. The embedded sign representation of the former ND-COBOL systems, (older than 1980) is not completely compatible with this standard, and may, in certain cases cause unpredictable results.

REAL This type represents floating-point data. The length and format of this type depends on the computer where the data was created.

The following list defines the type assumed in the different versions of the program and subroutine libraries:

```
ND-100, 48-bit version: 3 x 16-bit words = 6 bytes, (REAL*6)
ND-100, 32-bit version: 2 x 16-bit words = 4 bytes. (REAL*4)
ND-500, 32-bit version: 2 x 16-bit words = 4 bytes. (REAL*4)
```

Figure 21. REAL Data Type

The exact format can be found in the manual ND-100 Reference Manual ND-06.014, and ND-500 Reference Manual ND-05.009.

ND-100 vs ND-500 **Please note the different sizes of this data type in the ND-100 and the ND-500, since this may influence the position of the fields and the length of data records.**

CHAPTER 4
SOME ADDITIONAL COMMANDS

- HELP AND INFORMATION
- SCRATCH FILE
- ALTERNATIVE-COLLATING-SEQUENCE
- USING MAGNETIC TAPE AS INPUT OR OUTPUT
- SOME COMMANDS TO IMPROVE TIMING

HELP AND INFORMATION COMMANDS

These two commands may help you to find details on how to use the other commands in the SORT-MERGE program, and explain errors that have been found during interactive use of the program.

Format:

```
HELP [ <command-name> ]

INFORMATION
```

HELP command

The command has two functions: list available commands, or explain error messages. The text appears on the terminal.

[<command-name>]

If no name is specified, a list of all commands and their parameters is displayed.

If a command-name is given, a more detailed description of the command's function and its parameters is listed.

The name of the commands may be abbreviated.

Explanation of errors

After an error message has been displayed, typing HELP (without command-name) gives a short explanation of the error encountered.

Further details of error messages and explanations can be found on page 61.

INFORMATION

The command displays a short description of the most important commands, and the order in which they must be given to conduct a SORT or MERGE operation. This command directs its output to the terminal only.

This command has no parameters.

SCRATCH FILE

During the sort operation a work file is needed to store temporary results before writing the final output file.

Normally the program uses the "scratch" file assigned to all terminals and batch processors under the SINTRAN III operating system.

These scratch files are stored under a common user called SCRATCH, and the space reserved may be used for a variety of programs. Should there not be enough free space, the SORT-MERGE program may fail, and terminate with an error message.

This command allows you to define your own work file, which may under certain circumstances increase the speed of the SORT-MERGE program.

Format:

```
SCRATCH-FILE <name-of-file>
```

<name-of-file>

The name of a file, INDEXED or CONTIGUOUS, to be used as the standard scratch file. Default file type is :DATA.

The size of the scratch file

The size of the scratch file depends on the the input file(s), and the length of the records.

The size of input file(s) can be obtained by the @FILE-STATISTICS command.

Odd sized records

If the record-length is an odd number of bytes, the size of the scratch file must be twice the size of the sum of the input file(s).

When defining your own scratch file, you should make the file CONTIGUOUS, by specifying the required number of pages in the @CREATE-FILE command of the operating system.

For further details of the maximum size of input files, and the size of the scratch file, please see page 65.

ALTERNATIVE-COLLATING-SEQUENCE

In certain cases the character sequence used in the SORT or MERGE operation is not suited for the user, since in the standard ASCII character set, the digits and special symbols come before the uppercase letters, followed by the lowercase letters.

This command allows you to specify the name of a file containing this new character set, to be used for fields where the data type has been declared with ALTERNATIVE-ASCII.

Format:

```
ALTERNATIVE-COLLATING-SEQUENCE <name-of-file>
```

<name-of-file>

The name of a file containing the character sequence to be used. Default file type is :DATA.

Only one alternative character set may be active at one time.

What the file looks like

The contents of this file are a list of characters in the required sequence, each character on a single line, or several on the same line separated by a comma.

The file can be made by the PED or NOTIS-WP editors, as 7-bit ASCII-characters.

Characters not specified in the file is added to the list, in the order of their ASCII value. Non-graphic characters, such as line-feed, carriage-return, backspace, and so on, cannot be specified in the file.

The example below shows a file that makes a text field declared as ALTERNATIVE-ASCII to be sorted according to the "business" standard: first the alphabet with both uppercase and lowercase letters, followed by numbers and special symbols:

```
 ,A,a,B,b,C,c,D,d,E,e,F,f,G,g,H,h,I,i,J,j,K,k
L,l,M,m,N,n,O,o,P,p,Q,q,R,r,S,s,T,t,U,u,V,v,W
w,X,x,Y,y,Z,z,[,{,\,|,],},0,1,2,3,4,5,6,7,8,9
```

Figure 22. Alternative Character Set for Business Standard

Note that the first character is a blank (space), and that each line is terminated by a carriage-return and line-feed (CR+LF).

Below is an example of how a set of records are arranged using the normal sequence and the alternate "business" sequence:

Original sequence:	Normal sequence:	Alternative sequence:
ab123	AB123	ABCDE
AB123	ABCDE	AB123
abcde	ab123	abcde
ABCDE	abcde	ab123

Example 5. Sorting Data Using Alternative Collating Sequence

USING MAGNETIC TAPE AS INPUT OR OUTPUT

The following two commands may be necessary when using magnetic tapes as input or output to the SORT-MERGE program.

Normally the SINTRAN III operating system treats magnetic tapes in the same fashion as disk files; data is read or written in pages of 2048 characters. In some cases, where data is to be exchanged with "foreign" systems, the user may define different blocking of data.

The two commands are identical in terms of parameters and usage, except that one relates to reading tapes as input to the SORT or MERGE, and the other relates to writing output to tapes. The commands are therefore described together.

Format:

```
BLOCK-FACTOR-INPUT  <number>  [ < 'CHARACTERS' | 'RECORDS' > ]
BLOCK-FACTOR-OUTPUT <number>  [ < 'CHARACTERS' | 'RECORDS' > ]
```

<number>

The number of units, either specified as CHARACTERS or RECORDS, that gives the size of the input or output blocks. The maximum block size is 8K characters.

<unit>

Either 'CHARACTERS' or 'RECORDS'.

For 'CHARACTERS' the <number> is the length of blocks in bytes.

For 'RECORDS' the <number> is taken as records of fixed type, and the block-size is calculated as the product of the <number> and the <record-length> as specified in the RECORD-DESCRIPTION command.

If the parameter <unit> is not specified, the <number> parameter is assumed to be representing 'CHARACTERS'.

TWO COMMANDS FOR THE ND-500

The following two commands are available only in the ND-500 SORT-MERGE, and may be used to improve the time of the SORT or MERGE operation.

SECURE <'ON'|'OFF'>

This command is valid for the SORT command only, and indicates whether the scratch file or the input file should be used as temporary work file.

SECURE 'ON', which is the default setting, uses the scratch file, either the standard one or the one defined by the user, as work file.

SECURE 'OFF' uses the input file as work file. This leads to a faster sort, **but may cause serious damage to the input file** if the sort operation should fail.

ON-SEGMENT <'ON'|'OFF'>

This command is valid for SORT command only, and is used to reserve a large portion of physical memory as working area for the sorting operation.

ON-SEGMENT 'ON', reserves up to 256 pages of memory as working area for the sorting operation, thus significantly reducing the sort time.

ON-SEGMENT 'OFF', which is the default setting, uses a standard data-segment, 64 pages, as working area.

The use of this command may lead to inconvenience for others, since it takes exclusive use of scarce resources. This may lead to increased response-time for other tasks.

CHAPTER 5
USING THE SORT-MERGE FROM A USER-WRITTEN PROGRAM

THE SORT SUBROUTINE

In certain circumstances it may be more convenient for the end user of an application to be able to perform a SORT or MERGE operation directly from a program.

For example, the end user could provide the names of the input and output files only, the remaining parameters being embedded in the program.

The programmer calls the SORT or MERGE subroutine as a part of the program, specifying all necessary parameters; compile and load the program, with the SORT-MERGE library to build an executable program.

FORTRAN programs

A description of the subroutines and their parameters is given in the two following sections, and relates to FORTRAN programming language.

COBOL programs

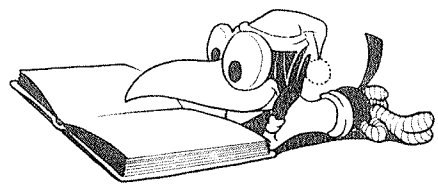
The COBOL programming language contains its own SORT and MERGE statements, which in turn calls these subroutines. However, the COBOL programmer need not load the SORT-MERGE library as a part of the program, as it is included in the COBOL runtime library.

Other languages

You may call the SORT subroutine from other programming languages provided that the language in question conform to the PLANC or FORTRAN subroutine calling conventions.

ND-100 vs ND-500

Please note the differences in sizes of the data types INTEGER and REAL in the ND-100 vs the ND-500, as this may influence the position of fields to be used by the sorting operation, as well as the length of data records. Further details can be found on page 29ff.



Format:

```
CALL SORT ( input, output, scratch, minlen, maxlen, rectype,
            no-of-fields, field-array, buffsize, buffarea,
            block-inp, block-out, coll-file, status )
```

<input>

The name of the input file in FORTRAN CHARACTER format, or an INTEGER containing a file number. If a file name is given, SORT opens the file; if an integer is given, SORT assumes the file is already opened. Default type is :DATA.

<output>

The name of the output file in FORTRAN CHARACTER format, or an INTEGER containing a file number. If a file name is given, SORT opens the file; if an integer is given, SORT assumes the file is already opened. Default file type is :DATA.

If the <output> parameter is given as an INTEGER containing the value zero (0), the input file is used as output file. This file must then be opened with random read-write access ('WX').

<scratch>

The name of a user defined scratch file in FORTRAN CHARACTER format, or an INTEGER containing a file number. Default file type is :DATA.

If the <scratch> parameter is given as an INTEGER containing the value zero (0), the system-defined scratch file is used.

<minlen>

An INTEGER, specifying the minimum length, in bytes, of the records in the file.

<maxlen>

An INTEGER, specifying the maximum length, in bytes, of the records in the file.

If the <rectype> parameter (see next) defines a file of the type FIXED, the <maxlen> parameter is ignored.

<rectype>

An INTEGER, defining the type of records in the file. The following values are accepted:

- 0 records are of FIXED type.
- 1 records are of TEXT type, that is each record is terminated by a carriage-return and a line-feed, as in the PED and NOTIS-WP 7-bit format, or COBOL 'RECORDING MODE T'.
- 2 records are of VARYING type, where each record contains a two-byte length field, as produced by COBOL programs with 'RECORDING MODE V'.

<no-of-fields>

An INTEGER, containing the number of fields to be used in the SORT comparison.

The maximum number of fields that can be sorted in one operation is 99.

<field-array>

An INTEGER ARRAY, consisting of four elements for each of the sort-fields that is specified in the parameter <no-of-fields>:

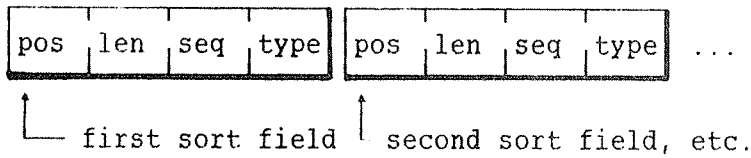


Figure 23. Field Array In Call Statements

<pos>

The first element specifies the position where the field begins in the record, expressed as byte positions, starting from 1.

<len>

The second element is the length of the field, in bytes.

<seq>

The third element is the sorting sequence, the value zero (0) for ASCENDING order, or the value one (1) for DESCENDING order.

<type>

The fourth element specifies the type of data, a number that gives the field's representation type from the following table:

code	data type	
0	ASCII	*)
1	ALTERNATIVE-ASCII	
2	NUMERIC-UNSIGNED	
3	NUMERIC-LEADING-SEPARATE	
4	NUMERIC-TRAILING-SEPARATE	
5	NUMERIC-LEADING-EMBEDDED	
6	NUMERIC-TRAILING-EMBEDDED	
7	INTEGER	*)
8	BCD	
9	ASCII-UPPER-CASE	
10	BITSTRING	
11	REAL	*)

*) Please note the difference in size of these data types on the ND-100 vs the ND-500, since this may influence the position of the fields, and the length of records.

Table 1. Data Types in Call Statement

Please refer to page 29, for detailed explanation of the different data types.

<buffsize>

An INTEGER, giving the size, in words, of the next parameter. This value must be larger than 4K words.

The size of this area greatly influences the speed of the sort operation, and should be as large as possible. See page 71.

<buffarea>

An INTEGER ARRAY, of the size specified in the previous parameter. This area is used during the sorting operation as working area.

<block-inp>

An INTEGER, specifying the size of magnetic tape blocks, in bytes, used for input. This parameter must be set to zero (0) if there are no tapes, or if the standard block size is employed.

<block-out>

An INTEGER, specifying the size of magnetic tape blocks, in bytes, used for output. This parameter must be set to zero (0) if there are no tapes, or if the standard block size is employed.

<coll-file>

The name of a file containing the ALTERNATIVE-ASCII character set, must be given if the type 2 is specified for any of the fields.

If there is no ALTERNATIVE-ASCII field in use, this parameter must be zero (0).

The name of the file must be given in FORTRAN CHARACTER format, or as an INTEGER containing the file number of a previously opened file. Default file type is :DATA.

The layout of this file is described on page 39.

<status>

An INTEGER, that is set to zero if the SORT terminates successfully, or non-zero to indicate an error.

The error number represents either an error from the SORT subroutine or an error detected in the operating system.

If the number is in the range 0 to 256 (400 octal) it denotes an operating system error, usually from the file system, and you should consult the SINTRAN III Reference Manual for an explanation.

The error numbers from the SORT subroutine, in the range of 2584 decimal (5030 octal) to 2604 decimal (5054 octal) are described on page 61.

THE MERGE SUBROUTINE

The MERGE operation can be embedded as a part of a user-written program, in the same way as with the SORT subroutine.

ND-100 vs ND-500

Please note the differences in sizes of the data types INTEGER and REAL in the ND-100 vs the ND-500, as this may influence the position of fields to be used by the sorting operation, as well as the length of data records. Further details can be found on page 29ff.

Format:

```
CALL MERGE (no-of-files, input, output, scratch, minlen, maxlen,
            rectype, no-of-fields, field-array, buffsize, buffarea,
            block-inp, block-out, coll-file, status )
```

<no-of-files>

An INTEGER, indicating the number of files to be used as input to the MERGE operation. This parameter controls the number of file names that must be specified in the next parameter.

The maximum number of files for input is 14.

<input>

A set of file names, in the FORTRAN CHARACTER format, giving names to the input files. All files are opened by the subroutine. Default filetype is :DATA.

<output>
<scratch>
<minlen>
<maxlen>
<rectype>
<no-of-fields>
<field-array>
<buffsize>
<buffarea>
<block-inp>
<block-out>
<coll-file>
<status>

All the remaining parameters are identical to the SORT subroutine.

EXAMPLE OF A USER PROGRAM

The following FORTRAN program should give you an example of how to call the SORT subroutine:

```
1234567
      PROGRAM SORTEX
      PARAMETER (IBUFSZ = 20000B)
      CHARACTER*32 INFILE, OUTFILE
      INTEGER BUFFER( IBUFSZ )
      INTEGER IFLDNO, IFIELDS( 8 )
      DATA    IFLDNO / 2 /, IFIELDS / 1,20,0,9, 25,5,1,2 /
      INFILE  = 'UNSORTED:DATA'
      OUTFILE = 'SORTED:DATA'
      IRECTYP = 1
      CALL SORT (INFILE,OUTFILE,0,10,80,IRECTYP,IFLDNO,IFIELDS,
     -           IBUFSZ, BUFFER, 0,0,0, ISTATUS)
      IF (ISTATUS .NE. 0) THEN
         WRITE (1,9000) ISTATUS
      ENDIF
9000  FORMAT(/, '*** ERROR IN SORT ***', O5)
      END
```

Example 6. User-Written Sort Program

The file is a 'TEXT' file, with record length varying from 10 to 80 characters, including the carriage-return and line-feed. There are two sort-fields, the first: position 1, length 20, sequence ASCENDING, type ASCII-UPPER, and the second: position 25, length 5, sequence DESCENDING, and type NUMERIC-UNSIGNED.

The PARAMETER-statement is used to define the size of the buffer-area, currently set to 20000 octal. This makes it easy to change the value to accommodate a larger area at a later stage.

LOADING ND-100 PROGRAMS

After the source program has been compiled, the object-code must be loaded together with the SORT-MERGE library, to make an executable program.

For the ND-100 computers, the following set of commands could be used:

```
@cc mode file to load ND-100 SORT-MERGE library
@cc (1-bank version)
@BRF-LINKER
prog-file sort-ex1
load sort-ex1
load SORT-MERGE-1B
load FORTRAN-1B
list-entries-defined
exit
@cc sort-ex1 is ready
```

Example 7. Loading a ND-100 Sort Program

You should look at the report from from the loader to find the size of remaining memory space, and adjust the parameters <buffsiz> and <buffarea> in the subroutine call accordingly. In the program example the buffer-size is defined by a PARAMETER-statement, if you change the value here, it automatically changes the size of the buffer-area also.

'2-banks' program

The ND-100 FORTRAN compiler can make object-code in the so-called 1-bank or 2-banks mode, using the compiler command $SEPARATE-DATA 'ON' or 'OFF'.

Larger buffer area

The 2-bank mode can be very useful, since it may allow larger work-area for the sorting operation.

SORT-MERGE-2B:BRF
FORTRAN-2B:BRF

In 2-bank mode, you must use the corresponding libraries SORT-MERGE-2B:BRF and FORTRAN-2B:BRF during the loading. Mixing 1-bank and 2-bank object-code modules is not allowed, and will be reported by the loader. Check that all your source programs are compiled with the compiler command $SEPARATE-DATA ON, and that you load the correct versions of the libraries.

LOADING ND-500 PROGRAMS

For the ND-500 computer, the @ND-500-LINKAGE-LOADER must be used to load the object-code, and the executable program is called a 'domain':

```
@cc mode-file to load ND-500 SORT-MERGE Library
@ND-500 LINKAGE-LOADER
set-domain sort-ex1
Load-segment sort-ex1
Library-segment SORT-MERGE-500:NRF
List-map
end-domain
exit
@cc SORT-EX1 is ready
```

Example 8. Loading a ND-500 Sort Program

In the ND-500, the instruction-part and the data-part of a program are automatically separated, hence, there is only one set of library files.

HANDLING ERRORS IN THE USER PROGRAM

The SORT and MERGE subroutines return a value in the parameter <status> of the subroutine call. This is a zero (0) if the operation terminates successfully, or a positive value indicating that an error has occurred.

No messages are displayed by either SORT or MERGE when called as subroutines. It is the programmer's responsibility to test the status-code returned, and inform the end user.

```
1234567
      PROGRAM   EXAMPLE
      :
      INTEGER   STATUS
      :
      CALL SORT (.....................,STATUS)
      IF (STATUS) THEN
         STOP ' SORT IS READY'
      ELSE
         CALL MESSAGE(STATUS)
      ENDIF
      END
```

Example 9. Testing Status Code Returned From SORT Or MERGE Subroutine

The subroutine MESSAGE should display a text to the end user in a form and style that can easily be understood.

SINTRAN III error numbers

Error numbers less than 256 decimal (400 octal) come from the operating system. This is usually returned from the file system, indicating that a file does not exists, that the user does not have the proper file access, or that there is no more space for files, etc. Please refer to the SINTRAN III Reference Manual for further details.

SORT-MERGE error numbers

Error numbers in the range 2584 to 2604 decimal, (5030 to 5054 octal), are detected within the SORT-MERGE subroutines, and correspond to the following table:

Decimal	Octal	Text as displayed in the interactive mode
2584	5030B	I/O ERROR
2585	5031B	ERROR IN DECIMAL NUMBER
2586	5032B	NO SUCH COLLATING SEQUENCE
2587	5033B	ERROR IN OCTAL NUMBER
2588	5034B	SORT FILE TOO BIG FOR SPECIFIED BUFFER SIZE
2589	5035B	TOO MANY KEYS
2590	5036B	TOO LONG TOTAL KEY
2591	5037B	NO VALUE GIVEN FOR PARAMETER
2592	5040B	ERROR IN SPECIFYING ALTERNATIVE COLLATING SEQUENCE
2593	5041B	IMPOSSIBLE COMBINATION OF PARAMETER VALUES
2594	5042B	NO SUCH RECORD TYPE
2595	5043B	ILLEGAL VALUE FOR PARAMETER
2596	5044B	RECORD TOO BIG FOR BUFFER SIZE
2597	5045B	RECORD GREATER THAN MAX SIZE
2598	5046B	EOF FOUND WITHIN RECORD
2599	5047B	MISMATCH OF RECORD LENGTH AND FILE SIZE
2600	5050B	ILLEGAL COMMAND SEQUENCE..KEY DESCRIPTION MISSING
2601	5051B	TOO MANY INPUT FILES
2602	5052B	RECORD SMALLER THAN MINIMUM SIZE
2603	5053B	NO SUCH KEY TYPE
2604	5054B	ILLEGAL COMMND SEQUENCE..RECORD DESCRIPTION MISSING

Table 2. Error Numbers and Message Returned from the SORT Program

HANDLING ERRORS IN MODE OR BATCH JOBS

Termination condition

When running the SORT-MERGE as mode or batch jobs, the command file may include statements that test the termination condition of the program.

A 'Completion-Code' and a 'Standard Subsystem Identification', is set up in the operating system by the SORT-MERGE program.

Job Execution Control

The *JEC* subsystem can be used to test and inform the user of the COMPLETION-CODE, and perhaps to decide if tasks using the result of the SORT-MERGE program should be started or not.

Example of JEC

The following example illustrates the use of JEC commands, with test and display of the condition-code:

```
@JEC begin
@JEC define <input-file>,<result-file>
@JEC inquire <input-file>;
 'Give the file name and type of the file you want to sort:'
@JEC message;
 'Give name and type of the result-file.'
@JEC inquire <result-file>;
 'Do not give the name in quotes (".."), since the file will be created:'
@JEC create-file <result-file> 0
@JEC SORT-MERGE
record-description 80, 1, text
key-description 1, 10, ascending, ascii
sort <input-file>, <result-file>
exit
@JEC if completion-code = 0 go to 100
@JEC message 'SORT failed'
@JEC print-completion-code
@JEC end
@JEC 100: message 'SORT successfully done'
@JEC %other tasks if SORT-MERGE performed OK
@JEC copy terminal, <result-file>
@JEC end
```

Example 10. Job Execution Control (JEC) in Mode or Batch Jobs

In this example you can enter the names of the files to be sorted, perform the sorting operation, and print the sorted file on the MODE-output file. You can find more information about JEC in the SINTRAN III Utility Manual ND-60.151 .

CHAPTER 6
ERROR MESSAGES

- LIST OF ERROR CODES AND EXPLANATIONS

ERROR IN DECIMAL NUMBER

- Only digits 0..9 can be given in this field. Please retype.

NO SUCH COLLATING SEQUENCE

- Only collating sequence ASCENDING or DESCENDING can be given. Please retype.

ERROR IN OCTAL NUMBER

- Only digits 0..7 can be given in this OCTAL field. Please retype.

SORT FILE TOO BIG FOR SPECIFIED BUFFER SIZE

- The file is too large to be sorted by the SORT-MERGE program. The file must be divided into several shorter files, then merged.

TOO MANY KEYS

- The maximum number of sort-fields are 10 when using the SORT-MERGE program. A user-written program may handle more than 10 sort-fields at a time.

TOO LONG TOTAL KEY

- The size for the field is larger than the RECORD-SIZE, or more than 255 bytes.

NO VALUE GIVEN FOR PARAMETER

- Required parameter(s) has/have not been given value(s), and there are no default value(s).

ERROR IN SPECIFYING ALTERNATIVE COLLATING SEQUENCE

- The alternative-collating-sequence contains a character that is not allowed. Only characters between 0 and 177 octal can be given.

IMPOSSIBLE COMBINATION OF PARAMETER VALUES

- Possible inconsistency with the block-factor command and the record type or number of characters/records per block. If the record type is "TEXT", then the size of block must be given in characters only. Please check, and retype command.

NO SUCH RECORD TYPE

- The record types allowed are: "FIXED", "TEXT", or "VARYING". Please retype command.

ILLEGAL VALUE FOR PARAMETER

- Inconsistency with the block-factor command and the block size on the tape. Please check, and retype the command.

RECORD TOO BIG FOR BUFFER SIZE

- Reading a record has failed as it is to too big for buffer size. The program terminates.

RECORD GREATER THAN MAX SIZE

- The maximum record-length is larger than the maximum length given. Please check, and retype command.

EOF FOUND WITHIN RECORD

- End of file mark is found within a record. The program terminates.

MISMATCH OF RECORD LENGTH AND FILE SIZE

- The record-length given for fixed records is not dividable by the maximum number of characters in the file. Please check, and restart the SORT-MERGE program.

ILLEGAL COMMAND SEQUENCE..KEY DESCRIPTION MISSING

- Possibly the key-description command is wrong, or not given at all. Please check, and retype the command.

TOO MANY INPUT FILES

- The maximum number of files accepted by the MERGE command is 14. Please check, and retype the command.

RECORD SMALLER THAN MINIMUM SIZE

- The SORT operation cannot be performed correctly. A record is found in the file which is smaller than the minimum size specified. Please check, and restart the SORT-MERGE program.

NO SUCH KEY TYPE

- One of the sort-fields specifies an unknown data type. The command HELP KEY-DESCRIPTION lists all available types. Please check, and retype the command.

ILLEGAL COMMAND SEQUENCE..RECORD DESCRIPTION MISSING

- Possibly the record-description command is wrong, or not given at all. Please check, and retype the command.

CHAPTER 7
THE CAPACITY OF SORT-MERGE

- SIZE OF INPUT FILE AND SCRATCH FILE

SIZE OF THE INPUT FILE

The file to be sorted is divided into partitions, which are sorted independently of each other. The sorted partitions are stored temporarily on the scratch file. When all partitions are sorted, they are merged and written to the output file. If the file is small, the entire file is sorted as one partition.

If not all sorted partitions can be merged in one pass (due to lack of available memory buffer area), they will be merged into greater partitions and stored temporarily back on the scratch file. The process is repeated until the number of partitions is less than the maximum number that can be merged in one pass.

The maximum input file size the SORT-MERGE program is capable to sort is approximately:

((30.000 * A) - 60.000.000) bytes,

where A is the buffer area size in bytes.

This gives the maximum size of the input file:

Buffer area in Kbytes	Maximum input file size in Megabytes *)
8	180 (90)
16	420 (210)
32	900 (450)
64	1860 (930)
128	3780 (1830)

*) Note: If the record size is an odd number of bytes, the maximum size is half (number in parenthesis), but the required size of the scratch file is shown in the left column.

The buffer area for the @SORT-MERGE-100 is 94 Kbytes, and for @SORT-MERGE-500 256 Kbytes, and may thus handle files of up to 2700 and 7620 Megabytes respectively.

SIZE OF THE SCRATCH FILE

The size of the scratch file depends on three factors: the size of the input file, the length of the records in the file, and the size of the buffer area in the sorting program.

Rules of thumb

In most cases when using the SORT-MERGE program, it is easy to determine the size of the scratch file:

- the scratch file uses the same number of pages as the input file
- if the record length is specified as an odd number of bytes, the scratch file must be twice as large as the input file

The @FILE-STATISTIC command displays the size of the input file. The command @USER-STATISTIC for user SCRATCH, shows the number of pages that is in use and that is reserved; the difference is the free space available.

The disk space for these scratch files are also used by other users on the computer. Care must therefore be emphasized since taking up all available space for the sorting job, may cause inconvenience for others.

Calculating the required size

The buffer area for the current version @SORT-MERGE-100 program is 94 Kbytes, and for the @SORT-MERGE-500 it is 256 Kbytes. If you are using a user-written sorting program, you must check the size of the buffer area with the programmer.

Depending on the size of the input file, the record length, , and the size of the buffer area, the three following situations may occur:

- the sort can be performed within the buffer area without using the scratch file at all
- the scratch file must be equal to the size of the input file
- the scratch file must be twice the size of the input file

To calculate the size of the scratch file, the following numbers are used:

B : the size of the buffer area, in bytes,
r : the maximum record length in bytes, and
ι : the size of the input file,

following the steps below.

Calculate the numbers α and β, and then compare them with ι. The outcome of this determines the size of the scratch file.

Step 1) to find if the sort needs any scratch file at all, calculate α:

record length <u>even</u>:

$$\alpha = \frac{B \times r}{(r+6)}$$

record length <u>odd</u>:

$$\alpha = \frac{B \times 2r}{(2r+6)}$$

If ι is less than or equal to α, then the sorting process can be performed within the buffer area, without using a scratch file at all.

If ι is greater than α, a scratch file is required.

Step 2) to find the size of the scratch file, calculate β:

$$\beta = \alpha \times \frac{B}{1K}$$

If ι is less than or equal to β, then the scratch file requires the same size as the input file.

If ι is greater than β, then the scratch file requires <u>twice</u> the size as the input file.

CHAPTER 8
METHODS USED

- SHORT INTRODUCTION TO MSD-RADIX SORT AND MERGE

SORTING

A most-significant-digit-first radix sort algorithm is used to sort the partitions, also referred to as the MSD-radix sort.

The number of records sorted in each partition is determined as the integral number of buffer size/record length. The sorting is performed from the most significant byte towards the least significant byte. Records with identical k first bytes in their keys are chained together. The sorting of their k+1'th key position will generally split the chain into several subchains. When a chain contains a single record, its position can be determined and this record is not involved in any further processing. The sequence of sorted records is built up in an array and each record will be moved once (at most). The terminal sort condition is reached when:

$$\frac{n}{C^{k}} = 1 \text{ until } k < kmax$$

where: n is the number of records in the partition,
C is the number of different characters in the key alphabet,
k is the average number of key characters to be processed,
kmax is the total number of key characters in a record

This means that

$$k = \ln n / \ln C$$

If we roughly assume the sorting time (exclusive input/output, which is proportional to the record length), to be proportional to the numbers of characters processed (all records in main memory), the algorithm is always better than normal radix sort where all key positions are processed (in reversed order) (k=kmax). When either the key-alphabet or the key-length are reduced, the improvements of MSD-radix are rather poor. However, in practical cases the improvements are significant. With a record length of 80 characters (all key characters randomly distributed), key length of 20, C=26 (all letters) and n=1000 (number of records), the MSD-radix is 9 times faster. If the key is extended to cover all 80 characters, the difference will increase to about 36 times faster because of its independent of key length.

MERGING

The merging system simply compares the keys of the first records in each partition and outputs the least (if ascending sequence is specified) of them to the output file. This is repeated until all partitions are empty. The merging system uses a variable length buffer for each input file partition and one 2 Kb buffer for the output file.

If the number of partitions sorted is greater than the number of partitions the system is capable of merging in one pass, then the maximum number of partitions will be merged and stored temporarily back on the scratch file. This will be repeated until all sorted partitions are merged and stored back on the scratch file. The scratch file will now contain sorted partitions with greater

partition size and a smaller number of partitions. A new pass of merging will be started and the process repeated until all partitions can be merged and written to the output file.

The number of passes the merge process will require is:

$$n = \left\lfloor \left| \log a \,/\, \log b \right| \right\rfloor + 1$$

Syntax: $\left\lfloor \left| -3.2 \right| \right\rfloor = 3$

$\left\lfloor \left| 3.2 \right| \right\rfloor = 3$

where: a is the number of partitions sorted from the sort phase and
b is the maximum number of partitions that can be merged.

$$a = \left\lfloor \left| F/A \right| \right\rfloor + 1 \quad \text{and}$$

$$b = \left\lfloor \left| \frac{A - U}{L + 16} \right| \right\rfloor$$

where: F is the size of the input file in bytes,
A is available memory buffer size in bytes,
U is the output buffer size, default 2048 bytes
L is the record length in bytes, or if the record length is an odd number of bytes, L is 2 * record length. This is due to even byte block transfer.

If a = 1 then the entire input file is sorted directly into the output file and no scratch file will be used. If a > 1 and n = 1 then the scratch file will be of the same size as the input file, and if v > 1, the scratch file needed is twice the size of the input file.

APPENDIX A
ASCII CHARACTER SET

- LIST OF ASCII CHARACTERS AND THEIR NUMERIC VALUE

CHAR	Byte Position Left	Byte Position Right	Dec.	CHAR	Byte Position Left	Byte Position Right	Dec.
NUL	000000	000000	0	0	030000	000060	48
SOH	000400	000001	1	1	030400	000061	49
STX	001000	000002	2	2	031000	000062	50
ETX	001400	000003	3	3	031400	000063	51
EOT	002000	000004	4	4	032000	000064	52
ENQ	002400	000005	5	5	032400	000065	53
ACK	003000	000006	6	6	033000	000066	54
BEL	003400	000007	7	7	033400	000067	55
BS	004000	000010	8	8	034000	000070	56
HT	004400	000011	9	9	034400	000071	57
LF	005000	000012	10	:	035000	000072	58
VT	005400	000013	11	;	035400	000073	59
FF	006000	000014	12	<	036000	000074	60
CR	006400	000015	13	=	036400	000075	61
SO	007000	000016	14	>	037000	000076	62
SI	007400	000017	15	?	037400	000077	63
DLE	010000	000020	16	@	040000	000100	64
DC1	010400	000021	17	A	040400	000101	65
DC2	011000	000022	18	B	041000	000102	66
DC3	011400	000023	19	C	041400	000103	67
DC4	012000	000024	20	D	042000	000104	68
NAK	012400	000025	21	E	042400	000105	69
SYN	013000	000026	22	F	043000	000106	70
ETB	013400	000027	23	G	043400	000107	71
CAN	014000	000030	24	H	044000	000110	72
EM	014400	000031	25	I	044400	000111	73
SUB	015000	000032	26	J	045000	000112	74
ESC	015400	000033	27	K	045400	000113	75
FS	016000	000034	28	L	046000	000114	76
GS	016400	000035	29	M	046400	000115	77
RS	017000	000036	30	N	047000	000116	78
US	017400	000037	31	O	047400	000117	79
SPACE	020000	000040	32	P	050000	000120	80
!	020400	000041	33	Q	050400	000121	81
"	021000	000042	34	R	051000	000122	82
#	021400	000043	35	S	051400	000123	83
S	022000	000044	36	T	052000	000124	84
%	022400	000045	37	U	052400	000125	85
&	023000	000046	38	V	053000	000126	86
'	023400	000047	39	W	053400	000127	87
(	024000	000050	40	X	054000	000130	88
)	024400	000051	41	Y	054400	000131	89
*	025000	000052	42	Z	055000	000132	90
+	025400	000053	43	[	055400	000133	91
,	026000	000054	44	\	056000	000134	92
—	026400	000055	45	]	056400	000135	93
.	027000	000056	46	^	057000	000136	94
/	027400	000057	47				

Table 3. The ASCII Character Set

CHAR	Byte Position Left	Right	Dec.	CHAR	Byte Position Left	Right	Dec.
—	057400	000137	95	o	067400	000157	111
	060000	000140	96	p	070000	000160	112
a	060400	000141	97	q	070400	000161	113
b	061000	000142	98	r	071000	000162	114
c	061400	000143	99	s	071400	000163	115
d	062000	000144	100	t	072000	000164	116
e	062400	000145	101	u	072400	000165	117
f	063000	000146	102	v	073000	000166	118
g	063400	000147	103	w	073400	000167	119
h	064000	000150	104	x	074000	000170	120
i	064400	000151	105	y	074400	000171	121
j	065000	000152	106	z	075000	000172	122
k	065400	000153	107		075400	000173	123
l	066000	000154	108		076000	000174	124
m	066400	000155	109		076400	000175	125
n	067000	000156	110		077000	000176	126
				DEL	077400	000177	127

APPENDIX B
SUMMARY OF SORT-MERGE COMMANDS

- SUMMARY OF SORT-MERGE COMMANDS

In the summary below the following notation is used:

words underlined	(:DATA)	indicate default values,
words in brackets	[...]	indicate optional items,
words in apostrophes	'.....'	indicate valid entries.

All parameters in the commands may be separated by blanks or commas.

ALTERNATIVE-COLLATING-SEQUENCE <File-name (:DATA) >

BLOCK-FACTOR-IN <Number> [<'RECORDS' | 'CHARACTERS'>]

BLOCK-FACTOR-OUT <Number> [<'RECORDS' | 'CHARACTERS'>]

EXIT (no parameters)

HELP [<command>]

INFORMATION (no parameters)

KEY-DESCRIPTION
<Position>
<Length (max: 255 bytes)>
<Sequence: 'ASCENDING' | 'DESCENDING'>
<Type: ['ASCII' | 'ASCII-UPPER' | 'ALTERNATIVE-ASCII' | 'BCD' | 'BITSTRING' | 'INTEGER' | 'NUMERIC-UNSIGNED' | 'NUMERIC-LEADING-SEPARATE' | 'NUMERIC-LEADING-EMBEDDED' | 'NUMERIC-TRAILING-SEPARATE' | 'NUMERIC-TRAILING-EMBEDDED' | 'REAL'] >
(... repeated for each sort-field ...)

MERGE
<Number-of-files (max: 14)>
<Input file-name (:DATA)>....
<Output file-name (:DATA)>

RECORD-DESCRIPTION
<Record-length: (minimum [: maximum]>
<Number-of-fields (max: 10)>
<Record-type: 'FIXED' | 'TEXT' | 'VARYING'>

SCRATCH-FILE <File-name (:DATA)>

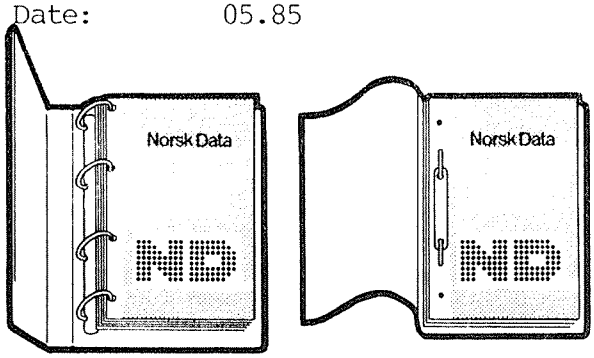
SORT
<Input file-name: (:DATA)>
<Output file-name: (:DATA)>

The SORT or MERGE-command may be repeated for several files, using the same RECORD-DESCRIPTION and KEY-DESCRIPTION commands.

Index

PRINTING RECORD	
PRINTING	NOTES
05.85	Version 1

Manual Name: ND-100/500 SORT-MERGE, User Guide
Manual No.: ND-60.236.1 EN
Date: 05.85



UPDATING

Manuals can be updated in two ways, new versions and revisions. New versions consist of a completely new manual which replaces the old one, and incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Customer Support Information and can be ordered from the address below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

RING BINDER OR PLASTIC COVER

The manual can be placed in a ring binder for greater protection and convenience of use. Ring binders may be ordered in two widths, 30 mm and 40 mm.

The manual may also be placed in a plastic cover. This cover is more suitable for manuals of less than 100 pages than for larger manuals.

Please send your order, as well as all types of inquiries and requensts for documentation to the local ND office, or (in Norway) to:

Norsk Data A.S
Graphic Center
P.O. Box 25 BOGERUD
N - 0621 OSLO 6 - Norway

I would like to order

. . . . . Ring Binders, 30 mm, at NOK 20.- per binder

. . . . . Ring Binders, 40 mm, at NOK 25.- per binder

. . . . . Plastic Covers, at NOK 10.- per cover

Name: . . . . . .

Company: . . . . . .

Address: . . . . . .

SEND US YOUR COMMENTS!

Are you frustrated because of unclear information in our manuals? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card – and an answer to your comments.

Please let us know if you:
– find errors
– cannot understand information
– cannot find information
– find needless information.

Do you think we could improve our manuals by rearranging the contents? You could also tell us if you like the manual.

Send to:
Norsk Data A.S
Documentation Department
P.O. Box 25 BOGERUD
N - 0621 OSLO 6 - Norway

NOTE!

This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

Manual Name: ND-100/500 SORT-MERGE, User Guide Manual number: ND-60.236.1 EN

Which version of the product are you using? _____

What problems do you have? (use extra pages if needed) _____

_____

_____

_____

_____

_____

Do you have suggestions for improving this manual? _____

_____

_____

_____

_____

_____

_____

Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

_____

What are you using this manual for? _____

_____

Norsk Data's answer will be found on the reverse side.

Answer from Norsk Data:

Answered by: Date:

Norsk Data A.S
Documentation Department
P.O. Box 25 BOGERUD
N - 0621 OSLO 6 - Norway

Systems that put people first

NORSK DATA A.S OLAF HELSETS VEI 5 P.O. BOX 25 BOGERUD 0621 OSLO 6 NORWAY
TEL.: 02 - 29 54 00 - TELEX: 18284 NDN