FUJITSU

FUJITSU Software PowerBSORT V7.0

User's Guide

Linux(64)

J2UL-1953-01ENZ0(00)
October 2014

Preface

PowerBSORT is a high-powered software product that can be used to sort, merge and copy application data files. The highly optimized sorting techniques employed by PowerBSORT are equally effective for small and very large data files. PowerBSORT offers the developer a variety of interfaces for batch execution and application integration.

Audience

Prior to using PowerBSORT it is assumed that you have the following knowledge:

- A basic understanding of Linux.
- Basic understandings of COBOL file systems.
- For C language integration, an understanding of C language inter-program communications.

Organization of This Documentation

This documentation is organized as follows:

Chapter 1 PowerBSORT Overview

This chapter explains the overview of PowerBSORT, the input output environment and the optional options.

Chapter 2 How to use PowerBSORT

This chapter explains how to run PowerBSORT.

Chapter 3 Using the PowerBSORT bsort command

This chapter explains the bsort command and options.

Chapter 4 Using the PowerBSORT bsortex command

This chapter explains the bsortex command and options.

Chapter 5 Using PowerBSORT with a COBOL program

This chapter explains how to use PowerBSORT in a NetCOBOL program.

Chapter 6 Using PowerBSORT with a C language program

This chapter explains how to use PowerBSORT in a C language program.

Chapter 7 Messages and Error codes

This chapter explains the output messages of the bsort command, the bsortex command, and the BSORT function, and the error detail codes of the BSORT function.

Appendix A Examples

This section provides sample executions and PowerBSORT coding examples.

Appendix B Notes

This section provides additional notes regarding the operation of PowerBSORT.

Related Manuals

The manual of this product and a related product includes the following manuals besides this document.

- PowerBSORT V7.0.0 Release Notes

Product Names

The names of the products described in this guide are abbreviated as follows.

Product name	Abbreviation
Red Hat(R) Enterprise Linux(R) 7 (for Intel64)	Linux

Product name	Abbreviation
Red Hat(R) Enterprise Linux(R) 6 (for Intel64)	

Trademarks

- NetCOBOL is a trademark or registered trademark of Fujitsu Limited or its subsidiaries in the United States or other countries or in both.
- Linux is a registered trademark of Linus Torvalds.
- Red Hat, RPM and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.
- Micro Focus and Net Express are registered trademarks of Micro Focus International Limited.
- In addition, the company names and the product names described in this book are the trademarks or registered trademarks of each company respectively.

Export Regulation

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Asking

- The contents of this manual may be revised without prior notice.
- No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Fujitsu Limited.

October 2014

Contents

Chapter 1 PowerBSORT Overview

PowerBSORT is a high-powered tool that can be used to sort, merge and copy application data files. PowerBSORT provides three primary functions: sort, merge and copy. Each of the functions provides a variety of processing options that are used to control the sort, merge and copy behavior.

1.1 Main Functions

The main functions of PowerBSORT include the sort, merge and copy function.

Sort function

Function to reorder records from a data file to ascending order (from 0 to 9 or from A to Z) or the descending order (from Z to A or from 9 to 0) by one or more specified key fields.

Merge function

Function to join two or more similarly sorted files.

Copy function

Function to copy one or more input file to output files.

1.2 Processing Options

In PowerBSORT, there are processing options that can be used by combining with the main functions.

Record selection option

The record selection option applies selection criteria to input records to determine their eligibility during sort, merge or copy operations. If the record selection is not used then all input records are selected for processing. The record selection option can be used with the sort, merge or copy functions.
Record selection is accomplished by specifying selection criteria. The criteria can compare fields in a record to other fields in the record, or to literal values. Records that fully match the criteria are selected for processing.

Example

The following example shows the result of selecting records where the ‘compared’ field is greater than the ‘comparing’ field. The order of the sorted records is determined by the ‘key’ field.

Input records

key	sel1	sel2
2	10	3
4	7	5
1	2	1
3	3	9

Output records

key	sel1	sel2
1	2	1
2	10	3
4	7	5

key : Key field sel1 : Compared field sel2 : Comparing field

Record reconstruction option

The record reconstruction option allows input records to be rearranged. The order of the fields in a record can be changed by this option, and literal values can be inserted a new fields. It is used with the sort, merge and copy functions.
When the reconstruction option is used, output records are built according to the list of fields you specify. Newly constructed records may

have one or more of the fields from the input record, and literal values inserted as needed. Output records are always constructed in the order specified by the reconstruction option.

Example

The following example shows the result of reconstructing records. The output record contains only the three fields selected from the input record; sorted by the ‘key’ field.

Input records

rcn1		rcn2		rcn3	
2		10		3	
4		7		5	
1		2		1	
3		3		9	

Output records

rcn1	rcn2	rcn3
1	2	1
2	10	3
3	3	9
4	7	5

rcn1 : Key field & Reconstruction field1 rcn2, rcn3 : Reconstruction field2, 3

Record summation option

The record summation option is used to summarize records having similar key fields, storing the sum of the summation fields in the output record. The summation option can be used with the sort or merge function only. The summation option cannot be used in combination with the copy function and the suppression option. The FIFO (first-in, first-out) option is disregarded when combined with the FIFO (first-in, first-out) option.
With this option, similarly keyed records are consolidated to a single output record. The specified summation fields in the output record are the accumulated values of the input records.

Example

The following example shows that the result of a record summation. The output records are consolidated to two records, one for each unique key from the input file. The ‘summation’ field contains the sum of the input records.

Input records

key		sum	
1		2	
2		4	
1		1	
1		5	

Output records

key		sum	
1		8	
2		4	

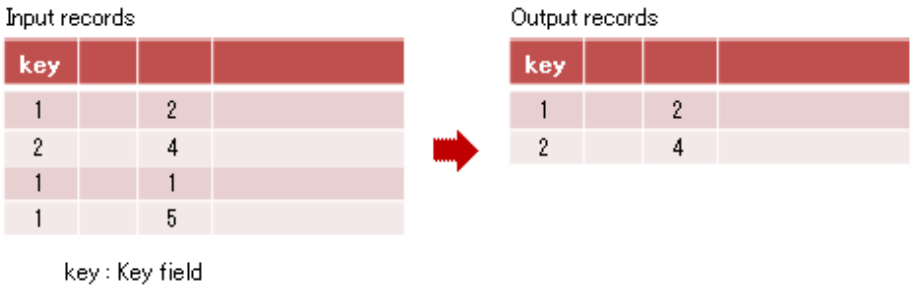
key : Key field sum : Summation field

Suppression option

The suppression option is used to filter records that are similarly keyed such that the output file contains only one record per key. This option can be used with the sort or merge function only. The suppression option cannot be used in combination with the copy function and the summation option. The FIFO (first-in, first-out) option is disregarded when combined with the FIFO (first-in, first-out) option.
When this option is used, similarly keyed records from the input file are ignored and not written to the output file.

Example

The following example shows that the result of using the suppression option. The first instance of the ‘key 1’ and ‘key 2’ is written to the output file. The additional instances of ‘key 1’ are ignored.

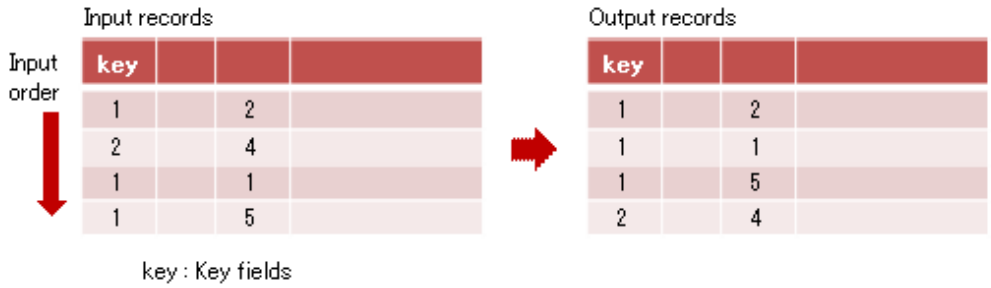


FIFO (first-in, first-out) option

The FIFO (first-in, first-out) option is used to ensure that similarly keyed records appear in the output file in the same sequence as they occur in the input file. The FIFO option can be used with the sort function. The FIFO (first-in, first-out) option is disregarded when combined with the merge function, the copy function, the record summation option, and the suppression option.
Use the FIFO options when the sequence of the records in output file of similarly keyed records must be maintained.

Example

The following example shows the result of using the FIFO option. The relative sequence of the ‘key 1’ records is maintained in the output file.



Note

When the FIFO (first-in, first-out) option is not used by the sort function, the order of outputting the record with the key field of equivalence is irregular.

1.3 Function / Processing Option Compatibility

Table 1.1 Availability of processing options for each main function

Processing Option	Main function		
	Sort	Merge	Copy
Record selection option	O	O	O
Record reconstruction option	O	O	O
Record summation option	O	O	X
Suppression option	O	O	X
FIFO option (FIFO: first-in first-out)	O	-	-

O: Available
X: Not available
-: The specification of the optional feature is disregarded

1.4 Environment variables and Startup file

1.4.1 Environment variables

The environment variable that influences the operation of PowerBSORT is shown below.

Table 1.2 Common environment variable of system

Environment variable	Meaning
LANG	Specification of character string used to specify internationalization information
LC_ALL	
LC_COLLATE	Specification of character collating sequence used
LC_CTYPE	Specification of width of character classification, character variable, and multiple byte character
LC_MESSAGES	Specification of language of message data base used
TMPDIR	Specification of directory path name of temporary file

Table 1.3 Peculiar environment variable of PowerBSORT

Environment variable	Meaning
BSORT_FIFO	FIFO (first-in, first-out) option
BSORT_UCS2TYPE	Unicode UCS-2 format byte order
BSORT_UTF32TYPE	Unicode UTF-32 format byte order
BSORT_UNICODEBOM	Whether or not BOM is skipped

LANG=locale, LC_ALL=locale

The default value of LC_*.
Refer to the manual of OS for details of environment variable LANG and LC_*.

LC_COLLATE=locale

Specify locale of the collation data base made by the localedef command.
Sort and merge use the collating sequence defined in locale.

LC_CTYPE=locale

Specify the width of the character classification, the character translation, and the multiple byte character.

LC_MESSAGES=locale

Specify the language of the message data base used.
For instance, the application can use the message data base for English and the message data base for Japanese properly.

TMPDIR=directory path name[:...]

Specify the directory path name of the temporary file.
Two or more directory path names can be specified by delimiting it by colon (:).When BSORT_TMPDIR of a startup file is omitted, it becomes effective.

BSORT_FIFO={ON | OFF}

The FIFO (first-in, first-out) option is specified.

Specified value	Meaning
ON	When the value of the specified key field is the same, those records are output in input order. This is effective for the sort function. The specification of the fifo (first-in, first-out) option is disregarded when combined with the merge function, the copy function, the record summation option, and the suppression option.
OFF	The order of records with the same key may be different in the output file compared to the input file.

Note

When this specification is omitted, the following specification becomes effective.

- FIFO option (-f) in bsort command
- fifo operand of -option option in bsortex command
- optionfunc of BSRTPRIM structure in BSORT function

BSORT_UCS2TYPE={BIG_ENDIAN | LITTLE_ENDIAN}

Specify the Unicode UCS-2 format byte order.
Valid when the data format is set to UCS-2 using the PowerBSORT command line interface.

Specified value	Meaning
BIG_ENDIAN	Sets Big Endian.
LITTLE_ENDIAN	Sets Little Endian.

Note

When this specification is omitted, the specification of BSORT_UCS2TYPE of a startup file becomes effective.

BSORT_UTF32TYPE={BIG_ENDIAN | LITTLE_ENDIAN}

Specify the Unicode UTF-32 format byte order.
Valid when the data format is set to UTF-32 using the PowerBSORT command line interface.

Specified value	Meaning
BIG_ENDIAN	Sets Big Endian.
LITTLE_ENDIAN	Sets Little Endian.

Note

When this specification is omitted, the specification of BSORT_UTF32TYPE of a startup file becomes effective.

BSORT_UNICODEBOM={ON | OFF}

Specify whether or not the Unicode byte order marker, BOM, is skipped.
Valid when the input file is a text file and the code set of that file is Unicode (UCS-2, UTF-32, or UTF-8).

If a BOM exists in the input file and is skipped, a BOM is still inserted in the output file. If multiple input files are specified, a BOM exists in the first input file specified, and the BOM was skipped、a BOM is inserted in the output file.

Specified value	Meaning
ON	Skips the BOM. Note that, if the head of the file is anything other than a BOM, it is not skipped.
OFF	Does not skip the BOM.

Note

When this specification is omitted, the specification of BSORT_UNICODEBOM of a startup file becomes effective.

1.4.2 Startup file

A startup file is a file that defines the default value of PowerBSORT.
Copy the model in user's home directory by the file name of ".bsortrc". The model of a startup file is installed in the following places.

```
/opt/FJSVXbsrt/config/BSORTRC
```

The startup file is in text format. Use the general text editor to change file contents. If the startup file contains identical features, the last entered specification is in effect. "#" indicates a comment line. In a model of startup file, all the definition values are the comment lines. The following variables can be defined in the startup file.

Table 1.4 Definition value of startup file

Definition value	Meaning
BSORT_CHKLEVEL	Checking level
BSORT_FILESYS_fs	Definition of file system
BSORT_FIFO	FIFO (first-in, first-out) option
BSORT_MEMSIZE	Work area size
BSORT_MSGFILE	Message output file name
BSORT_MSGLEVEL	Message output level
BSORT_MSGSTYLE	Output form of message
BSORT_OUTFILE_FSYNC	Specify whether to issue fsync, before closing the output files of standard file system
BSORT_SIGNEDZERO	Whether or not +0 and -0 are treated as the same value
BSORT_SUMFILD	Define the command termination status when a character other than numeric, signs, blank, or tab is found in text records summation field
BSORT_SUMOVER	Define the command termination status when an overflow occurs at the summation process
BSORT_SUMOVERCONT	Whether or not record summation continues after an overflow occurs
BSORT_SUMSHRT	Define the command termination status when the input records do not include the summation field during text input or variable-length records
BSORT_SYSLOG	syslog output
BSORT_TMPDIR	Temporary file directory name
BSORT_TOPYY	Defining the beginning year for data form of "two digit years"
BSORT_UCS2TYPE	Unicode UCS-2 format byte order
BSORT_UTF32TYPE	Unicode UTF-32 format byte order
BSORT_UNICODEBOM	Whether or not BOM is skipped

BSORT_CHKLEVEL={0 | 1}

The check option within PowerBSORT is specified.

Level	Meaning
0	Nothing checked **(Default value)**
1	Interface check with BSORT function is performed.

BSORT_FILESYS_fs=bslibname[:fslibname]

Defines the file system.

- fs

  Specify the identifier of the file system.
  The identifier of the file system that can be specified is shown below.

  - COB1, COB2, COB3

  Information

  - Specify the identifier of the file system as follows.
    - I/O file system option (-F) in bsort command
    - filesys operand of -input option or -output option in bsortex command
    - BSRTFILE structure in bsrtopen function
  - There is no difference in the identifier of the file system, and specify it arbitrarily.

- bslibname

  Specifies the PowerBSORT file access library.
  The following file access libraries are available.

File access library name	Meaning
libbscblrt64.so libbscblfast64.so	Library for NetCOBOL sequential files
libbscblps64.so libbscblpsfast64.so	Library for NetCOBOL physical sequential files
libbscblrl64.so	Library for NetCOBOL relative files
libbscblidx64.so	Library for NetCOBOL indexed files

- fslibname

  Specifies the file system library.
  The path name can be specified with an absolute path or a relative path. The following file system library can be specified.

File system library name	Meaning
libcobflm64.so	NetCOBOL file system library (Large file)

BSORT_FIFO={ON | OFF}

The FIFO (first-in, first-out) option is specified.

Specified value	Meaning
ON	When the value of the specified key field is the same, those records are output in input order. This is effective for the sort function. The specification of the fifo (first-in, first-out) option is disregarded when combined with the merge function, the copy function, the record summation option, and the suppression option.
OFF	The order of records with the same key may be different in the output file compared to the input file. **(Default value)**

BSORT_MEMSIZE=n

The size of the work area that PowerBSORT uses is specified in kilobytes.
The work area is specified within the range from 64 to 2097151 kilobytes. When 0 is specified, or specifying BSORT_MEMSIZE is omitted, the work area is automatically set. When the value of 2097152 or more is specified, 2097151 are assumed.
To run PowerBSORT efficiently, specify the memory size of 1/50 or more of the amount of input data. However, when the calculation value (1/50 of the amounts of input data) falls below 8192 kilobytes, 8192 kilobytes is recommended.

BSORT_MSGFILE=Message output file name

This specifies the file path name in which the message output by PowerBSORT is stored.
When a file name contains a blank, the file name need not be enclosed with double quotation (").

BSORT_MSGLEVEL={N | E | W | I | 0 | 1 | 2}

This specifies the level of messages output by PowerBSORT.

Level	Meaning
N	Nothing output.
E	Error messages are output.
W	Error messages and warning messages are output. **(Default value)**
I	Error messages, warning messages and information messages are output.

The following level can be specified for use with previous versions.

Level	Meaning
0	Nothing is output.
1	Error messages and warning messages are output. **(Default value)**
2	Error messages, warning messages and information messages are output.

BSORT_MSGSTYLE={1 | 2}

The output form of the message is specified.

Specified value	Meaning
1	Output the header, and the message text. This is the form output by the version before V5.
2	Output the header, the message type, the date, the message number, and the message text. **(Default value)**

BSORT_OUTFILE_FSYNC={ON | OFF}

Specify whether to issue fsync, before closing the output files of standard file system.

Specified value	Meaning
ON	fsync(2) is issued.

Specified value	Meaning
OFF	fsync(2) is not issued. **(Default value)**

BSORT_SIGNEDZERO={EFFECT | IGNORE}

For data that can be expressed as "+0" and "-0", specify whether "+0" and "-0" are to be treated as equivalent.

Specified value	Meaning
EFFECT	"+0" and "-0" are treated as different.
IGNORE	"+0" and "-0" are treated as equivalent. **(Default value)**

Information

- The following data forms can be expressed as “+0” and “-0”:
  - Packed decimal number
  - External decimal number
  - Leading separate signed number
  - Trailing separate signed number
  - Leading overpunch signed number
  - Trailing overpunch signed number
  - When n(BSOPT_NUMERIC) is specified for an operation of the key field and the selection field at the text file.
  - When N(BSOPT_CHARNUM) is specified for an operation of the key field at the text file.
- When EFFECT is specified, "+0" and "-0" becomes "+0">"-0".
- The version before V5 operate as if EFFECT was selected.

BSORT_SUMFILD=n

Define the bsortex command or bsort command termination status when a character other than numeric, signs, blank, or tab is found in text records summation field.
An arbitrary integer value from 0 to 9 can be specified in n. When specification is omitted, 0 is assumed.

BSORT_SUMOVER=n

Define the bsortex command or bsort command termination status when an overflow occurs at the summation process.
An arbitrary integer value from 0 to 9 can be specified in n. When specification is omitted, 0 is assumed.

BSORT_SUMOVERCONT={ON | OFF}

Specify whether or not record summation continues after an overflow occurs.

Specified value	Meaning
ON	The record summated immediately before the overflow occurred is output and subsequent records with the same key field are summated as separate records. This is repeated whenever an overflow occurs.
OFF	The record summated immediately before the overflow occurred is output. Subsequent records with the same key field are not summated. **(Default value)**

BSORT_SUMSHRT=n

Define the bsortex command or bsort command termination status when the input records do not include the summation field during text input or variable-length records.
An arbitrary integer value from 0 to 9 can be specified in n. When specification is omitted, 0 is assumed.

BSORT_SYSLOG={N | E | W | I}

Specifies the level of messages output to the syslog by PowerBSORT.

Level	Meaning
N	Nothing output. **(Default value)**
E	Error messages are output.
W	Error messages and warning messages are output.
I	Error messages, warning messages and information messages are output.

BSORT_TMPDIR=Temporary file directory name[:Temporary file directory name...]

A directory name specifying where PowerBSORT temporary files are created.
When you process to sort the file of the amount that cannot be processed within the memory alone, a temporary file is created in a directory specified by BSORT_TMPDIR. When the specified directory name does not exist, it is an error. You can also specify two or more directory names by delimiting the directories by colon (:). Specifying two or more directory names in different file systems could prevent the error due to free space shortage. Do not put an unnecessary character string such as a blank before and behind the colon (:) when you specify two or more directory names.
The directory that makes the temporary file observes the order of precedence and is decided. Refer to "Priority level of the directory that creates the temporary file" for the priority level.

BSORT_TOPYY=yy

Define the beginning year for data form of "two digit years".
For instance, if BSORT_TOPYY=50 is specified, the effective range is from 1950 to 2049. The default value for yy is 60. BSORT_TOPYY is used when a data form of "two digit years" field is specified.

BSORT_UCS2TYPE={BIG_ENDIAN | LITTLE_ENDIAN}

Specify the Unicode UCS-2 format byte order.
Valid when the data format is set to UCS-2 using the PowerBSORT command line interface.

Specified value	Meaning
BIG_ENDIAN	Sets Big Endian.
LITTLE_ENDIAN	Sets Little Endian. **(Default value)**

BSORT_UTF32TYPE={LITTLE_ENDIAN | BIG_ENDIAN}

Specify the Unicode UTF-32 format byte order.
Valid when the data format is set to UTF-32 using the PowerBSORT command line interface.

Specified value	Meaning
BIG_ENDIAN	Sets Big Endian.
LITTLE_ENDIAN	Sets Little Endian. **(Default value)**

BSORT_UNICODEBOM={ON | OFF}

Specify whether or not the Unicode byte order marker, BOM, is skipped.
Valid when the input file is a text file and the code set of that file is Unicode (UCS-2, UTF-32, or UTF-8).

If a BOM exists in the input file and is skipped, a BOM is still inserted in the output file. If multiple input files are specified, a BOM exists in the first input file specified, and the BOM was skipped、a BOM is inserted in the output file.

Specified value	Meaning
ON	Skips the BOM. Note that, if the head of the file is anything other than a BOM, it is not skipped. **(Default value)**
OFF	Does not skip the BOM.

1.4.3 Priority level

The priority level is decided to the specification of an environment variable, a startup file, the bsortex command, the bsort command, and the BSORT function.
The priority level is as follows.

1. Environment variable
2. bsortex command, bsort command, or BSORT function
3. Startup file

However, the directory that creates the temporary file and the language used are the following priority levels.

Priority level of the directory that creates the temporary file

1. tmpfile_tbl of BSRTFILE structure in BSORT function
2. BSORT_TMPDIR of startup file
3. tmpdir operand of -option option in bsortex command
4. Environment variable TMPDIR
5. Directory of system defaults (/tmp)

Priority level of the language used

1. Environment variable LC_CTYPE
2. Environment variable LC_ALL
3. Environment variable LANG

1.5 PowerBSORT Input Output Environment

This section explains the type and the combination of input and output files that PowerBSORT supports.

1.5.1 File types

Text file

Text files are document files composed of character-string data. Records are delimited by new line characters. It is not possible to exist together though in new line characters, there is CR, LF or CRLF. The first EOF character signifies end of file.

Binary fixed length file

Fixed length files are composed of records of fixed length containing character strings and hexadecimal data.

NetCOBOL sequential, physical sequential, index, and relative file

They are file formats supported in NetCOBOL, including older versions of the file system. Fixed length and variable length records are supported.

1.5.2 Combination of the file type of the input files and the output files

As for the file types of input file and output file, different file types can be specified within the range of the following combinations.

Input file type	Output file type
Text file	Text file
Binary fixed length file	Binary fixed length file NetCOBOL sequential, physical sequential, indexed, relative file
NetCOBOL sequential, physical sequential, indexed, relative file	Binary fixed length file (for the fixed-length record form only) NetCOBOL sequential, physical sequential, indexed, relative file

Note

The maximum file size that can be handled in PowerBSORT is based on the limitations of each file system. Refer to the manual of each file system for its limitation value.

1.6 Field and data formats

This section explains the fields and data formats that can be specified for each field.

1.6.1 Fields

Key fields are used by the Sort function and the Merge function, a summation field is used by the Record summation option, a selection field (compared field and comparing field) is used by the Record selection option, and reconstruction fields are used by the Record reconstruction option.

Key fields

Key fields are fields in a record used to determine collating sequence. The field definition consists of information on location, length, data format and operation method.

Summation fields

Summation fields are used in the summation option for summarizing fields. The summation field consists of information on location, length, data format and output form.

Selection fields

Selection fields are specified in the record selection option. The selection field consists of information on a compared field, comparing field or literal value and comparing operator. When two fields in a record are compared, the first specified field is called a compared field and the second is called a comparing field. When a field in a record is compared with a literal value, the first specified field is called a compared field.
A compared field consists of information on location, length, data format and operation method. A comparing field consists of information on location, length and data format of the field. A literal value is a character-string constant shown in constant of the decimal numbers or hexadecimal numbers and character strings.

Reconstruction fields

The reconstruction fields are specified in the record reconstruction option and consists of information on location and length of the field in the record.
For reconstruction fields, constants of the decimal numbers or the character-string constants shown in the hexadecimal numbers and or character strings can be specified as literal values. In the record reconstruction option, literal values are used for embedding arbitrary figures or characters in a record.

1.6.2 Specifying fields

Both fixed fields and floating fields can be specified.

Specifying fixed fields

Specifying fixed fields is a method for specifying a field by byte position from the start of the record. The position of the field is common in all records.

Specifying floating fields

Specifying floating fields is a method for specifying fields delimited by field separation characters. Floating fields are valid in text files only. Since a position of a field depends on the position of the field separation characters, the position of the field may be different in each record.
In the text file floating field specification, there are three forms:

- Text file floating fields
  An arbitrary field separation character can be specified, and double quotation marks that enclose the field are not considered.
- Text file CSV format
  The field separation character is a comma, and double quotation marks that enclose the field are considered.
- Text file TSV format
  The field separation character is a tab, and double quotation marks that enclose the field are considered.

- Text file floating field specification:
  - Double quotation marks (") at the beginning of the field are not treated as double quotation marks (") that enclose the field.
  - The key field, the summation field, the selection field, and the reconstruction field cannot contain the field separation character and the record separator.
  - When specifying field separation characters are omitted, blanks and tabs are used as the default field separation characters. When specifying field separation characters are omitted and blanks are consecutive, the first blank is the field separation character and PowerBSORT considers the remaining blanks to be part of the field.

    Example 1: When blanks are consecutive (the specification of the field separation character is omitted).

    ```
    field1___field2_field3

      The second field is "__field2".  "_" means blank.
    ```

  - When a field separation character is specified, PowerBSORT processes it considering that a null field exists if the field separation characters are consecutive.

    Example 2: When the second field is a null field (";" is specified for a field separation character).

    ```
    field1;;field3
    ```

  - The self-defined value cannot contain the field separation character and the record separator.
- Text file CSV format:
  - When the record separator and the field separation character (comma) are included in a field enclosed with double quotations ("), the field is treated as data.

    Example 1:

```
field1,"field2",field3

  The second field is "field2".
```

Example 2:

```
field1,"field2"data,field3

  The second field is "field2data".
```

Example 3:

```
field1,"field2,field2",field3

  The second field is "field2,field2".
```

- Double quotation marks (") in a field enclosed with double quotation marks (") are treated as two consecutive double quotation marks.

Example 4:

```
field1,"field2""field2",field3

  The second field is "field2” field2".
```

- When the field separation character (comma) is consecutive, it is considered that the null field exists.

Example 5:

```
field1,,field3

  The second field is a null field.
```

- The field consisting only of two double quotation marks (") is treated as a null field.

Example 6:

```
field1,"",field3

  The second field is a null field.
```

- If a field separation character (comma), a record separator, and double quotation marks (") are specified by the self-defined value, it is treated as a data field.

Example 7:

```
self-defined value : self,char

Actual data : "self,char"
```

Example 8:

```
self-defined value : self"char

Actual data : "self""char"
```

- Text file TSV format (see the text file CSV format for examples) :
  - When the record separator and the field separation character (tab) are included in the field enclosed with double quotation marks ("), the field is treated as data.
  - Double quotations marks (") in a field enclosed with double quotation marks (") is treated as two consecutive double quotation marks.

- When the field separation character (tab) is consecutive, it is considered that the null field exists.
- The field consisting only of two double quotation marks (") is treated as a null field.
- If a field separation character (tab), a record separator, and double quotation marks (") are specified by the self-defined value, it is treated as a data field.

1.6.3 Data format

The data formats include the following 3 types; character, numeric and number.

1.6.3.1 Character Data Types

ASCII code

Characters are compared in 8 bit transparent order. When two or more byte characters coexist, each character can be compared by specifying the option. ASCII code, EUC, and eight unit Japanese kana code can be processed. When the code system of the input file is EBCDIC, the data is sorted in the ASCII code order.

EBCDIC code

EBCDIC characters are compared in 8 bit transparent order. When the code system of the input file is ASCII, the data is sorted in the EBCDIC code order.

EUC

EUC are compared by character. EUC includes file codes in display format and internal codes (2 byte and 4 byte process codes). PowerBSORT uses the keywords: EUC for file codes, EU2 for 2 byte process codes, and EU4 for 4 byte process codes.

JEF code

Japanese code used in general-purpose computers of Fujitsu. Characters are compared in 16-bit transparent order.

Unicode

It is an International character code in which characters all over the world can be described. In PowerBSORT, the following three forms are supported.

- UCS-2 form
  It is the form that 1 character in Unicode is shown in 2 bytes. UCS-2 and UTF-16 can also be handled.
- UTF-32 form
  It is the form that 1 character in Unicode is shown in 4 bytes.
- UTF-8 form
  It is the form that 1 character in Unicode is converted to 8 bit code from 1 to 6 bytes. In PowerBSORT, 1 to 3 bytes are supported.

Character form two digit years

Enables comparison of 2 byte numbers as 2 digit years according to a value specified by the BSORT_TOPYY of startup file.

Modify collation sequence

The character in one byte is replaced with other characters according to information on the given collation order and fields are compared.

1.6.3.2 Numeric Data Types

Unsigned binary number

An arbitrary bit in 1 byte is taken out and used as a comparing value of a compared field or comparing field. Mask values are used for taking out the bit. The logical product of the mask value and the value of the field become the comparing values.

Fixed-point binary number

Fixed-point binary number of which the leading 1 bit is a sign. The forms of the fixed-point binary numbers include the form named little endian used by 8086 family MPU and the form of big endian. When it is simply described as the fixed-point binary number, it is in the form of big endian.

Fixed-point binary number - little endian

It is a field of the fixed-point binary number used by 8086 family MPU. This form is known as little endian.

Fixed-point binary number - system dependent

When the native of the system is big endian, it means it is the fixed-point binary number. When the native of the system is little endian, it means it is fixed-point binary number - little endian.

Unsigned fixed-point binary number

It is a fixed-point binary number without a sign flag. It includes a form named little endian used by 8086 family MPU and a form of big endian. When it is simply described as the unsigned fixed-point binary number, it is in the form of big endian.

Unsigned fixed-point binary number - little endian

It is a field of the unsigned fixed-point binary number used by 8086 family MPU. This form is known as little endian.

Unsigned fixed-point binary number - system dependent

When the native of the system is big endian, it means it is the unsigned fixed-point binary number. When the native of the system is little endian, it means it is the unsigned fixed-point binary number - little endian.

IEEE floating-point binary number

Binary expression with exponent and mantissa. The first bit is a sign.

M floating-point binary number

Floating-point binary number used with a Fujitsu general-purpose computer. Binary expression with exponent and mantissa. The first bit is a sign.

Packed decimal number

It is an expression form in decimal number that the decimal numbers of 2 digits are stored in a byte.

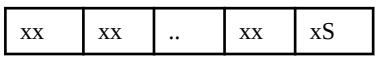

```
x:0x0-0x9
S:[Positive value]0xa, 0xc, 0xe, 0xf
  [Negative value]0xb, 0xd

Note) When other codes than the ones described as positive or negative in the table appear in the
position of the sign to be shown in 4 bits, they are handled as the positive sign.
```

Unsigned packed decimal number

The following two formats are supported in PowerBSORT:

- Unsigned packed decimal number (with sign nibble)

  This stores numbers, two decimal digits in a single byte, with sufficient length to store the decimal number. The last four bits (nibble) are not signed.



```
x:0x0-0x9
S:0xf

Note) The value of the last four bits is not checked.
```

- Unsigned packed decimal number (without sign nibble)

  This expresses the decimal number without a sign nibble storing numbers as two decimal digits in a single byte.

xx	xx	..	xx	xx

```
x:0x0-0x9
```

External decimal number

It is an expression form in decimal number that the decimal numbers of 1 digit are stored in the form that can be displayed.

[ASCII code system (NetCOBOL form)]

3x	3x	..	3x	Sx

```
3:0x3
x:0x0-0x9
S:[Positive value]0x4
  [Negative value]0x5
```

[ASCII code system (Micro Focus COBOL form)]

3x	3x	..	3x	Sx

```
3:0x3
x:0x0-0x9
S:[Positive value]0x3
  [Negative value]0x7
```

[EBCDIC code system]

Fx	Fx	..	Fx	Sx

```
F:0xf
x:0x0-0x9
S:[Positive value]0xa, 0xc, 0xe, 0xf
  [Negative value]0xb, 0xd

Note) When other codes than the ones described as positive or negative in the table appear in the
position of the sign to be shown in 4 bits, they are handled as the positive sign.
```

Unsigned external decimal number

It is an expression form in decimal number that the decimal numbers of 1 digit are stored in the form that can be displayed.

[ASCII code system]

3x	3x	..	3x	3x

```
3:0x3
x:0x0-0x9
```

[EBCDIC code system]

Fx	Fx	..	Fx	Fx

```
F:0xf
x:0x0-0x9
```

External decimal form two digit years

2 byte external decimal numbers are compared as 2 digit years according to a specified value of BSORT_TOPYY. Define BSORT_TOPYY with a startup file.

Packed decimal form two digit years

2 byte packed decimal numbers are compared as 2 digit years according to a specified value of BSORT_TOPYY. Define BSORT_TOPYY with a startup file.

Decimal form two digit years

1 byte unsigned packed decimal numbers are compared as 2 digit years according to a specified value of BSORT_TOPYY. Define BSORT_TOPYY with a startup file.

1.6.3.3 Number Data Types

Unsigned number

Same as the Unsigned external decimal number.

Leading separate signed number

Expression form in decimal number that the decimal numbers of 1 digit are stored in the form that can be displayed with the sign section in the leading 1 byte.

[ASCII code system]

SS	3x	..	3x	3x

```
3:0x3
x:0x0-0x9
SS:[Positive value]0x2b
   [Negative value]0x2d
```

[EBCDIC code system]

SS	Fx	..	Fx	Fx

```
F:0xf
x:0x0-0x9
SS:[Positive value]0x4e
   [Negative value]0x60
```

```
Note) When other codes than the ones described as positive or negative in the table appear in the
position of the sign to be shown in 8 bits, they are handled as the positive sign.
```

Trailing separate signed number

Expression form in decimal number that the decimal numbers of 1 digit are stored in the form that can be displayed with the sign section in the trailing 1 byte.

[ASCII code system]

3x	3x	..	3x	SS

```
3:0x3
x:0x0-0x9
```

```
SS:[Positive value]0x2b
   [Negative value]0x2d
```

[EBCDIC code system]

Fx	Fx	..	Fx	SS

```
F:0xf
x:0x0-0x9
SS:[Positive value]0x4e
   [Negative value]0x60
```

```
Note)When other codes than the ones described as positive or negative in the table appear in the
position of the sign to be shown in 8 bits, they are handled as the positive sign.
```

Leading overpunch signed number

Expression form in decimal number that the decimal numbers of 1 digit are stored in the form that can be displayed. In the leading 1 byte, signs and numbers are stored.

[ASCII code system (NetCOBOL form)]

Sx	3x	..	3x	3x

```
3:0x3
x:0x0-0x9
S:[Positive value]0x4
  [Negative value]0x5
```

[ASCII code system (Micro Focus COBOL form)]

Sx	3x	..	3x	3x

```
3:0x3
x:0x0-0x9
S:[Positive value]0x3
  [Negative value]0x7
```

[EBCDIC code system]

Sx	Fx	..	Fx	Fx

```
F:0xf
x:0x0-0x9
S:[Positive value]0xa, 0xc, 0xe, 0xf
  [Negative value]0xb, 0xd
```

```
Note) When other codes than the ones described as positive or negative in the table appear in the
position of the sign to be shown in 4 bits, they are handled as the positive sign.
```

Trailing overpunch signed number

It is the same form as the External decimal number.

1.6.4 Data forms that can be specified in each field

The fields include the key field used in the sort option and merge option, the record summation field used in the record summation option, selection field (compared field and comparing field) used in the record selection option and reconstruction field used in the record reconstruction options.

1.6.4.1 Data formats that can be specified in the key field

The following shows the data formats that can be specified in each key field and its length. In the table, typ is used when specifying in the PowerBSORT command line interface and definition value is used in the BSORT function.

Table 1.5 For binary files

Type	Data format	typ	Defined value	Length(Byte)
Character	ASCII code	asc	BSKEY_ASC	1-Record length (Note 1)
	EBCDIC code (Note 2)	ebc	BSKEY_EBC	1-Record length (Note 1)
	EUC (Note 3)	euc	BSKEY_EUC	1-Record length (Note 1)
	EUC 2 byte process codes (Note 3)	eu2	BSKEY_EU2	2-Record length (Multiples of 2) (Note 1)
	EUC 4 byte process codes (Note 3)	eu4	BSKEY_EU4	4-Record length (Multiples of 4) (Note 1)
	JEF code (Note 4)	jef	BSKEY_JEF	2-Record length (Multiples of 2) (Note 1)
	Unicode UCS-2 form (Note 5)(Note 12) (Byte order of system standard)	uc2	BSKEY_UCS2	2-Record length (Multiples of 2) (Note 1)
	Big endian	u2b	BSKEY_UCS2B	2-Record length (Multiples of 2) (Note 1)
	Little endian	u2l	BSKEY_UCS2L	2-Record length (Multiples of 2) (Note 1)
	Unicode UTF-32 form (Note 5)(Note 13) (Byte order of system standard)	u32	BSKEY_UTF32	4-Record length (Multiples of 4) (Note 1)
	Big endian	u3b	BSKEY_UTF32B	4-Record length (Multiples of 4) (Note 1)
	Little endian	u3l	BSKEY_UTF32L	4-Record length (Multiples of 4) (Note 1)
	Unicode UTF-8 form (Note 5)	ut8	BSKEY_UTF8	1-Record length (Note 1)
	Character form two digit years (Note 6)	yyc	BSKEY_YYC	2, 4 or 8 (Note 7)
	Modify collation sequence	col	BSKEY_COL	1-Record length (Note 1)
Numeric	Unsigned binary number	bit	BSKEY_BIT	1-8(bit)
	Fixed-point binary number	fbi	BSKEY_FBI	1-256
	Unsigned fixed-point binary number	ufb	BSKEY_UFB	1-256
	Fixed-point binary number - little endian	fbl	BSKEY_FBL86	1-256

Type	Data format	typ	Defined value	Length(Byte)
	Unsigned fixed-point binary number - little endian	ufl	BSKEY_UFB86	1-256
	Fixed-point binary number - system depend	fbm	BSKEY_FBIM	1-256
	Unsigned fixed-point binary number - system depend	ufm	BSKEY_UFBM	1-256
	IEEE floating-point binary number	ifl	BSKEY_IFL	1-256
	M floating-point binary number	mfl	BSKEY_MFL	1-256
	Packed decimal number	pdl	BSKEY_PDL	1-256
	Unsigned packed decimal number (with sign nibble)	pdf	BSKEY_PDF	1-256
	Unsigned packed decimal number (without sign nibble)	pdu	BSKEY_PDU	1-256
	External decimal number	zdl	BSKEY_ZDL	1-256
	Unsigned external decimal number	zdu	BSKEY_ZDU	1-256
	Packed decimal form two digit years (Note 6)	yyp	BSKEY_YYP	2
	External decimal form two digit years (Note 6)	yyz	BSKEY_YYZ	2
	Decimal form two digit years (Note 6)	yyd	BSKEY_YYD	1
Number	Unsigned number	azu	BSKEY_AZU	1-256
	Leading separate signed number	als	BSKEY_ALS	2-256
	Trailing separate signed number	ats	BSKEY_ATS	2-256
	Leading overpunch signed number	alo	BSKEY_ALO	1-256
	Trailing overpunch signed number	ato	BSKEY_ATO	1-256

Table 1.6 For text files

Type	Data format	typ	Defined value	Length(Byte)
Character	ASCII code (Note 8)	asc	BSKEY_ASC	1-Record length (Note 1)
	EBCDIC code (Note 8)	ebc	BSKEY_EBC	1-Record length (Note 1)
	EUC (Note 3)	euc	BSKEY_EUC	1-Record length (Note 1)
	Unicode UCS-2 form (Note 9)(Note 12) (Byte order of system standard)	uc2	BSKEY_UCS2	2-Record length (Multiples of 2) (Note 1)
	Big endian	---	BSKEY_UCS2B	2-Record length (Multiples of 2) (Note 1)
	Little endian	---	BSKEY_UCS2L	2-Record length (Multiples of 2) (Note 1)
	Unicode UTF-32 form (Note 10)(Note 13) (Byte order of system standard)	u32	BSKEY_UTF32	4-Record length (Multiples of 4) (Note 1)
	Big endian	---	BSKEY_UTF32B	4-Record length (Multiples of 4) (Note 1)

Type	Data format	typ	Defined value	Length(Byte)
	Little endian	---	BSKEY_UTF32L	4-Record length (Multiples of 4) (Note 1)
	Unicode UTF-8 form (Note 11)	ut8	BSKEY_UTF8	1-Record length (Note 1)
	Character form two digit years (Note 6)	yyc	BSKEY_YYC	2, 4 or 8 (Note 7)
	Modify collation sequence (Note 8)	col	BSKEY_COL	1-Record length (Note 1)

NOTES

1. When the record reconstruction option of the input file is specified, up to the record length after the reconstruction of the input record can be specified.
2. Can be specified if the code system of the input file is ASCII or EBCDIC.
3. Can be specified if the code system of the input file is ASCII, and C or EUC locale is defined in the LANG environment variable.
4. Can be specified if the code system of the input file is EBCDIC.
5. Can be specified if the code system of the input file is other than the EBCDIC, and C or UTF-8 locale is defined in the LANG environment variable.
6. Years processing is executed according to the value specified by the BSORT_TOPYY. Define BSORT_TOPYY with a startup file.
7. When the code system of the input file is Unicode (UCS-2 form), the length is 4 bytes. When the code system of the input file is Unicode (UTF-32 form), the length is 8 bytes.
8. Can be specified if the code system of the input file is ASCII.
9. Can be specified if the code system of the input file is Unicode (UCS-2 form), and C or UTF-8 locale is defined in the LANG environment variable.
10. Can be specified if the code system of the input file is Unicode (UTF-32 form), and C or UTF-8 locale is defined in the LANG environment variable.
11. Can be specified if the code system of the input file is Unicode (UTF-8 form), and C or UTF-8 locale is defined in the LANG environment variable.
12. When specifying data format in the bsort command and the bsortex command, byte order follows the specification of the environment variable BSORT_UCS2TYPE or the BSORT_UCS2TYPE of startup file.
13. When specifying data format in the bsort command and the bsortex command, byte order follows the specification of the environment variable BSORT_UTF32TYPE or the BSORT_UTF32TYPE of startup file.

1.6.4.2 Data formats that can be specified in the summation field

The following shows the data formats that can be specified in the summation field and its length. In the table, typ is used when specifying in the PowerBSORT command line interface and the definition value is used in the BSORT function.

Table 1.7 For binary files

Type	Data format	typ	Defined value	Length(Byte)
Numeric	Fixed-point binary number	fbi	BSKEY_FBI	1-16
	Unsigned fixed-point binary number	ufb	BSKEY_UFB	1-16
	Fixed-point binary number - little endian	fbl	BSKEY_FBL86	1-16
	Unsigned fixed-point binary number - little endian	ufl	BSKEY_UFB86	1-16
	Fixed-point binary number - system depend	fbm	BSKEY_FBIM	1-16

Type	Data format	typ	Defined value	Length(Byte)
	Unsigned fixed-point binary number - system depend	ufm	BSKEY_UFBM	1-16
	Packed decimal number	pdl	BSKEY_PDL	1-16
	Unsigned packed decimal number (with sign nibble)	pdf	BSKEY_PDF	1-16
	Unsigned packed decimal number (without sign nibble)	pdu	BSKEY_PDU	1-16
	External decimal number	zdl	BSKEY_ZDL	1-32
	Unsigned external decimal number	zdu	BSKEY_ZDU	1-32
Number (Note 1)	Unsigned number	azu	---	1-32
	Leading separate signed number	als	---	2-32
	Trailing separate signed number	ats	---	2-32
	Leading overpunch signed number	alo	---	1-32
	Trailing overpunch signed number	ato	---	1-32

Table 1.8 For text files

Type	Data format	typ	Defined value	Length(Byte)
Character	ASCII code (Note 2)	asc	BSKEY_ASC	1-256 or Record length (Note 3)
	Unicode UCS-2 form (Note 4)(Note 7) (Byte order of system standard)	uc2	BSKEY_UCS2	2-256 or Record length (Multiples of 2) (Note 3)
	Big endian	---	BSKEY_UCS2B	2-256 (Multiples of 2) (Note 3)
	Little endian	---	BSKEY_UCS2L	2-256 (Multiples of 2) (Note 3)
	Unicode UTF-32 form (Note 5)(Note 8) (Byte order of system standard)	u32	BSKEY_UTF32	4-256 or Record length (Multiples of 4) (Note 3)
	Big endian	---	BSKEY_UTF32B	4-256 or Record length (Multiples of 4) (Note 3)
	Little endian	---	BSKEY_UTF32L	4-256 or Record length (Multiples of 4) (Note 3)
	Unicode UTF-8 form (Note 6)	ut8	BSKEY_UTF8	1-256 or Record length (Note 3)

NOTES

1. Can be specified only in the bsortex command.
2. Can be specified if the code system of the input file is ASCII.
3. When the bsortex command is used, up to the record length can be specified. With the bsort command, and the BSORT function, up to 256 bytes can be specified. When the record reconstruction option of the input file is specified, up to the record length after the input record is reconstructed can be specified.
4. Can be specified if the code system of the input file is Unicode (UCS-2 form), and C or UTF-8 locale is defined in the LANG environment variable.

5. Can be specified if the code system of the input file is Unicode (UTF-32 form), and C or UTF-8 locale is defined in the LANG environment variable.
6. Can be specified if the code system of the input file is Unicode (UTF-8 form), and C or UTF-8 locale is defined in the LANG environment variable.
7. When specifying data format in the bsort command and the bsortex command, byte order follows the specification of the environment variable BSORT_UCS2TYPE or the BSORT_UCS2TYPE of startup file.
8. When specifying data format in the bsort command and the bsortex command, byte order follows the specification of the environment variable BSORT_UTF32TYPE or the BSORT_UTF32TYPE of startup file.

1.6.4.3 Data formats that can be specified in the selection field

The selection fields specified in the record selection option include the compared field and comparing field.
This section shows the combination of the data formats that can be specified in the compared field, comparing field and literal value.

- Data formats that can be specified in the compared field and comparing field
- Combinations of the data formats that can be specified in the compared field and comparing field
- Combinations of the data formats that can be specified in the compared field and literal value

Data formats that can be specified in the compared field and comparing field

The following shows the data formats that can be specified in the compared field and comparing field. In the table, typ is used when specifying in the PowerBSORT command line interface and the definition values are used in the BSORT function.

Table 1.9 For binary files

Type	Data format	typ	Defined value	Length(Byte)
Character	ASCII code (Note 1)	asc	BSKEY_ASC	1-256
	EBCDIC code (Note 2)	ebc	BSKEY_EBC	1-256
	EUC (Note 3)	euc	BSKEY_EUC	1-256
	EUC 2 byte process codes (Note 3)	eu2	BSKEY_EU2	2-256 (Multiples of 2)
	EUC 4 byte process codes (Note 3)	eu4	BSKEY_EU4	4-256 (Multiples of 4)
	Unicode UCS-2 form (Note 4)(Note 11) (Byte order of system standard)	uc2	BSKEY_UCS2	2-256 (Multiples of 2)
	Big endian	u2b	BSKEY_UCS2B	2-256 (Multiples of 2)
	Little endian	u2l	BSKEY_UCS2L	2-256 (Multiples of 2)
	Unicode UTF-32 form (Note 4)(Note 12) (Byte order of system standard)	u32	BSKEY_UTF32	4-256 (Multiples of 4)
	Big endian	u3b	BSKEY_UTF32B	4-256 (Multiples of 4)
	Little endian	u3l	BSKEY_UTF32L	4-256 (Multiples of 4)
	Unicode UTF-8 form (Note 4)	ut8	BSKEY_UTF8	1-256
	Character form two digit years (Note 5)	yyc	----	2, 4 or 8 (Note 6)
	Modify collation sequence	col	BSKEY_COL	1-256
Numeric	Unsigned binary number	bit	BSKEY_BIT	1-8(bit)
	Fixed-point binary number	fbi	BSKEY_FBI	1-256

Type	Data format	typ	Defined value	Length(Byte)
	Unsigned fixed-point binary number	ufb	BSKEY_UFB	1-256
	Fixed-point binary number - little endian	fbl	BSKEY_FBL86	1-256
	Unsigned fixed-point binary number - little endian	ufl	BSKEY_UFB86	1-256
	Fixed-point binary number - system depend	fbm	BSKEY_FBIM	1-256
	Unsigned fixed-point binary number - system depend	ufm	BSKEY_UFBM	1-256
	Packed decimal number	pdl	BSKEY_PDL	1-256
	Unsigned packed decimal number (with sign nibble)	pdf	BSKEY_PDF	1-256
	Unsigned packed decimal number (without sign nibble)	pdu	BSKEY_PDU	1-256
	External decimal number	zdl	BSKEY_ZDL	1-256
	Unsigned external decimal number	zdu	BSKEY_ZDU	1-256
	Packed decimal form two digit years (Note 5)	yyp	----	2
	External decimal form two digit years (Note 5)	yyz	----	2
	Decimal form two digit years (Note 5)	yyd	----	1
Number	Unsigned number	azu	BSKEY_AZU	1-256
	Leading separate signed number	als	BSKEY_ALS	2-256
	Trailing separate signed number	ats	BSKEY_ATS	2-256
	Leading overpunch signed number	alo	BSKEY_ALO	1-256
	Trailing overpunch signed number	ato	BSKEY_ATO	1-256

Table 1.10 For text files

Type	Data format	typ	Defined value	Length(Byte)
Character	ASCII code (Note 7)	asc	BSKEY_ASC	1-256
	EUC (Note 3)	euc	BSKEY_EUC	1-256
	Unicode UCS-2 form (Note 8)(Note 11) (Byte order of system standard)	uc2	BSKEY_UCS2	2-256 (Multiples of 2)
	Big endian	---	BSKEY_UCS2B	2-256 (Multiples of 2)
	Little endian	---	BSKEY_UCS2L	2-256 (Multiples of 2)
	Unicode UTF-32 form (Note 9)(Note 12) (Byte order of system standard)	u32	BSKEY_UTF32	4-256 (Multiples of 4)
	Big endian	---	BSKEY_UTF32B	4-256 (Multiples of 4)
	Little endian	---	BSKEY_UTF32L	4-256 (Multiples of 4)
	Unicode UTF-8 form (Note 10)	ut8	BSKEY_UTF8	1-256
	Character form two digit years (Note 5)	yyc	----	2, 4 or 8 (Note 6)
	Modify collation sequence (Note 7)	col	BSKEY_COL	1-256

NOTES

1. Can be specified if the code system of the input file is other than the EBCDIC.
2. Can be specified if the code system of the input file is EBCDIC.

3. Can be specified if the code system of the input file is ASCII, and C or EUC locale is defined in the LANG environment variable.
4. Can be specified if the code system of the input file is other than the EBCDIC, and C or UTF-8 locale is defined in the LANG environment variable.
5. Can be specified in the bsortex command only. Years processing is executed according to the value specified by the BSORT_TOPYY. Define BSORT_TOPYY with a startup file.
6. When the code system of the input file is Unicode (UCS-2 form), the length is 4 bytes. When the code system of the input file is Unicode (UTF-32 form), the length is 8 bytes.
7. Can be specified if the code system of the input file is ASCII.
8. Can be specified if the code system of the input file is Unicode (UCS-2 form), and C or UTF-8 locale is defined in the LANG environment variable.
9. Can be specified if the code system of the input file is Unicode (UTF-32 form), and C or UTF-8 locale is defined in the LANG environment variable.
10. Can be specified if the code system of the input file is Unicode (UTF-8 form), and C or UTF-8 locale is defined in the LANG environment variable.
11. When specifying data format in the bsort command and the bsortex command, byte order follows the specification of the environment variable BSORT_UCS2TYPE or the BSORT_UCS2TYPE of startup file.
12. When specifying data format in the bsort command and the bsortex command, byte order follows the specification of the environment variable BSORT_UTF32TYPE or the BSORT_UTF32TYPE of startup file.

Combinations of the data formats that can be specified in the compared field and comparing field

The following shows the combinations of data formats that can be specified in the compared field and comparing field.

Table 1.11 For binary files

Type	Data format of compared field	Data format of comparing field
Character	ASCII code	ASCII code
	EBCDIC code	EBCDIC code
	EUC	EUC EUC 2 byte process codes EUC 4 byte process codes
	EUC 2 byte process codes	EUC EUC 2 byte process codes EUC 4 byte process codes
	EUC 4 byte process codes	EUC EUC 2 byte process codes EUC 4 byte process codes
	Unicode UCS-2 form (Byte order of system standard)	Unicode UCS-2 form (Byte order of system standard)
	Unicode UCS-2 form (Big endian)	Unicode UCS-2 form (Big endian)
	Unicode UCS-2 form (Little endian)	Unicode UCS-2 form (Little endian)
	Unicode UTF-32 form (Byte order of system standard)	Unicode UTF-32 form (Byte order of system standard)
	Unicode UTF-32 form (Big endian)	Unicode UTF-32 form (Big endian)
	Unicode UTF-32 form (Little endian)	Unicode UTF-32 form (Little endian)
	Unicode UTF-8 form	Unicode UTF-8 form
	Character form two digit years	Character form two digit years Packed decimal form two digit years

Type	Data format of compared field	Data format of comparing field
		External decimal form two digit years Decimal form two digit years
	Modify collation sequence	Modify collation sequence
Numeric	Unsigned binary number	Unsigned binary number
	Fixed-point binary number	Fixed-point binary number Fixed-point binary number - little endian Fixed-point binary number - system depend
	Unsigned fixed-point binary number	Unsigned fixed-point binary number Unsigned fixed-point binary number - little endian Unsigned fixed-point binary number - system depend
	Fixed-point binary number - little endian	Fixed-point binary number Fixed-point binary number - little endian Fixed-point binary number - system depend
	Unsigned fixed-point binary number - little endian	Unsigned fixed-point binary number Unsigned fixed-point binary number - little endian Unsigned fixed-point binary number - system depend
	Fixed-point binary number - system dependent	Fixed-point binary number Fixed-point binary number - little endian Fixed-point binary number - system depend
	Unsigned fixed-point binary number - system dependent	Unsigned fixed-point binary number Unsigned fixed-point binary number - little endian Unsigned fixed-point binary number - system depend
	Packed decimal number	Packed decimal number Unsigned packed decimal number (with sign nibble) Unsigned packed decimal number (without sign nibble) External decimal number Unsigned external decimal number
	Unsigned packed decimal number (with sign nibble)	Packed decimal number Unsigned packed decimal number (with sign nibble) Unsigned packed decimal number (without sign nibble) External decimal number Unsigned external decimal number
	Unsigned packed decimal number (without sign nibble)	Packed decimal number Unsigned packed decimal number (with sign nibble) Unsigned packed decimal number (without sign nibble) External decimal number Unsigned external decimal number
	External decimal number	Packed decimal number Unsigned packed decimal number (with sign nibble) Unsigned packed decimal number (without sign nibble) External decimal number Unsigned external decimal number
	Unsigned external decimal number	Packed decimal number Unsigned packed decimal number (with sign nibble) Unsigned packed decimal number (without sign nibble)

Type	Data format of compared field	Data format of comparing field
		External decimal number Unsigned external decimal number
	Packed decimal form two digit years	Character form two digit years Packed decimal form two digit years External decimal form two digit years Decimal form two digit years
	External decimal form two digit years	Character form two digit years Packed decimal form two digit years External decimal form two digit years Decimal form two digit years
	Decimal form two digit years	Character form two digit years Packed decimal form two digit years External decimal form two digit years Decimal form two digit years
Number	Unsigned number	Unsigned number Leading separate signed number Trailing separate signed number Leading overpunch signed number Trailing overpunch signed number
	Leading separate signed number	Unsigned number Leading separate signed number Trailing separate signed number Leading overpunch signed number Trailing overpunch signed number
	Trailing separate signed number	Unsigned number Leading separate signed number Trailing separate signed number Leading overpunch signed number Trailing overpunch signed number
	Leading overpunch signed number	Unsigned number Leading separate signed number Trailing separate signed number Leading overpunch signed number Trailing overpunch signed number
	Trailing overpunch signed number	Unsigned number Leading separate signed number Trailing separate signed number Leading overpunch signed number Trailing overpunch signed number

Table 1.12 For text files

Type	Data format of compared field	Data format of comparing field
Character	ASCII code	ASCII code
	EUC	EUC
	Unicode UCS-2 form (Byte order of system standard)	Unicode UCS-2 form (Byte order of system standard)
	Unicode UCS-2 form (Big endian)	Unicode UCS-2 form (Big endian)
	Unicode UCS-2 form (Little endian)	Unicode UCS-2 form (Little endian)
	Unicode UTF-32 form (Byte order of system standard)	Unicode UTF-32 form (Byte order of system standard)

Type	Data format of compared field	Data format of comparing field
	Unicode UTF-32 form (Big endian)	Unicode UTF-32 form (Big endian)
	Unicode UTF-32 form (Little endian)	Unicode UTF-32 form (Little endian)
	Unicode UTF-8 form	Unicode UTF-8 form
	Character form two digit years	Character form two digit years
	Modify collation sequence	Modify collation sequence

Combinations of data formats that can be specified in the compared field and literal value

The following shows the combination of the data formats that can be specified in the compared field and literal value.

Table 1.13 For binary files

Type	Data format of compared field	Literal value
Character	ASCII code	Character, Hexadecimal
	EBCDIC code	Character, Hexadecimal
	EUC	Character
	Unicode UCS-2 form	Character, Hexadecimal
	Unicode UTF-32 form	Character, Hexadecimal
	Unicode UTF-8 form	Character, Hexadecimal
	Character form two digit years	Character, Hexadecimal, Decimal number
	Modify collation sequence	Character, Hexadecimal
Numeric	Unsigned binary number	Decimal number
	Fixed-point binary number	Decimal number
	Unsigned fixed-point binary number	Decimal number
	Fixed-point binary number - little endian	Decimal number
	Unsigned fixed-point binary number - little endian	Decimal number
	Fixed-point binary number - system depend	Decimal number
	Unsigned fixed-point binary number - system depend	Decimal number
	Packed decimal number	Decimal number
	Unsigned packed decimal number (with sign nibble)	Decimal number
	Unsigned packed decimal number (without sign nibble)	Decimal number
	External decimal number	Decimal number
	Unsigned external decimal number	Decimal number
	Packed decimal form two digit years	Decimal number
	External decimal form two digit years	Decimal number
	Decimal form two digit years	Decimal number
Number	Unsigned number	Decimal number
	Leading separate signed number	Decimal number
	Trailing separate signed number	Decimal number
	Leading overpunch signed number	Decimal number
	Trailing overpunch signed number	Decimal number

Table 1.14 For text files

Type	Data format of compared field	Literal value
Character	ASCII code	Character, Hexadecimal
	EUC	Character
	Unicode UCS-2 form	Character, Hexadecimal
	Unicode UTF-32 form	Character, Hexadecimal
	Unicode UTF-8 form	Character, Hexadecimal
	Character form two digit years	Character, Hexadecimal, Decimal number
	Modify collation sequence	Character, Hexadecimal

1.6.4.4 Data formats that can be specified in the literal value of the reconstruction field

The following shows the data formats that can be specified in the literal value of reconstruction field and their length. In the table, typ is used when specifying in the PowerBSORT command line interface.

Table 1.15 For binary files

Literal value	Data format	typ	Length(Byte)
Character	ASCII code (Note 1)	asc	1-256
	EBCDIC code (Note 2)	ebc	1-256
	EUC (Note 3)	euc	1-256
	Unicode UCS-2 form (Note 4)(Note 9) (Byte order of system standard)	uc2	2-256 (Multiples of 2)
	Big endian	u2b	2-256 (Multiples of 2)
	Little endian	u2l	2-256 (Multiples of 2)
	Unicode UTF-32 form (Note 4)(Note 10) (Byte order of system standard)	u32	4-256 (Multiples of 4)
	Big endian	u3b	4-256 (Multiples of 4)
	Little endian	u3l	4-256 (Multiples of 4)
	Unicode UTF-8 form (Note 4)	ut8	1-256
Hexadecimal	ASCII code (Note 1)	asc	1-256
	EBCDIC code (Note 2)	ebc	1-256
	Unicode UCS-2 form (Note 4)(Note 9) (Byte order of system standard)	uc2	2-256 (Multiples of 2)
	Big endian	u2b	2-256 (Multiples of 2)
	Little endian	u2l	2-256 (Multiples of 2)
	Unicode UTF-32 form (Note 4)(Note 10) (Byte order of system standard)	u32	4-256 (Multiples of 4)
	Big endian	u3b	4～256 (4の倍数)

Literal value	Data format	typ	Length(Byte)
	Little endian	u3l	4～256 (4の倍数)
	Unicode UTF-8 form (Note 4)	ut8	1-256
Decimal number	Fixed-point binary number	fbi	1-16
	Unsigned fixed-point binary number	ufb	1-16
	Fixed-point binary number - little endian	fbl	1-16
	Unsigned fixed-point binary number - little endian	ufl	1-16
	Fixed-point binary number - system depend	fbm	1-16
	Unsigned fixed-point binary number - system depend	ufm	1-16
	Packed decimal number	pdl	1-16
	Unsigned packed decimal number (with sign nibble)	pdf	1-16
	Unsigned packed decimal number (without sign nibble)	pdu	1-16
	External decimal number	zdl	1-32
	Unsigned external decimal number	zdu	1-32
	Unsigned number	azu	1-32
	Leading separate signed number	als	2-32
	Trailing separate signed number	ats	2-32
	Leading overpunch signed number	alo	1-32
	Trailing overpunch signed number	ato	1-32

Table 1.16 For text files

Literal value	Data format	typ	Length(Byte)
Character	ASCII code (Note 5)	asc	1-256
	EUC (Note 3)	euc	1-256
	Unicode UCS-2 form (Note 6)(Note 9) (Byte order of system standard)	uc2	2-256 (Multiples of 2)
	Unicode UTF-32 form (Note 7)(Note 10) (Byte order of system standard)	u32	4-256 (Multiples of 4)
	Unicode UTF-8 form (Note 8)	ut8	1-256
Hexadecimal	ASCII code (Note 5)	asc	1-256
	Unicode UCS-2 form (Note 6)(Note 9) (Byte order of system standard)	uc2	2-256 (Multiples of 2)
	Unicode UTF-32 form (Note 7)(Note 10) (Byte order of system standard)	u32	4-256 (Multiples of 4)
	Unicode UTF-8 form (Note 8)	ut8	1-256

NOTES

1. Can be specified if the code system of the input file is other than the EBCDIC.
2. Can be specified if the code system of the input file is EBCDIC.
3. Can be specified if the code system of the input file is ASCII, and C or EUC locale is defined in the LANG environment variable.
4. Can be specified if the code system of the input file is other than the EBCDIC, and C or UTF-8 locale is defined in the LANG environment variable.

5. Can be specified if the code system of the input file is ASCII.
6. Can be specified if the code system of the input file is Unicode (UCS-2 form), and C or UTF-8 locale is defined in the LANG environment variable.
7. Can be specified if the code system of the input file is Unicode (UTF-32 form), and C or UTF-8 locale is defined in the LANG environment variable.
8. Can be specified if the code system of the input file is Unicode (UTF-8 form), and C or UTF-8 locale is defined in the LANG environment variable.
9. When specifying data format in the bsort command and the bsortex command, byte order follows the specification of the environment variable BSORT_UCS2TYPE or the BSORT_UCS2TYPE of startup file.
10. When specifying data format in the bsort command and the bsortex command, byte order follows the specification of the environment variable BSORT_UTF32TYPE or the BSORT_UTF32TYPE of startup file.

1.7 Environment configuration

This section explains the estimate of the work area and file space PowerBSORT uses.

1.7.1 Estimate of memory (work area) size that PowerBSORT uses

In PowerBSORT, the best size of the work area for executing the sort merge processing is calculated and the high-speed sort merge processing can be achieved. PowerBSORT uses a work area of up to 2 gigabytes or less. When a user specifies the memory size in an environment variable or a command option, the work area to exceed the value specified by the user is not secured.
To run PowerBSORT efficiently, we recommend to specify 1/50 or more of the memory size of the amounts of input data. However, when the calculation value (1/50 of the amounts of the input data) falls below 8192 kilobytes, we recommend that you specify 8192 kilobytes or more.

See

- BSORT_MEMSIZE=n
- 3.2.28 Memory size option (-y)
- 4.2.7.5 memsize operand
- memory_size

1.7.2 Estimate of amount of temporary file that PowerBSORT uses

PowerBSORT makes the temporary file while executing the sort function.
The approximation formula of the needed capacity of the temporary file is shown below. If enough work area of PowerBSORT is not secured for the amount of the input data, it might exceed the following approximate formula. Therefore, specify the directory with an enough free space for the directory that makes the temporary file.

```
Temporary file capacity=(Administration area + Key length + Record length) X Number of records
```

- Administration area: Administration area of PowerBSORT - Fixed value 24.
- Key length: Total of the values that the length of each key field specified is adjusted to be multiples of 8.
- Record length: Value adjusted the specified record length to be multiples of 8.
- Number of records: The number of input records.

See

- TMPDIR=directory path name[:...]
- BSORT_TMPDIR=Temporary file directory name[:Temporary file directory name...]

- 4.2.7.9 tmpdir operand
- tmpfile_tbl

Chapter 2 How to use PowerBSORT

This section explains how to run PowerBSORT.

2.1 Using the PowerBSORT command line interface

You can use PowerBSORT command line interface by using the bsort or bsortex command. These commands can also be embedded in standard batch files for automated processing.

See

- Chapter 3 Using the PowerBSORT bsort command
- Chapter 4 Using the PowerBSORT bsortex command

About the functional difference between the bsort command and the bsortex command

The bsortex command is enhancing of the function of the bsort command. The following options are implemented only in the bsortex command.

- **File separate output option** (maxfilesize operand, maxrecnum operand of -output option)

  Data can be divided and output into two or more files. The size in which a file is output can be specified by size of a file and the number of records. Also, when files reaches full of the file system, the files can be switched.

- **Selection option for output files** (include operand, omit operand, case operand of -output option)

  This option allows you to divide files according to the condition specified for the output record of each output file.

- **Reconstruction option for output files** (reconst operand of -output option)

  This option allows you to reconstruct records right before the record is written to the output file in addition to the record reconstruction in inputting.

- **Summation or suppression processing for output files** (-summary option)

  The record summation option and suppression option allow you to leave results to the first record and final record of 1 or more record group that have equal key values.

- **Recognizes EOF control statement character as the end of the file** (eof operand of -input option)

  If an input file is a text file, whether EOF control statement character is recognized as the end of the file or not can be specified.

2.2 Using PowerBSORT in a COBOL program

PowerBSORT is automatically used by NetCOBOL programs when fully installed. No additional coding or modification is required to the COBOL program to take advantage of PowerBSORT.

See

- Chapter 5 Using PowerBSORT with a COBOL program

2.3 Using PowerBSORT in a C language program

PowerBSORT can be used by describing the BSORT function in a C language program. The BSORT function includes the five functions available in the PowerBSORT library.

See

- Chapter 6 Using PowerBSORT with a C language program

Chapter 3 Using the PowerBSORT bsort command

This section explains the bsort command.

The bsort command allows you to execute PowerBSORT using the command line interface. Execution of bsort can be included in batch files. Return the termination status of 0 when the bsort command ends normally. Return the termination status other than 0 when the error occurs.

3.1 bsort command format

The forms of the bsort command include 3 forms. The options that can be specified differ respectively. Specify an appropriate option according to the option to use.

Notes for the bsort command description

- Lexical units enclosed with square brackets ([ ]) can be omitted.
- Curly braces ({ }) show selections of lexical units delimited with vertical bars (|).
- For the italics, values are set as required.
- ... shows that it can be specified repeatedly.
- Options without option arguments (-s,-m,-c,-h,-f,-u,-w,-v,-r) can be successively specified.
  Example) -sf
- Option arguments to be specified delimiting with commas (,) can be specified by delimiting with spaces in place of commas (,) and enclosing the entire option argument with double quotation marks (").
  Example) -p "*sel-def sel-def*..."

3.1.1 Using the sort option

The command form for using the sort option is shown below.

```
bsort  [ -s ] [ -h ] [ -v ] -z recsize [ -Z recform ]
       [ -l msglevel ] [ -y memsize ] [ -G msgfile ]
       { [ -key-def [ ,key-def...] ] | [ -r ] }
       [ -T { flt | fix | csv | tsv }[ ,{ l | n | N } bdij ] ]
       { [ -t fldsep ] | [ -S fldsep-hex ] } [ -L line-delimiter ]
       [ -R skiprec-no [ ,skiprec-no...] ]
       [ -p sel-def [ ,sel-def...] ]
       [ -e recon-def [ ,recon-def...] ]
       [ -f ]
       { [ -u ] | [ -g sum-def [ ,sum-def...] ] }
       [ -a argfile ]
       { -w | -o outfile [ -o outfile...] }
       [ -F ofs,ifs [ ,ifs...] ]
       [ -I index-flg ] [ -X index-def [ ,index-def...] ]
       [ infile [ infile...] ] [ -q cdmode ] [ -Q altmode ]
       [ -x col-def [ ,col-def...] ]
```

3.1.2 Using the merge option

The command form for using the merge option is shown below.

```
bsort  -m [ -h ] -z recsize [ -Z recform ]
       [ -l msglevel ] [ -y memsize ] [ -G msgfile ]
       { [ -key-def [ ,key-def...] ] | [ -r ] }
       [ -T { flt | fix | csv | tsv }[ ,{ l | n | N } bdij ] ]
```

```
       { [ -t fldsep ] | [ -S fldsep-hex ] } [ -L line-delimiter ]
       [ -R skiprec-no [ ,skiprec-no...] ]
       [ -p sel-def [ ,sel-def...] ]
       [ -e recon-def [ ,recon-def...] ]
       { [ -u ] | [ -g sum-def [ ,sum-def...] ] }
       [ -a argfile ]
       { -w | -o outfile [ -o outfile...] }
       [ -F ofs,ifs [ ,ifs...] ]
       [ -I index-flg ] [ -X index-def [ ,index-def...] ]
       infile [ infile...] [ -q cdmode ] [ -Q altmode ]
       [ -x col-def [ ,col-def...] ]
```

3.1.3 Using the copy option

The command form for using the copy option is shown below.

```
bsort  -c [ -h ] -z recsize [ -Z recform ]
       [ -l msglevel ] [ -y memsize ] [ -G msgfile ]
       [ -T { flt | fix | csv | tsv } ]
       { [ -t fldsep ] | [ -S fldsep-hex ] } [ -L line-delimiter ]
       [ -R skiprec-no [ ,skiprec-no...] ]
       [ -p sel-def [ ,sel-def...] ]
       [ -e recon-def [ ,recon-def...] ]
       [ -a argfile ]
       { -w | -o outfile [ -o outfile...] }
       [ -F ofs,ifs [ ,ifs...] ]
       [ -I index-flg ] [ -X index-def [ ,index-def...] ]
       [ infile [ infile...] ]
       [ -x col-def [ ,col-def...] ]
```

3.2 bsort command arguments

The bsort command accepts a variety of arguments to control the operation of the command. This section explains each argument of the bsort command.

3.2.1 Argument file option (-a)

This specifies an argument file.
An argument file is a text file in which various option arguments of the bsort command are defined. An argument file is used to make a separate specification for a particular argument and when a command character string is longer than the maximum length allowed in the operating system.
The argument file must contain only bsort command option information. Individual options in the argument file can span more than one line, but must not include a line break. Argument file options cannot be specified in the argument file.
If other options are specified along with the argument file option, the option arguments specified in the argument file are interpreted as if they were inserted at the location of the argument file option. You can specify more than one argument file option.

Format

```
-a argfile
```

argfile

This specifies the argument file name.
If the argument file name contains spaces, it must be specified in double quotation marks (" ").

Example

The following shows an example of an argument file.

```
-s -z200 -32.12pdla,25.4fbia/n
-p 25.4fbi.ge.d35/n
-o sortout/n
sortin/n (Note)
```

NOTE

/n sign indicates the line feed code sequence <LF>.

Information

Assuming that the argument file (argfile) has the following content:

```
-s -z100
```

If

```
bsort sortin -a argfile -o sortout
```

is specified, it is the same as specifying

```
bsort sortin -s -z100 -o sortout
```

Similarly, if

```
bsort sortin -z70 -a argfile -o sortout
```

is specified, it is the same as specifying

```
bsort sortin -z70 -s -z100 -o sortout
```

In the latter case, the effect is that the -z option is specified twice. As the -z option can only be specified once, an error occurs.

3.2.2 Reconstruction field option (-e)

To use the record reconstruction option, specify the reconstruction field.
The field specified for the reconstruction field is output from right after the left of the output record sequentially. When the reconstruction field is specified, specify the field positions after the reconstruction for the key field and the summation field.

Format

```
-e recon-def [ ,recon-def ...]
```

- **Format 1 of *recon-def***

```
pos.len
```

Format 1 specifies a field in the input record.
When the specified field does not exist on the record of the input, it is an error.

- **Format 2 of *recon-def***

```
self.len typ [ opt ]
```

Format 2 specifies literal values.

- **Format 3 of *recon-def***

```
pos.END
```

Format 3 specifies from the position of the input record specification to the end of the reconstruction field.
If the specified field does not exist in the input record, an error occurs.

- **Format 4 of *recon-def***

```
EMPTY [ opt ]
```

Format 4 specifies an empty field.
Format 4 can be specified for the text file CSV format and the text file TSV format.

pos

This specifies positions of the reconstruction field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

len

This specifies a length of the reconstruction field in the number of bytes.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.
In the specification of description format 1, the length is not limited. For the information about each data format that can be specified in the description format 2, refer to Data formats that can be specified in the literal value of the reconstruction field.

Note

In the text file CSV format and text file TSV format specification, the length of the reconstruction field does not contain double quotation marks (“). When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

Example: Reconstruction field and length in text file CSV format.

Reconstruction field	Characters effective as reconstruction field	Length of reconstruction field
ABC	ABC	3bytes
"ABC"	ABC	3bytes
"A""B""C"	A"B"C	5byte
"A,B,C"	A,B,C	5byte

self

This specifies literal value. The description form of *self* is shown below.

- **Format 1 of *self***

```
'Character string'
```

- **Format 2 of *self***

```
xHexadecimal number
```

- **Format 3 of *self***

```
dDecimal number
```

Note

- The description method when a literal value is specified is different according to the shell used. The description example in a typical shell is shown below. When a literal value is specified for the argument file (argument file option (-a)), it is same as the method of describing the Bourne shell.
  - Bourne Shell
    - To specify a character string with a command argument, enclose the entire argument as shown below.

      Example) When you specify character string "ABC"

      ```
      -e "'ABC'.3asc"
      ```

    - To specify a character string including a double quotation with a command argument, specify a backslash (\) immediately before the double quotation (").The character with a special meaning as a shell is similar. (example: $, \, `)

      Example) When you specify character string ""ABC""

      ```
      -e "'\"ABC\"'.5asc"
      ```

    - To insert a quotation (') into a character string, specify two quotations in succession.

      Example) When you specify character string "'ABC'"

      ```
      -e "'''ABC'''.5asc"
      ```

  - C shell
    - The backslash(\) is specified before quotation (') before and behind the character string when the character string of a literal value is specified.

      Example) When you specify character string "ABC"

      ```
      -e \'ABC\'.3asc
      ```

    - The backslash (\) is specified before double quotation (") when the character string of a literal value which contains double quotation (") is specified. The character with a special meaning as a shell is similar. (example: $, &, (,), |, \, `, {, }, ;, *, <, >, ?, blank)

      Example) When you specify character string ""ABC""

      ```
      -e \'\"ABC\"\'.5asc
      ```

    - When the character string of a literal value which contains quotation (') is specified, it is continued two specification of the backslash (\) before quotation (').

      Example) When you specify character string "'ABC'"

      ```
      -e \'\'\'ABC\'\'\'.5asc
      ```

- When the character string of a literal value is specified, the character string should be one character or more.
- When you specify a decimal number of a literal value, signs can be specified.
- When you specify a text file fixed field, literal values cannot contain record separation character.
- When you specify a text file floating field, literal values cannot contain field separation character or record separation character.
- Output text file CSV format and text file TSV format with literal values enclosed in double quotation marks (") when the literal values contains the field separation character, the record separation character (Note) or double quotation marks ("). In this case, a literal value double quotation mark (") is output when two double quotation marks (") are consecutive.

  Note: Enclose literal values with double quotation marks (") when it is included by not only the record separator of the input file but also one of CRLF, CR, and LF permitted as a record separator.

```
Example: The data output to the reconstruction field when literal values contain the field
separation character (comma) or the double quotation marks (") is as follows:
```

```
Specification of literal values    Data output to reconstruction field
FIELD"2"                           "FIELD""2"""
FIELD2,3                           "FIELD2,3"
FIELD"2",3                         "FIELD""2"",3"
```

- When a value specified with a literal value and the length specified with *len* are different, process them as shown below.
  - When the literal value is character strings, the literal value is specified to start at the left and any blank spaces to the right is filled with the appropriate number of spaces if the length of the character strings specified in the literal value is shorter than the length specified in *len*. If the length of the character strings specified in the literal value is longer than the length specified in *len*, it is an error.
  - If the literal value is number, convert the value specified in the literal value into the data format and the length specified in *typ* and *len*, and embed them. If the conversion result is more than len, it is an error.
- When specifying literal value of fixed-point binary numbers, unsigned fixed-point binary numbers, fixed-point binary numbers - little endian, unsigned fixed-point binary numbers - little endian, fixed-point binary numbers - system dependent, and unsigned fixed-point binary numbers - system dependent as decimal numbers, a literal value can specify even the value expressible by the length specified with *len*.

Example

Length	Signed Data Format	Unsigned Data Format
1 byte	-128 - 127	0 - 255
2 byte	-32768 - 32767	0 - 65535
3 byte	-8388608 - 8388607	0 - 16777215
4 byte	-2147483648 - 2147483647	0 - 4294967295
:	:	:

typ

This specifies the data format of literal values.
For the information about each data format that can be specified, refer to Data formats that can be specified in the literal value of the reconstruction field.

Note

- In the binary file processing, when Input code system option (-q) is excluding EBCDIC, ASCII code can be specified.
- In the text file processing, when Input code system option (-q) is ASCII, ASCII code can be specified.
- When the Input code system option (-q) is EBCDIC, EBCDIC code can be specified.
- When the Input code system option (-q) is ASCII, and C or EUC locale is defined in the LANG environment variable, EUC can be specified.
- Unicode UCS-2 form can be specified, except when the Input code system option (-q) is EBCDIC at the binary file. When the Input code system option (-q) is Unicode (UCS-2 form), the Unicode UCS-2 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UCS-2 form can be specified.
- Unicode UTF-32 form can be specified, except when the Input code system option (-q) is EBCDIC at the binary file. When the Input code system option (-q) is Unicode (UTF-32 form), the Unicode UTF-32 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UTF-32 form can be specified.

- Unicode UTF-8 form can be specified, except when the Input code system option (-q) is EBCDIC at the binary file. When the Input code system option (-q) is Unicode (UTF-8 form), the Unicode UTF-8 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UTF-8 form can be specified.

END

Specify the fixed string 'END' at format 3 of *recon-def*.

EMPTY

Specify the fixed string 'EMPTY' at format 4 of *recon-def*.

opt

This specifies the operation of the reconstruction field.
It can be specified in text file CSV format or text file TSV format, at format 2 of *recon-def* or format 4 of *recon-def*. When this specification is omitted, it operates assuming that L is specified.

opt	Meaning
A	Enclose the reconstruction field with double quotation marks (").
L	Do not enclose the reconstruction field with double quotation marks ("). However, enclose it with double quotation marks (") when a field separation character, a record separation character (Note) or double quotation marks (") is included in the reconstruction field.

NOTE

Enclose literal values with double quotation marks (") when it is included by not only the record separator of the input file but also one of CRLF, CR, and LF permitted as a record separator.

Information

- In text file CSV format and text file TSV format, when format 1 of *recon-def* or format 3 of *recon-def* is specified, enclosing the reconstruction field with double quotation marks (") depends on the condition. When the field on the input record is enclosed with double quotation marks ("), the field after the record is reconstructed is enclosed with double quotation marks (").

  Example: Specify "-e1.3,2.2" for the reconstruction field.

```
Input record                Output record
"001","ABC",60              "ABC",60
"002","ABCDE",50            "ABC",50
"003","AB,CDE",40           "AB,",40
"004","AB"CDE,30 (Note)     "AB"C,30

NOTE
"AB" is enclosed by double quotation marks (") in the second field and "CDE" continues afterwards.
```

Example

1. To reconstruct a 10 byte field from the 5th byte of the input record, specify as follows.

```
-e 4.10
```

2. To specify the character strings of literal value (abc), specify as follows.

   For Bourne shell

```
-e "'abc'.3asc"
```

For C shell

```
-e \'abc\'.3asc
```

3. To specify the character strings of literal value (ab"cd) that contain double quotation mark ("), specify as follows.

For Bourne shell

```
-e "'ab\"cd'.5asc"
```

For C shell

```
-e \'ab\"cd\'.5asc
```

4. To specify the character strings of a literal value (abc'd) that contain quotation mark ('), specify as follows.

For Bourne shell

```
-e "'abc''d'.5asc"
```

For C shell

```
-e \'abc\'\'d\'.5asc
```

5. To specify hexadecimal number of a literal value (abc), specify as follows.

```
-e x616263.3asc
```

6. To specify external decimal number of a literal value (-32) by 4 bytes, specify as follows.

```
-e d-32.4zdl
```

7. To reconstruct the field from the 5th byte of the input record to the end of the record, specify as follows.

```
-e 4.END
```

8. To reconstruct a 8 byte field from the 3rd byte of the input record, and character strings of a literal value (,), and 5 byte field from the 20th byte of the input record, specify as follows.

For Bourne shell

```
-e "2.8,',.1asc,19.5"
```

For C shell

```
-e 2.8,\',\'.1asc,19.5
```

9. Enclose with double quotation marks ("), and specify the field in the text file CSV format and text file TSV format to specify the character string of literal value (abc).

For Bourne shell

```
-e "'abc'.3ascA"
```

For C shell

```
-e \'abc\'.3ascA
```

10. In text file CSV format and text file TSV format, specify the empty field enclosed with double quotation marks (") between the first field in the input record and the second field.

```
-e 0.1,EMPTYA,1.END
```

3.2.3 I/O file system option (-F)

This specifies file systems of the input files and the output files.

Format

```
-F ofs,ifs [ ,ifs ...]
```

ofs

This specifies the file system of the output file.
Specify a file system in an identifier. For more information, refer to the File system that can be specified.

ifs

This specifies the file system of the input file.
Specify a file system in an identifier. For more information, refer to the File system that can be specified.
File systems of input files can be specified corresponding to the input file. To specify two or more file systems, specify the file systems delimiting with comma (,) in the specified order of the input files. When the number of the input files is more than the number specified for the file system, the last file system specified for the file system is applied to the remaining input files. When the number of the input files is less than the number specified for the file system, the excessively specified file systems are ignored.

File systems that can be specified

The file systems that can be specified are shown below.
When cob1, cob2, and cob3 are specified for an identifier, it is necessary to define BSORT_FILESYS_fs of a startup file to associate the identifier with an actual file system. NetCOBOL file system can be specified for COBOL file system.

Identifier	Description of file systems
ufs	Native file system of the system **(Default)**
cobs64	NetCOBOL file system (sequential file)
cobp64	NetCOBOL file system (physical sequential file)
cobr64	NetCOBOL file system (relative file)
cobi64	NetCOBOL file system (indexed file)
cob1	COBOL file system
cob2	COBOL file system
cob3	COBOL file system

3.2.4 FIFO option (-f)

This specifies the FIFO option.
When the vales of the specified key fields are equal, the firstly input records are output first. This option is effective in the sort option. When omitted, the output order is not necessarily first-in first-out. When specified at the same time as the Merge option (-m), the Copy option (-c), the Summation field option (-g), and the Suppression option (-u), the specification of the first-in first-out (FIFO) option is disregarded.

Format

```
-f
```

Note

A priority level specified of the FIFO option is as follows.

1. Environment variable BSORT_FIFO
2. FIFO option (-f)
3. BSORT_FIFO of startup file

3.2.5 Message file option (-G)

This specifies file path name to output the messages.
By default, if the BSORT_MSGFILE of startup file is specified, it is output to the file. If BSORT_MSGFILE is not specified, standard error output is used.

Format

```
-G msgfile
```

msgfile

This specifies file path name to output the messages.
To specify a file name that contains a space, enclose the whole name with double quotation marks (").

Note

- Message file option (-G) takes priority over BSORT_MSGFILE of startup file.
- When the message file is specified, the information message is output to the message file. The warning message and the error message are output to the message file and the standard error output.

3.2.6 Summation field option (-g)

To use the record summation option, specify the summation field.
When the values of the specified key fields are equal, add the summation fields to make 1 record. In the case of text files, only the numbers described in ASCII code and Unicode are to be processed. This option is effective in the Sort option (-s) and the Merge option (-m). The record summation option has an exclusive relationship with the Suppression option (-u). When specified at the same time as the first-in first-out (FIFO) option (-f, BSORT_FIFO of startup file, or environment variable BSORT_FIFO), the specification of the first-in first-out (FIFO) option is disregarded.

Format

```
-g sum-def [ ,sum-def ...]
```

- **Format of *sum-def***

```
pos.len typ [ opt ]
```

pos

This specifies the position of the summation field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

len

This specifies the length of the summation field in the number of bytes.
For fields that are longer than the specified field, process the text file floating field, the text file CSV format, and the text file TSV format with the specified field length. When a field that is shorter than the specified field length appears, it enhances to the specified field length.

In the text file CSV format and text file TSV format specification, the length of the summation field does not contain the double quotation marks (") that enclose the field.

```
Example: Summation field and length in text file CSV format.
```

```
Summation field    Characters effective as    Length of summation field
                   summation field
123                123                        3bytes
"123"              123                        3bytes
```

typ

This specifies the data format of the summation field.
For more information, refer to the Data formats that can be specified in the summation field.

opt

This specifies the output format options of the summation field for a text file.
If these are specified for a binary file, an error occurs. The output format options that can be specified are shown below.

opt	Meaning
i	Appends a sign to the value in the summation field. This option and the u option are mutually exclusive.
u	Appends a minus sign (-) to the value in the summation field if that value is negative. This option and the i option are mutually exclusive.
z	If the length of the value in the summation field is less than the field length, the area to the left of the value is padded with zeroes (0). For example, if the value in the summation field is "1234", and the summation field length is six bytes, the value is recorded as "001234". This option, the b, and d options are mutually exclusive.
b	If the length of the value in the summation field is less than the field length, the area to the left of the value is padded with spaces. For example, if the value in the summation field is "1234", and the summation field length is six bytes, the value is recorded as " 1234". This option, the z, and d options are mutually exclusive.
d	Deletes any spaces, tabs, or zeroes that appear at the head of the summation field. If there are any spaces, tabs, or zeroes at the head of the summation field, the field is evaluated from the left and any spaces, tabs, or zeroes that appear before the first number (other than zero) are deleted. For example, if the value in the summation field is "00123", it is changed to "123". Note that, if the value in the summation field is equivalent to zero (such as "0", "0000" or "+000"), the last "0" is not deleted. This option can be specified for the text file floating field, text file CSV format, and text file TSV format. This option, the z, and b options are mutually exclusive.

- opt d can be specified for the text file floating field, text file CSV format, and text file TSV format. If it is specified for a text file fixed field, an error occurs.
- When summating with opt i specified, if the summated value becomes zero, a plus sign (+) is appended.

- When opt b is specified and a plus or minus sign exists, the sign is positioned immediately before the number.
- When neither opt i nor u is specified, processing is as follows:
  - If the summation result is a negative value, a minus sign (-) is included in the summation result.
  - If the summation result of unsigned data and signed data is a positive value, no sign is included in the summation result.
  - If two unsigned data items are summated, no sign is included in the summation result.
  - If signed and unsigned data are summated and the result is zero, no sign is included.
  - If two signed data items are summated, a sign is included in the summation result. If the summation result is zero, a plus sign (+) is included.
- When a sign is to be included in the value in the summation field, if the value in the summation field is already as long as the specified summation field length, the sign cannot be included and this causes an overflow to occur.
- When neither z, b nor d are specified, processing is as follows:
  - When the summation field is summated under the conditions listed below, if the length of the result in the summation field is less than the field length, the area to the left of the value is padded with zeroes (0).
    - When both summated data items are padded to the left with zeroes
    - When one summated data item is padded to the left with zeroes and the other is padded to the left with spaces
    - When one summated data item is padded to the left with zeroes and the other is padded to the left with tabs
  - When the summation field is summated under the conditions listed below, if the length of the result in the summation field is less than the field length, the area to the left of the value is padded with spaces.
    - When both summated data items are padded to the left with spaces
    - When both summated data items are padded to the left with tabs
    - When one summated data item is padded to the left with spaces and the other is padded to the left with tabs
- Summation fields of records that are not targeted for summation (that is, records with identical key field values for which no other record exists) are processed as follows:
  - If *opt* is specified, results are output in the format specified by *opt*.
  - If *opt* is not specified, results are output in the input format.

Example

1. To specify the field of the packed decimal number in 8 bytes from the 7th byte of the input record for the summation field, specify as follows:

```
-g 6.8pdl
```

2. To specify the field of ASCII code in 8 bytes in length from the 4th byte of the input record for the summation field in the text file. Appends a sign to the value in the summation field. If the length of the value in the summation field is less than the field length, the area to the left of the value is padded with spaces.

```
-g 3.8ascib
```

Note

- Of the records that are objects of the summation process, it is not possible to predict which record is output with the summation results.
- Specify so that the summation field does not overlap a key field or another summation field.

- Note also that the summation field must be completely included in the record.
- When using the record summation feature, the key field specification cannot be omitted.
- If an overflow occurs during addition of a summation field, subsequent behavior is determined by the specification of the BSORT_SUMOVERCONT of startup file.
- Numbers with decimal points cannot be summated.
- For text files, only single-byte numbers in ASCII code, Unicode UCS-2 format, Unicode UTF-32 format, or Unicode UTF-8 format are processed.
- With a text file floating field specification, if the summation field contains a field separation character and this causes the summation field position to change, correct processing cannot be guaranteed.
- With a text file floating field specification, summation results are processed to the specified field length. When a field that is longer than the specified field length appears, the part that exceeds the specified field length is output without changing the content. If the field is shorter than the specified field length, it is extended to the specified field length, and then processed.

  Example: Specify "1.5asc" for the summation field. Specify "0.3asca" for the key field. The field separation character string is a comma (,).

```
Input record                  Output record
001,12345ABC,OPQ (Note1)      001,12456ABC,OPQ (Note2)
001,111,RST                   002,00127,UVW (Note3)
002,15,UVW (Note1)
002,00112DEF,XYZ

NOTE
1. Assume that the record shown here is the one output by the record summation option.
2. Part ("ABC") that exceeds the specified field length is output without changing the content.
3. When the field is shorter than the specified field length, it is extended to the specified field
length.
```

- In the text file CSV format and text file TSV format, summation results are processed to the specified field length. When a field is longer than the specified field length, the part that exceeds the specified field length is not output. If the field is shorter than the specified field length, it is extended to the specified field length, and then processed.

  Example: Specify "1.5asc" for the summation field of the text file CSV format. Specify "0.3asca" for the key field.

```
Input record                  Output record
001,12345ABC,OPQ (Note1)      001,12468,OPQ (Note2)
001,123,RST                   002,00027,UVW (Note3)
002,15,UVW (Note1)
002,00012DEF,XYZ

NOTE
1. Assume that the record shown here is the one output by the record summation option.
2. Part ("ABC") that exceeds the specified field length is not output.
3. When the field is shorter than the specified field length, it is extended to the specified field
length.
```

- In the text file CSV format and the text file TSV format, whether the summation field is enclosed with double quotation marks (") or it doesn't enclose it is decided depending on the following conditions. When the field on the input record of the target for the output is enclosed with double quotation marks ("), the field after the record is summation is enclosed with double quotation marks ("). When the field on the input record of the target for the output is not enclosed with double quotation marks ("), the field after the record is summation is not enclosed with double quotation marks (").

  Example: Specify "1.5asc" for the summation field of the text file CSV format. Specify "0.3asca" for the key field.

```
Input record                  Output record
001,"12345",OPQ (Note)        001,"12468",OPQ
001,"123",RST                 002,"00027",UVW
```

```
002,"15",UVW (Note)           003,11900,GHI
002,00012,XYZ                 004,98769,MNO
003,11111,GHI (Note)
003,"00789",JKL
004,98765,MNO (Note)
004,4,PQR

NOTE
Assume that the record shown here is the one output by the record summation option.
```

- In the text file floating field specification, when a field that is shorter than the specified field length appears, it enhances to the specified field length. As a result, when the record length exceeds the specified maximum record length, it processes as an overflow.

3.2.7 Help option (-h)

This lists the command forms of the bsort command.
After the command syntax is analyzed, the Format is output to the standard error output. When this option is specified, processes in PowerBSORT are not executed. If the command syntax contains an error, the Format is output after the error message is output.

Format

```
-h
```

3.2.8 COBOL file index creating method option (-I)

This specifies the method for creating the index in the indexed file of COBOL file system.

Format

```
-I index-flg
```

index-flg

This selects and specifies from the following. When two or more are specified, specify options continuously.

index-flg	Meaning
c	The index is a compressed key.
r	The record is a compressed data.

3.2.9 Input file option (infile)

This specifies the input file.
Two or more input files can be specified. When two or more input files are specified, the record is input from the head of the file in the sort option and the copy option in specification the order. In the merge option, order by which the record is read depends on the value of the key field of each record. When omitted, it is treated as the standard input specification. The standard input specification cannot be used in the merge option.

Format

```
infile [ infile ...]
```

infile

This specifies input file path names.
As for the input file path names containing blanks, enclose the file name with double quotation marks (").

Example

As for the input file path names containing blanks, describe as follows.

```
"/in files/sortin 01"
```

3.2.10 Key field option (-key-def)

This specifies the key fields.
A key field is composed of position, length, data format, and operation.
Two or more key fields can be specified at the same time. When two or more key fields are specified, key field is delimited by the comma (*-key-def,key-def*). Moreover, two or more specification of one following the specification of the previous key field (*-key-def key-def*), and the key field options (*-key-def -key-def*) can be specified. If the specification of key fields is omitted, the whole records is considered to be a key field and sorted in ascending order of the code. To omit the key field specification and sort them in descending order, use the Descending option (-r).

Note

The key field cannot be omitted for text file CSV format and text file TSV format.

Format

```
-key-def [ ,key-def ...]
```

- **Format of *key-def***

```
pos.len typ [ opt ]
```

pos

This specifies the position of the key field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

len

This specifies the length of the key field in the number of bytes.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.
To specify an unsigned binary number for *typ*, specify mask value for *len* in decimal number. The logical products of the field value and mask value are considered to be the key value.

Note

The length of the key field does not contain double quotation marks (") that enclose the field for text file CSV format and text file TSV format. When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

Example: Key field and length in text file CSV format.

Key field	Characters effective as key field	Length of key field
ABC	ABC	3bytes
"ABC"	ABC	3bytes
"A""B""C"	A"B"C	5bytes
"A,B,C"	A,B,C	5bytes

typ

This specifies the data format of the key field.
For more information, refer to the Data formats that can be specified in the key field.

opt

This specifies operation of the key field.
To specify two or more operations, describe them successively. "a", "l", and "r" can be specified for binary files. All the operations can be specified for the text file. When both "a" and "r" are omitted, it operates assuming that "a" is specified. The operations that can be specified are shown below.

opt	Meaning
a	It sorts data in ascending order. It is in the exclusive relationship with "r".
b	Blanks and tabs in the head of the key field are disregarded.
d	Only the blank and the alphanumeric character are compared.
i	Control character codes are disregarded. Single shifts 2 (SS2) 0x8e and 3 (SS3) 0x8f of EUC are regarded as control characters when ASCII code is specified but regarded as characters when EUC is specified.
j	Lowercase letters are compared as uppercase letters.
l	Compares according to the collating sequence defined in LC_COLLATE environmental variable. Only when the data format is ASCII code, EUC, EUC 2 byte process codes, EUC 4 byte process codes or Unicode and the Input code system (-q) is ASCII code system or Unicode system, this option can be specified. It is in the exclusive relationship with "n" and "N".
n	Character strings of the numbers that contains signs are compared with arithmetic value. It can be specified only for text files. The result is not guaranteed when characters other than the numbers exist in the character string. When the data format is ASCII code, EBCDIC code or Unicode, it can be specified. It is in the exclusive relationship with "l" and "N".
N	Connected data of alphabet and numbers (for instance, data123) are evaluated and sorted separately for the alphabet and the numerical value. After only alphabets are targeted for the comparison and compared, character strings exclusively with numbers are compared with arithmetic value. Data is evaluated from the left of the specified field, and the data that appeared after effective data is disregarded. It can be specified when the data format is ASCII code, EBCDIC code or Unicode. It is in the exclusive relationship with "l" and "n".
r	It sorts the data in descending order. It is in the exclusive relationship with "a".

Note

- When modify collation sequence is specified for a data form of the key field, the operations other than "a" or "r" cannot be specified.
- When character form two digit years, packed decimal form two digit years, external decimal form two digit years and decimal form two digit years are specified for a data form of the key field, the operations other than "a" or "r" are disregarded.

Example

1. To specify to sort the fields of ASCII code in 10 bytes length from the head of the record in ascending order:

```
-0.10asca
```

2. To specify an external decimal number in 8 bytes length from the 5th byte of the input record, arranged in descending order:

```
-4.8zdlr
```

3. To specify ASCII code in 20 bytes length, arranged from the 10th byte of the input record in ascending order at the text file fixed field specification:

```
-9.20ascabj
```

The first blank and tab of the key field are disregarded, and lower case letters are treated as upper case letters.

Information

For the binary file variable-length record form or the text file, when the record where the key field does not exist is input, the value of the part where the key field does not exist is processed as 0.

Example 1) binary file variable-length record form or the text file fixed field:

```
key field:6.4asca

  0123456789 : Record where key field exists
  012345     : Record where key field doesn't exist
```

Example 2) the text file floating field, the text file CSV format or the text file TSV format:

```
key field:2.1asca

  fld0,fld1,fld2,fld3 : Record where key field exists
  fld0,fld1           : Record where key field doesn't exist
  fld0,fld1,,fld3     : Record where key field doesn't exist
```

3.2.11 Record separation character option (-L)

Specify separator string of each record in the text file. It is effective in text files.

Format

```
-L line-delimiter
```

line-delimiter

Select and specify one from the following.

line-delimiter	Meaning
cl	CRLF is treated as a record separator.
cr	CR is treated as a record separator.
lf	LF is treated as a record separator. **(Default value)**

Note

When two or more record separator exists in the input file, it is not possible to operate normally.

3.2.12 Message level option (-l)

Specify messages to be output.
When Message level option (-l) is omitted, BSORT_MSGLEVEL of startup file is used. When Message level option (-l) and the BSORT_MSGLEVEL of startup file specification are omitted, W (error message and warning message are output) is assumed for *msglevel*. The output destination of the message is decided by specifying Message file option (-G) or the BSORT_MSGFILE of startup file.

Format

```
-l msglevel
```

msglevel

The type of the output message is specified. The message level that can be specified are shown below.

msglevel	Meaning
N	Nothing is output.
E	The error message is output.
W	The error message and the warning message are output. **(Default value)**
I	The error message, the warning message, and the information message are output.

The type of the output message used in previous versions can also be specified.

msglevel	Meaning
0	Nothing is output.
1	The error message and the warning message are output. **(Default value)**
2	The error message, the warning message, and the information message are output.

Note

- The errors of the command syntax and command help are output to the standard error output even when specifying that nothing is output.
- Message level option (-l) takes priority over BSORT_MSGLEVEL of startup file.

3.2.13 Output file option (-o)

This specifies the output file.
When the Output file option (-o) and the Standard output option (-w) are specified at the same time, the specification of the Standard output option (-w) is disregarded. Files can be divided and output to two or more files as a recovery processing by specifying two or more Output file options (-o) when the output file abnormally terminates. Therefore, set on another file system when two or more files are specified. The recovery processing can be specified at the following files.

- System standard file (binary file or text file)
- NetCOBOL file uses the high speed access library for a NetCOBOL sequential file (libbscblfast64.so)
- NetCOBOL file uses the high speed access library for a NetCOBOL physical sequential file (libbscblpsfast64.so)

Format

```
-o outfile [ -o outfile ...]
```

outfile

This specifies output file path name.
As for the output file path names containing blanks, enclose the file name with double quotation marks (").

Example

1. The example of describing the output file path name that contains blank is shown below.

```
-o "/out files/output file"
```

2. When /output1/sortout1 is specified for the output file, and continuation is output to /output2/sortout2 as recovery processing when the output file exceeds the limit of the file system, it is specified as follows:

```
-o /output1/sortout1 -o /output2/sortout2
```

3.2.14 Selection field option (-p)

To use the record selection option, specify the selection field (selection condition).
Two or more selection fields can be specified. When two or more selection fields are delimited by the comma (,), it becomes the logical product of the selection condition. When two or more Selection field options (-p) are specified, it becomes the logical add of the selection condition.

Format

```
-p sel-def [ ,sel-def ...]
```

- **Format 1 of *sel-def***

```
pos.len typ [ opt ] .cmp.pos.len typ
```

For the format 1 of *sel-def*, two selection fields are compared. The left side of *cmp* shows the compared field and the right side shows the comparing field. An error occurs when the field to be compared or the comparing field does not exist on the input record.

- **Format 2 of *sel-def***

```
pos.len typ [ opt ] .cmp.self
```

For the format 2 of *sel-def*, the selection field is compared with a literal value. The left side of *cmp* shows compared field and the right side shows a literal value. An error occurs when the field to be compared does not exist on the input record.

pos

This specifies the position of the compared field or the comparing field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

len

This specifies the length of the compared field or the comparing field in the number of byte.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.
To specify an unsigned binary number for the data format, specify mask values. Compare the logical product of the mask value and field values. The mask value specifies the same value as the compared field and the comparing field.
For information about the lengths of each data format, refer to the Data formats that can be specified in the compared field and the comparing field.

Note

In text file CSV format and text file TSV format specification, the length of the compared field or the comparing field does not contain double quotation marks (") that enclose the field. When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

Example: Compared field and length in text file CSV format

Compared field	Characters effective as compared field	Length of compared field
ABC	ABC	3bytes
"ABC"	ABC	3bytes
"A""B""C"	A"B"C	5bytes
"A,B,C"	A,B,C	5bytes

self

This specifies the literal values. The description form of *self* is shown below.

- **Format 1 of *self***

```
'Character string'
```

  Specifies the character string for the format 1 of *self*.

- **Format 2 of *self***

```
xHexadecimal number
```

  Specifies the hexadecimal number for the character string for the format 2 of *self*.

- **Format 3 of *self***

```
dDecimal number
```

  Specifies the decimal number for numeric or number for the format 3 of *self*.

Note

- The description method when a literal value is specified is different according to the shell used. The description example in a typical shell is shown below. When a literal value is specified for the argument file (argument file option (-a)), it is same as the method of describing the Bourne shell.
  - Bourne Shell
    - To specify a character string with a command argument, enclose the entire argument as shown below.

      Example) When you specify character string "ABC"

      ```
      -p "0.3asc.eq.'ABC'"
      ```

    - To specify a character string including a double quotation (") with a command argument, specify a backslash (\) immediately before the double quotation (").The character with a special meaning as a shell is similar. (example: $, \, `)

      Example) When you specify character string ""ABC""

      ```
      -p "0.5asc.eq.'\"ABC\"'"
      ```

    - To insert a quotation (') into a character string, specify two quotations in succession.

      Example) When you specify character string "'ABC'"

      ```
      -p "0.5asc.eq.'''ABC'''"
      ```

- C shell
  - The backslash (\) is specified before quotation (') before and behind the character string when the character string of a literal value is specified.

    Example) When you specify character string "ABC"

    ```
    -p 0.3asc.eq.\'ABC\'
    ```

  - The backslash (\) is specified before double quotation (") when the character string of a literal value which contains double quotation (") is specified. The character with a special meaning as a shell is similar. (example: $, &, (,), |, \, `, {, }, ;, *, <, >, ?, blank)

    Example) When you specify character string ""ABC""

    ```
    -p 0.5asc.eq.\'\"ABC\"\'
    ```

  - When the character string of a literal value which contains quotation (') is specified, it is continued two specification of the backslash (\) before quotation (').

    Example) When you specify character string "'ABC'"

    ```
    -p 0.5asc.eq.\'\'\'ABC\'\'\'
    ```

- When the character string of a literal value is specified, the character string should be one character or more.
- When you specify a decimal number of a literal value, signs can be specified.
- Literal value is compared after adjusting to the data format of the compared field.
- When you specify a text file fixed field, literal values cannot contain record separation character.
- When you specify a text file floating field, literal values cannot contain field separation character or record separation character.
- For text file CSV format and text file TSV format, double quotation marks (") do not need to enclose literal values.
- When specifying literal value of fixed-point binary numbers, unsigned fixed-point binary numbers, fixed-point binary numbers - little endian, unsigned fixed-point binary numbers - little endian, fixed-point binary numbers - system dependent, and unsigned fixed-point binary numbers - system dependent as decimal numbers, a literal value can specify even the value expressible by 16 bytes.

typ

This specifies the data formats of the compared field and the comparing field.
For more information, refer to the Data formats that can be specified in the compared field and the comparing field.

Note

- When the data format of the compared field and comparing field are different, it compares the data after adjusting them to the data format of the compared field.
- When the data format of the compared field is character, it compares data in the shorter length of the compared field and comparing field.
- When the data format of the compared field is numeric or number, the shorter field of the compared field and comparing field are compared with the longer one after moving to the right end.
- In the binary file processing, when Input code system option (-q) is excluding EBCDIC, ASCII code can be specified.
- In the text file processing, when Input code system option (-q) is ASCII, ASCII code can be specified.
- When the Input code system option (-q) is EBCDIC, EBCDIC code can be specified.
- When the Input code system option (-q) is ASCII, and C or EUC locale is defined in the LANG environment variable, EUC, EUC 2 byte process codes, and EUC 4 byte process codes can be specified.

- Unicode UCS-2 form can be specified, except when the Input code system option (-q) is EBCDIC at the binary file. When the Input code system option (-q) is Unicode (UCS-2 form), the Unicode UCS-2 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UCS-2 form can be specified.
- Unicode UTF-32 form can be specified, except when the Input code system option (-q) is EBCDIC at the binary file. When the Input code system option (-q) is Unicode (UTF-32 form), the Unicode UTF-32 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UTF-32 form can be specified.
- Unicode UTF-8 form can be specified, except when the Input code system option (-q) is EBCDIC at the binary file. When the Input code system option (-q) is Unicode (UTF-8 form), the Unicode UTF-8 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UTF-8 form can be specified.

opt

This specifies the operations of the compared field.
To specify two or more operations, describe them successively. "l" can be specified for binary files. All the operations can be specified for the text file. The operations that can be specified are shown below.

opt	Meaning
b	Blanks and tabs in the head of the field are disregarded.
d	Only the blank and the alphanumeric character are compared.
i	Control character codes are disregarded. Single shifts 2 (SS2) 0x8e and 3 (SS3) 0x8f of EUC are regarded as control characters when ASCII code is specified but regarded as characters when EUC is specified.
j	Lowercase letters are compared as uppercase letters.
l	Compares according to the collating sequence defined in LC_COLLATE environmental variable. Only when the data format is ASCII code, EUC, EUC 2 byte process codes, EUC 4 byte process codes or Unicode and the Input code system (-q) is ASCII code system or Unicode system, this option can be specified. It is in the exclusive relationship with "n".
n	Character strings of the numbers that contains signs are compared with arithmetic value. It can be specified only for text files. The result is not guaranteed when characters other than the numbers exist in the character string. When the data format is ASCII code or Unicode, it can be specified. It is in the exclusive relationship with "l".

If modify collation sequence is specified as the data format of the compared field or comparing field, field operations cannot be specified in *opt*.

cmp

This specifies the comparison operator.
The following shows comparison operators and their meanings.

cmp	Meaning(True condition)
eq	Compared field = Comparing field or literal value
ne	Compared field != Comparing field or literal value
gt	Compared field > Comparing field or literal value
ge	Compared field >= Comparing field or literal value
lt	Compared field < Comparing field or literal value
le	Compared field <= Comparing field or literal value

Note

When the compared field is EUC, EUC 2 byte process codes and EUC 4 byte process codes, the data is compared in the collating sequence according to the environment variable LC_COLLATE.

Example

1. To select records that the field of packed decimal number in 4 bytes from the 32nd byte and the field of the external decimal number in 4 bytes from the 12th byte are equal, specify as follows.

```
-p 11.4zdl.eq.31.4pdl
```

2. To select records that have more than 30 fields of the fixed-point binary number in 4 bytes from the 12th byte, specify as follows.

```
-p 11.4fbi.ge.d30
```

3. To select records that have less than -1000 fields of the external decimal number in 4 bytes from the 10th byte, specify as follows.

```
-p 9.4zdl.lt.d-1000
```

4. To select the record in which the character string 'abcde' exists in the field of ASCII code in 5 bytes from the 10th byte, specify as follows.

   For Bourne shell

```
-p "9.5asc.eq.'abcde'"
```

   For C shell

```
-p 9.5asc.eq.\'abcde\'
```

5. To select the record in which the character string 'ab"cd' exists in the field of ASCII code in 5 bytes from the 10th byte, specify as follows.

   For Bourne shell

```
-p "9.5asc.eq.'ab\"cd'"
```

   For C shell

```
-p 9.5asc.eq.\'ab\"cd\'
```

6. To select the record in which the character string 'abc'd' exists in the field of ASCII code in 5 bytes from the 10th byte, specify as follows.

   For Bourne shell

```
-p "9.5asc.eq.'abc''d'"
```

   For C shell

```
-p 9.5asc.eq.\'abc\'\'d\'
```

7. To select the record in which the character string 'abc' exists in the field of ASCII code in 3 bytes from the 4th byte and character string 'abc' in the field of ASCII code in 3 bytes from the 10th byte, the record in which the character strings 'abcde' in the field of ASCII code in 5 bytes from 4th byte, specify as follows.

   For Bourne shell

```
-p "3.3asc.eq.'abc'","9.3asc.eq.'abc'" -p "3.5asc.eq.'abcde'"
```

For C shell

```
-p 3.3asc.eq.\'abc\',9.3asc.eq.\'abc\' -p 3.5asc.eq.\'abcde\'
```

3.2.15 Character code system conversion option (-Q)

Specify conversion methods between ASCII code and EBCDIC code.

Information

In the following cases, the code conversion between ASCII code and the EBCDIC code is done according to the character code system conversion option (-Q).

- Comparison of key field when data form of key field is EBCDIC code and when Input code system option (-q) is ASCII code system.
- Comparison of key field when data form of key field is ASCII code and when Input code system option (-q) is EBCDIC code system.
- Conversion of character strings of a literal value, when data form of selection field is EBCDIC code.
- Conversion of character strings of a literal value, when data form of reconstruction field is EBCDIC code.

Format

```
-Q altmode
```

altmode

Specify conversion methods between ASCII code and EBCDIC code.
Select and specify one of the following.

altmode	Meaning
0	It executes the code conversion between the EBCDIC code and US ASCII code. **(Default value)**
1	It executes the code conversion between the EBCDIC (Japanese kana) code and ASCII (JIS8) code.
2	It executes the code conversion between the EBCDIC (Lowercase letters) code and ASCII (JIS7) code.

3.2.16 Input code system option (-q)

Specify input code system.

Format

```
-q cdmode
```

cdmode

Specify input code system.
Select and specify one of the following.

cdmode	Meaning
au	ASCII code system (NetCOBOL form) **(Default value)**
ac	ASCII code system (Micro Focus COBOL form)
eb	EBCDIC code system (Note)

cdmode	Meaning
u2	Unicode system (UCS-2 form)
u3	Unicode system (UTF-32 form)
u8	Unicode system (UTF-8 form)

NOTE

The EBCDIC code system can be specified for binary file.

3.2.17 Record skipping option (-R)

This specifies the records not to be processed from the head of the input files in the number of cases.
This is used when data that is not to be processed such as header information exists in the head of the file. When two or more input files exist, specify the number of the case delimiting with comma (,) according to the specified order of the file. By default, all records are to be processed.

Format

```
-R skiprec-no [ ,skiprec-no ...]
```

skiprec-no

This specifies the record that is not to be processed in the number of cases.

Example

1. 10 records from the head of the first input file, 20 records from the head of the 2nd input file are skipped.

   ```
   -R 10,20
   ```

2. 100 records from the head of the 3rd input file are skipped when 3 input files exist.

   ```
   -R ,,100
   ```

3. 100 records from the head of the first input file are skipped when 3 input files exist. The records of the 2nd and 3rd input file are not skipped.

   ```
   -R 100
   ```

3.2.18 Descending order option (-r)

This specifies to sort the data in descending order.
It is an effective option when the specification of the key field is omitted.

Format

```
-r
```

3.2.19 Field separation character hexadecimal option (-S)

This specifies field separation character string by the hexadecimal number for the text file floating field specification.
The field separation character hexadecimal option has an exclusive relationship with the Field separation character option (-t). When either specification of Field separation character hexadecimal option (-S) and Field separation character option (-t) is omitted, tabs or spaces are treated as separation characters of the field.

Format

```
-S fldsep-hex
```

fldsep-hex

The field separation character string is specified by the hexadecimal number.

Note

The field separation character string cannot specify the same character as the Record separation character string (-L).

Example

1. Character string "AB" (ASCII code) is specified by the hexadecimal number for a field separation character string.

   ```
   -S 4142
   ```

2. Character string "<->" (UTF-32LE) is specified by the hexadecimal number for a field separation character string.

   ```
   -S 600000002d00000062000000
   ```

3.2.20 Processing (sort, merge, and copy) option (-s, -m、-c)

It specifies the processes to execute for the input files.

Format

```
[ -s ] | -m | -c
```

The processing options and their meanings are shown below. Each processing option is mutually in the exclusive relationship.

Processing options	Meaning
-s	Sort option **(Default value)**
-m	Merge option
-c	Copy option

3.2.21 Text file option (-T)

This specifies the field type when the input file is a text file.

Format

```
-T { flt | fix | csv | tsv } [ ,{ l | n | N } b d i j ]
```

Information

flt, fix, csv and tsv set the method for specifying the key field, selection field, reconstruction field and summation field. Continuing "l", "n", "N", "b", "d", "i", and "j" are the option arguments that specify the operations of the key fields and effective when the specification of the key field is omitted. To specify two or more operations, describe them successively. When csv and tsv are specified, the key field is not omissible.

flt

This shows the text file floating field specification.
It is a form specified by the number of the field delimited by the field separator string. For information about the field separator string, refer to Field separator character option (-t). It has an exclusive relationship with the fix option argument, the csv option argument and the tsv option argument.

fix

This shows the text file fixed field specification.
It is a form to specify in the byte position from the head. It has an exclusive relationship with the flt option argument, the csv option argument and the tsv option argument.

csv

This shows the text file CSV format specification.
It is specified by the number of the field delimited by the comma character. It has an exclusive relationship with the flt option argument, the fix option argument and the tsv option argument.

tsv

This shows the text file TSV format specification.
It is specified by the number of the field delimited by the tab character. It has an exclusive relationship with the flt option argument, the fix option argument and the csv option argument.

l

Compares according to the collating sequence defined in LC_COLLATE environmental variable.
It is in the exclusive relationship with "n" and "N".

n

Character strings of numbers that contain signs are compared with arithmetic values.
The result is not guaranteed when characters other than the numbers exist in the character string. It is in the exclusive relationship with "l" and "N".

N

Connected data of alphabet and numbers (for instance, data123) are evaluated and sorted separately for the alphabet and the numerical value.
After only alphabets are targeted for the comparison and compared, character strings exclusively with numbers are compared with arithmetic value. Data is evaluated from the left of the specified field, and the data that appeared after effective data is disregarded. It is in the exclusive relationship with "l" and "n".

b

Blanks and tabs in the head of the key field are disregarded.

d

Only the blank and the alphanumeric character are compared.

i

Control character codes are disregarded.
Single shifts 2 (SS2) 0x8e and 3 (SS3) 0x8f of EUC are regarded as control characters when ASCII code is specified but regarded as characters when EUC is specified.

j

Lowercase letters are compared as uppercase letters.

Note

When the input file is a text file, it is necessary to specify Text file option (-T). When Text file option (-T) is omitted, it is treated as a binary file.

3.2.22 Field separation character option (-t)

This specifies field separation character string for the text file floating field specification.
The field separation character option has an exclusive relationship with the Field separation character hexadecimal option (-S). When either specification of Field separation character option (-t) and Field separation character hexadecimal option (-S) is omitted, tabs or spaces are treated as separation characters of the field.

Format

```
-t fldsep
```

fldsep

The field separation character string is specified.
The backslash (\) is specified before the character when the character with a special meaning as a shell is contained in the field separating character string. (example: blank, $, &, ', ", (,), \, |, `, *, <, >, ?)

Note

The field separation character string cannot specify the same character as the Record separation character string (-L).

Example

1. Characters "SEPARATE" are specified as field separation character string.

   ```
   -t SEPARATE
   ```

2. Characters "F S" are specified as field separation character string.

   ```
   -t F\ S
   ```

3. Characters "'" are specified as field separation character string.

   ```
   -t \'
   ```

4. Characters "\"" are specified as field separation character string.

   ```
   -t \"
   ```

5. Characters "\" are specified as field separation character string.

   ```
   -t \\
   ```

6. Characters "**" are specified as field separation character string.

   ```
   -t \*\*
   ```

7. Characters "* *" are specified as field separation character string.

   ```
   -t \*\ \*
   ```

3.2.23 Suppression option (-u)

This specifies the suppression option.
When the values of the specified key fields are equal, 1 record is left and others are deleted. Then the records to be deleted are irregular.
The suppression option is effective in the Sort option (-s) and Merge option (-m). The suppression option has an exclusive relationship with the record summation option (-g). When specified at the same time as the first-in first-out (FIFO) option (-f, BSORT_FIFO of startup file, or environment variable BSORT_FIFO), the specification of the first-in first-out (FIFO) option is disregarded.

Format

```
-u
```

3.2.24 I/O overwrite option (-v)

This specifies to continue processing when one of the input files is the same as the output files.
Because the processing result is overwritten to the input file when I/O overwrite option (-v) is specified, the disk space can be saved. When this option is omitted, and the same file as the output file is specified for one of the input files, it is an error. Only when the Sort option (-s) is specified, it is enabled.

Format

```
-v
```

Note that the data of the input files might not be secured when an error occurs in the sort processing.

3.2.25 Standard output option (-w)

This specifies to output the processing result to a standard output when the output file has not been specified.
When it is specified at the same time as the Output file option (-o), the Output file option (-o) takes priority.

Format

```
-w
```

3.2.26 COBOL file index specification option (-X)

This specifies index fields in the indexed file of COBOL file system.
It cannot be omitted for the indexed file of COBOL file system.

Format

```
-X index-def [ ,index-def ...]
```

index-def

This specifies the main key first, and then the sub-key.

- **Format of *index-def***

```
pos.len typ opt
```

pos

This specifies the position of the index field in a byte position.

len

This specifies the length of the index field in byte.

typ

This specifies the data format of the index field.
The data formats and the lengths of the index field are shown as follows.

Type	Data format		*typ*	Length (byte)	Meaning
Character	ASCII code (Note 1)		asc	1-254	CHARTYPE of NetCOBOL
	Unicode UCS-2 form (Note 2)(Note 3)		uc2	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL
		Big endian	u2b	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL
		Little endian	u2l	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL
	Unicode UTF-32 form (Note 2)(Note 4)		u32	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
		Big endian	u3b	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
		Little endian	u3l	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
	Unicode UTF-8 form (Note 2)		ut8	1-254	CHARTYPE of NetCOBOL

NOTES

1. Can be specified if Input code system (-q) is other than the EBCDIC.
2. Can be specified if Input code system (-q) is other than the EBCDIC, and C or UTF-8 locale is defined in the LANG environment variable.
3. Byte order follows the specification of the environment variable BSORT_UCS2TYPE or the BSORT_UCS2TYPE of startup file.
4. Byte order follows the specification of the environment variable BSORT_UTF32TYPE or the BSORT_UTF32TYPE of startup file.

opt

This specifies the operation of the index field.
The operations that can be specified are shown below. When both "d" and "n" are omitted, it operates assuming that "d" is specified. Always specify "e" for the final segment.

opt	Meaning
a	It sorts data in ascending order.
d	The index can have duplicate key values. It is in an exclusive relation with "n".
e	It shows the end of segments.
n	The index is a unique key. It is in an exclusive relation with "d".

3.2.27 Modify collation sequence option (-x)

Specifies information required to modify the collation sequence.
If col is specified in the key field data format, selection field (compared field or comparison field) data format, with the modify collation

sequence function, comparison is made in the collation sequence specified in the Modify collation sequence option (-x). It is possible to specify information to modify multiple byte collating sequence values by separating the *col-def*s with commas (,).

Format

```
-x col-def [ ,col-def ...]
```

col-def

This specifies information required to modify the collation sequence.
When describing in the format *cctt* and comparing records, if a field contains a byte equivalent to the hexadecimal number *cc*, that byte is evaluated as the hexadecimal number *tt* and compared. A hexadecimal number that is not specified in *cc* is not replaced and is compared as is.

Example

1. To compare the hexadecimal number 0x20 as the hexadecimal number 0x23, specify as follows:

   ```
   -x 2023
   ```

2. If a *col-def* is specified more than once for the same hexadecimal number, the last specification is valid. In the example below, the hexadecimal number 0x20 is compared as the hexadecimal number 0x3A.

   ```
   -x 2023,2040,203A
   ```

3. Specification when figures of ASCII code (0x30-0x39) are compared as a figure of the EBCDIC code (0xF0-0xF9) is as follows:

   ```
   -x 30F0,31F1,32F2,33F3,34F4,35F5,36F6,37F7,38F8,39F9
   ```

3.2.28 Memory size option (-y)

Specifies the size of the work area (memory size) that PowerBSORT uses.
When the Memory size option (-y) is omitted or 0 is specified, if the BSORT_MEMSIZE of startup file is specified, the work area of the size is used and if it has not been specified, the work area is automatically set.

Format

```
-y memsize
```

memsize

This specifies the size of the work area that PowerBSORT uses in kilobytes.
Specify within the range from 64 to 2097151 kilobytes. When the value of 2097152 or more is specified, it is considered that 2097151 were specified.

Note

Memory size option (-y) takes priority over BSORT_MEMSIZE of startup file.

3.2.29 Record format option (-Z)

This specifies the record format for binary files.
It is not possible to specify this option for the text file. It is not necessary to specify this option for the standard file system of the system. It is necessary to specify this option for COBOL file. Refer to PowerBSORT Input Output Environment for the record format that can be specified.

Format

```
-Z recform
```

recform

This specifies the record format.
The record format that can be specified is shown below.

recform	Meaning
f	Fixed-length record format
v	Variable-length record format

3.2.30 Record length option (-z)

This specifies the record length or the maximum record length of the input record.

Format

```
-z recsize
```

recsize

This specifies the record length in byte.
For binary files, the record length is specified when the record format is a fixed-length record form. When the record format is a variable-length record form, the maximum record length is specified. For text files, the maximum record length including the line feed code is specified.

In the binary file (variable-length record form) and the text file, if the record length specified by *recsize* is longer than the actual maximum record length, it is possible to process. However, when the record length that is longer than the actual maximum record length is specified by *recsize*, it becomes the factor of the performance deterioration according to the difference between a specified record length and the actual maximum record length.

Chapter 4 Using the PowerBSORT bsortex command

This section explains the bsortex command.
The bsortex command allows you to execute PowerBSORT using the command line interface. Execution of bsortex can be included in batch files. Return the termination status of 0 when the bsortex command ends normally. Return the termination status other than 0 when the error occurs.

4.1 bsortex command format

The forms of the bsortex command include 3 forms. The options that can be specified differ respectively. Specify an appropriate option according to the option to use.

Note

Notes for the bsortex command description

- The commands with hyphens (-) are called options and the ones specified after options are called operand.
  Example) key operand of -sort option
- Lexical units enclosed with square brackets ([ ]) can be omitted.
- Curly braces ({ }) show selections of lexical units delimited with vertical bars (|).
- For the italics, values are set as required.
- ... shows that it can be specified repeatedly.
- When you specify operands to put an equals sign (=) to the value, specify not to put any blank before and after the equals sign (=).

4.1.1 Using the sort option

The command form for using the sort option is shown below.

```
bsortex  [ -h ]
         [ -a argument-file ]
         [ -define keyword=word-definition [ keyword=word-definition...] [ -define...] ]
         [ -sort [ key=key-definition [ ,key-definition...] [ key=...] ] ]
         [ -record [ recform=record-format ]
             { [ fldsep=field-separate ] | [ fldsepx=field-separate-hex ] } ]
        -input reclen=record-length
             [ file=file-name [ ,file-name...] [ file=...]
                 [ filesys=file-system ] ]
             [ { include=condition-expression [ ,condition-expression ] [ include=...]
                 | omit=condition-expression [ ,condition-expression ] [ omit=...] } ]
             [ reconst=reconst-definition [ ,reconst-definition...] [ reconst=...] ]
             [ eof= { effect | ignore } ]
             [ overwrite ]
         [ -summary { field=summary-definition [ ,summary-definition...] [ field=...]
             | suppress } [ { first | last } ] ]
         [ -output [ file=file-name [ ,file-name...] [ file=...]
                 [ filesys=file-system ] ]
             [ maxfilesize=max-filesize ] [ maxrecnum=max-recordnumber ]
             [ { include=condition-expression [ ,condition-expression ] [ include=...]
                 | omit=condition-expression [ ,condition-expression ] [ omit=...]
                 | case= { condition-expression [ ,condition-expression ]
                     | other } [ case=...] } ]
             [ reconst=reconst-definition [ ,reconst-definition...] [ reconst=...] ]
             [ idxflag=index-flag ]
             [ idxkey=index-key [ ,index-key...] [ idxkey=...] ]
             [ linedlmt=line-delimiter ] [ removeeof ] [ -output...] ]
         [ -option [ colseq=col-def [ ,col-def...] [ colseq=...] ]
```

```
            [ fifo ] [ icode=input-code ] [ iconv=input-conversion ]
            [ memsize=memory-size ]
            [ msgfile=message-file ] [ msglevel=message-level ]
            [ supfile=startup-file ]
            [ tmpdir=temp-directory [ ,temp-directory...] [ tmpdir=...] ] ]
```

4.1.2 Using the merge option

The command form for using the merge option is shown below.

```
bsortex  [ -h ]
         [ -a argument-file ]
         [ -define keyword=word-definition [ keyword=word-definition...] [ -define...] ]
         -merge [ key=key-definition [ ,key-definition...] [ key=...] ]
         [ -record [ recform=record-format ]
             { [ fldsep=field-separate ] | [ fldsepx=field-separate-hex ] } ]
         -input reclen=record-length
             [ file=file-name [ ,file-name...] [ file=...]
                 [ filesys=file-system ] ]
             [ { include=condition-expression [ ,condition-expression ] [ include=...]
                 | omit=condition-expression [ ,condition-expression ] [ omit=...] } ]
             [ reconst=reconst-definition [ ,reconst-definition...] [ reconst=...] ]
             [ eof= { effect | ignore } ]
         [ -summary { field=summary-definition [ ,summary-definition...] [ field=...]
             | suppress } [ { first | last } ] ]
         [ -output [ file=file-name [ ,file-name...] [ file=...]
                 [ filesys=file-system ] ]
             [ maxfilesize=max-filesize ] [ maxrecnum=max-recordnumber ]
             [ { include=condition-expression [ ,condition-expression ] [ include=...]
                 | omit=condition-expression [ ,condition-expression ] [ omit=...]
                 | case= { condition-expression [ ,condition-expression ]
                     | other } [ case=...] } ]
             [ reconst=reconst-definition [ ,reconst-definition...] [ reconst=...] ]
             [ idxflag=index-flag ]
             [ idxkey=index-key [ ,index-key...] [ idxkey=...] ]
             [ linedlmt=line-delimiter ] [ removeeof ] [ -output...] ]
         [ -option [ colseq=col-def [ ,col-def...] [ colseq=...] ]
             [ icode=input-code ] [ iconv=input-conversion ]
             [ memsize=memory-size ]
             [ msgfile=message-file ] [ msglevel=message-level ]
             [ supfile=startup-file ] ]
```

4.1.3 Using the copy option

The command form for using the copy option is shown below.

```
bsortex  [ -h ]
         [ -a argument-file ]
         [ -define keyword=word-definition [ keyword=word-definition...] [ -define...] ]
         -copy
         [ -record [ recform=record-format ]
             { [ fldsep=field-separate ] | [ fldsepx=field-separate-hex ] } ]
         -input reclen=record-length
             [ file=file-name [ ,file-name...] [ file=...]
                 [ filesys=file-system ] ]
             [ { include=condition-expression [ ,condition-expression ] [ include=...]
                 | omit=condition-expression [ ,condition-expression ] [ omit=...] } ]
             [ reconst=reconst-definition [ ,reconst-definition...] [ reconst=...] ]
             [ eof= { effect | ignore } ]
         [ -output [ file=file-name [ ,file-name...] [ file=...]
                 [ filesys=file-system ] ]
             [ maxfilesize=max-filesize ] [ maxrecnum=max-recordnumber ]
```

```
        [ { include=condition-expression [ ,condition-expression ] [ include=...]
            | omit=condition-expression [ ,condition-expression ] [ omit=...]
            | case= { condition-expression [ ,condition-expression ]
                | other } [ case=...] } ]
        [ reconst=reconst-definition [ ,reconst-definition...] [ reconst=...] ]
        [ idxflag=index-flag ]
        [ idxkey=index-key [ ,index-key...] [ idxkey=...] ]
        [ linedlmt=line-delimiter ] [ removeeof ] [ -output...] ]
    [ -option [ colseq=col-def [ ,col-def...] [ colseq=...] ]
        [ icode=input-code ] [ iconv=input-conversion ]
        [ memsize=memory-size ]
        [ msgfile=message-file ] [ msglevel=message-level ]
        [ supfile=startup-file ] ]
```

4.2 bsortex command arguments

The bsortex command accepts a variety of arguments to control the operation of the command. This section explains each argument of the bsortex command.

4.2.1 Argument file option (-a)

This specifies an argument file.
An argument file is a text file in which various option arguments of the bsortex command are defined. An argument file is used to make a separate specification for a particular argument and when a command character string is longer than the maximum length allowed in the operating system.
The argument file must contain only bsortex command option information. Individual options in the argument file can span more than one line, but must not include a line break. Argument file options cannot be specified in the argument file.
If other options are specified along with the argument file option, the option arguments specified in the argument file are interpreted as if they were inserted at the location of the argument file option. You can specify more than one argument file option.

Format

```
-a argument-file
```

argument-file

This specifies the argument file name.
If the argument file name contains spaces, it must be specified in double quotation marks (" ").

Example

1. The following shows an example of an argument file.

```
-sort key=32.12pdla,25.4fbia/n
-input reclen=200/n
       file=sortin/n
       include=25.4fbi.ge.d35/n
-output file=sortout/n (Note)
```

2. The following shows an incorrect example of an argument file.
   It is incorrect because the key operand is split by a line break.

```
-sort key=32.12pdla,/n
          25.4fbia/n
-input reclen=200/n
       file=sortin/n
       include=25.4fbi.ge.d35/n
-output file=sortout/n (Note)
```

NOTE

/n sign indicates the line feed code sequence <LF>.

Information

Assuming that the argument file (argument-file) has the following content:

```
-sort key=0.10asca
```

If

```
bsortex -a argument-file -input reclen=100 file=sortin -output file=sortout
```

is specified, it is the same as specifying

```
bsortex -sort key=0.10asca -input reclen=100 file=sortin -output file=sortout
```

Similarly, if

```
bsortex -sort -a argument-file -input reclen=100 file=sortin -output file=sortout
```

is specified, it is the same as specifying

```
bsortex -sort -sort key=0.10asca -input reclen=100 file=sortin -output file=sortout
```

In the latter case, the effect is that the -sort option is specified twice. As the -sort option can only be specified once, an error occurs.

4.2.2 Copy option (-copy)

This specifies the copy option.
The Copy option (-copy) does not have operand. The Copy option (-copy), Sort option (-sort), and Merge option (-merge) are mutually in the exclusive relationship.

Format

```
-copy
```

4.2.3 Define option (-define)

This defines arbitrary mnemonic names for field definition and file path names.
The arbitrary mnemonic names defined in this option can be used in other options.

Format

```
-define keyword=word-definition [ keyword=word-definition ...][ -define ...]
```

keyword=word-definition

Describe *keyword* in the alphanumeric characters to which @ is put to the first character.
The character string defined in *keyword* is specified for *word-definition*.

Note

- The left-hand part of the operand (for instance, key) that specifies values in option names (for instance, -sort) and equals sign (=) cannot be specified for *word-definition*.
- When two or more mnemonic names are defined, they are replaced in the specified order. Therefore, when you define other mnemonic names including a certain mnemonic name, note that because the result changes by the specified order of the mnemonic names.

  Example 1)

Before the replacement

```
-define @HOK=hoken/hoken1.dat @HOKEN2=hoken/hoken2.dat
-input file=@HOKEN2
```

After the replacement

```
-input file=hoken/hoken1.datEN2
```

Example 2)

Before the replacement

```
-define @HOKEN2=hoken/hoken2.dat @HOK=hoken/hoken1.dat
-input file=@HOKEN2
```

After the replacement

```
-input file=hoken/hoken2.dat
```

Example

```
-define @HOKEN1=hoken/hoken1.dat
        @HOKEN2=hoken/hoken2.dat
        @HOKEN34=hoken/hoken3.dat,hoken/hoken4.dat
        @NAME=12.8asc
        @AGE=20.2fbi
-sort key=@AGEr,@NAMEa
-input file=@HOKEN1,@HOKEN2 include=@AGE.ge.d30
```

4.2.4 Help option (-h)

This lists the command formats of the bsortex command.
After the command syntax is analyzed, the description example is output to the standard error output. When this option is specified, processes in PowerBSORT are not executed. If the command syntax contains an error, the description example is output after the error messages are output.

Format

```
-h
```

4.2.5 Input file information option (-input)

This specifies the input file information option.

In the standard input, omit inputting file operand and filesys operand.

Format

```
-input reclen=record-length
      [ file=file-name [ ,file-name ...] [ file= ...]
          [ filesys=file-system ] ]
      [ { include=condition-expression [ ,condition-expression ] [ include= ...]
          | omit=condition-expression [ ,condition-expression ] [ omit= ...] } ]
      [ reconst=reconst-definition [ ,reconst-definition ...] [ reconst= ...] ]
      [ eof= { effect | ignore } ]
      [ overwrite ]
```

4.2.5.1 reclen operand

This specifies the record length or the maximum record length of the input record.

Format

```
reclen=record-length
```

record-length

This specifies the record length in byte.
For binary files, the record length is specified when the record format is a fixed-length record form. When the record format is a variable-length record form, the maximum record length is specified. For text files, the maximum record length including the line feed code is specified.

Note

In the binary file (variable-length record form) and the text file, if the record length specified by *recsize* is longer than the actual maximum record length, it is possible to process. However, when the record length that is longer than the actual maximum record length is specified by *recsize*, it becomes the factor of the performance deterioration according to the difference between a specified record length and the actual maximum record length.

4.2.5.2 file operand

This specifies the input file.
Two or more input files can be specified by delimiting them with comma (,). However, two or more input files with a different file system cannot be specified. When two or more input files are specified, the record is input from the head of the file in the sort option and the copy option in specification the order. In the merge option, order by which the record is read depends on the value of the key field of each record.

Format

```
file=file-name
```

file-name

This specifies input file path names.

Note

- When you specify the file path names that are operand arguments and contain any blanks, enclose the entire *file-name* with double quotation marks (").
- When you specify the file path names that are operand arguments and contain comma (,), enclose the entire file-name with quotation marks ('), and in addition to that, enclose it with double quotation marks (").
- When you specify the file path names that are operand arguments and contain quotation marks ('), specify it with the successive 2 quotations marks ('), and enclose the entire file-name with double quotation marks (").
- When you specify the file path names that are operand arguments and contain double quotation marks (''), specify a backslash immediately before the double quotation mark (").

Example

1. The example of specifying two or more input files is shown below.

```
file=sortin1,sortin2
```

2. The example of specifying the file path name that contains blank is shown below.

```
file="sortin 01"
```

3. The example of specifying the file path name that contains comma (,) is shown below.

```
file="'sortin,01'"
```

4. The example of specifying the file path name that contains quotation marks (') is shown below.

```
file="sortin''01"
```

5. The example of specifying the file path name that contains double quotation marks (") is shown below.

```
file=sortin\"01
```

4.2.5.3 filesys operand

This specifies file system of input files specified in file operand.

記述形式

```
filesys=file-system
```

file-system

This specifies the file system of the input file.
Specify a file system in an identifier. The file systems that can be specified are shown below. When cob1, cob2, and cob3 are specified for an identifier, it is necessary to define BSORT_FILESYS_fs of a startup file to associate the identifier with an actual file system. NetCOBOL file system can be specified for COBOL file system.

Identifier	Description of file systems
ufs	Native file system of the system **(Default)**
cobs64	NetCOBOL file system (sequential file)
cobp64	NetCOBOL file system (physical sequential file)
cobr64	NetCOBOL file system (relative file)
cobi64	NetCOBOL file system (indexed file)
cob1	COBOL file system
cob2	COBOL file system
cob3	COBOL file system

4.2.5.4 include operand

To use the record selection option for the input file, specify the selection field (selection condition).
The records that meet the condition are to be processed. The include operand is in the exclusive relationship with omit operand.

Format

```
include=condition-expression
```

condition-expression

This specifies the selection field (selection condition).
When you specify two or more *condition-expression* delimiting with comma (,), it creates the logical product of the selection condition.
When two or more include operands are specified, they create the logical sum of the selection condition.

- **Format of *condition-expression***

```
condition-definition [ { .and. | .or. } [ ( ] condition-expression [ ) ] ]
```

and

This shows the logical product of the selection conditions.
It is equal to the values specified by delimiting two or more conditions with comma (,).

or

This shows the logical sum of the selection conditions.
It is equal to the case that two ore more include operands are specified.

condition-definition

One selection field (selection condition) is specified.

- **Format 1 of *condition-definition***

```
pos.len typ [ opt ].cmp.pos.len typ
```

For the format 1 of *condition-definition*, two selection fields are compared. The left side of *cmp* shows the compared field and the right side shows the comparing field. An error occurs when the field to be compared or the comparing field does not exist on the input record.

- **Format 2 of *condition-definition***

```
pos.len typ [ opt ].cmp.self-def
```

For the format 2 of *condition-definition*, the selection field is compared with a literal value. The left side of *cmp* shows compared field and the right side shows a literal value. An error occurs when the field to be compared does not exist on the input record.

- **Format 3 of *condition-definition***

```
RECNUM.cmp.num
```

For the format 3 of *condition-definition*, the record is selected according to the number of records.

pos

This specifies the position of the compared field or the comparing field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

len

This specifies the length of the compared field or the comparing field in the number of byte.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.
To specify an unsigned binary number for the data format, specify mask values. Compare the logical product of the mask value and field values. The mask value specifies the same value as the compared field and the comparing field.
For information about the lengths of each data format, refer to the Data formats that can be specified in the compared field and the comparing field.

In text file CSV format and text file TSV format specification, the length of the compared field or the comparing field does not contain double quotation marks (") that enclose the field. When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

```
Example: Compared field and length in text file CSV format
```

```
Compared field    Characters effective as compared field    Length of compared field
ABC               ABC                                       3bytes
"ABC"             ABC                                       3bytes
"A""B""C"         A"B"C                                     5bytes
"A,B,C"           A,B,C                                     5bytes
```

typ

This specifies the data formats of the compared field and the comparing field.
For more information, refer to the Data formats that can be specified in the compared field and the comparing field.

Note

- When the data format of the compared field and comparing field are different, it compares the data after adjusting them to the data format of the compared field.
- When the data format of the compared field is character, it compares data in the shorter length of the compared field and comparing field.
- When the data format of the compared field is numeric or number, the shorter field of the compared field and comparing field are compared with the longer one after moving to the right end.
- In the binary file processing, when input file code system (icode operand) is excluding EBCDIC, ASCII code can be specified.
- In the text file processing, when input file code system (icode operand) is ASCII, ASCII code can be specified.
- When the input file code system (icode operand) is EBCDIC, EBCDIC code can be specified.
- When the input file code system (icode operand) is ASCII, and C or EUC locale is defined in the LANG environment variable, EUC, EUC 2 byte process codes, and EUC 4 byte process codes can be specified.
- Unicode UCS-2 form can be specified, except when the input file code system (icode operand) is EBCDIC at the binary file. When the input file code system (icode operand) is Unicode (UCS-2 form), the Unicode UCS-2 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UCS-2 form can be specified.
- Unicode UTF-32 form can be specified, except when the input file code system (icode operand) is EBCDIC at the binary file. When the input file code system (icode operand) is Unicode (UTF-32 form), the Unicode UTF-32 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UTF-32 form can be specified.
- Unicode UTF-8 form can be specified, except when the input file code system (icode operand) is EBCDIC at the binary file. When the input file code system (icode operand) is Unicode (UTF-8 form), the Unicode UTF-8 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UTF-8 form can be specified.

opt

This specifies the operations of the compared field.
To specify two or more operations, describe them successively. "l" can be specified for binary files. All the operations can be specified for the text file. The operations that can be specified are shown below.

opt	Meaning
b	Blanks and tabs in the head of the field are disregarded.
d	Only the blank and the alphanumeric character are compared.
i	Control character codes are disregarded. Single shifts 2 (SS2) 0x8e and 3 (SS3) 0x8f of EUC are regarded as control characters when ASCII code is specified but regarded as characters when EUC is specified.
j	Lowercase letters are compared as uppercase letters.
l	Compares according to the collating sequence defined in LC_COLLATE environmental variable. Only when the data format is ASCII code, EUC, EUC 2 byte process codes, EUC

opt	Meaning
	4 byte process codes or Unicode and the input file code system (icode operand) is ASCII code system or Unicode system, this option can be specified. It is in the exclusive relationship with "n".
n	Character strings of the numbers that contains signs are compared with arithmetic value. It can be specified only for text files. The result is not guaranteed when characters other than the numbers exist in the character string. When the data format is ASCII code or Unicode, it can be specified. It is in the exclusive relationship with "l".

- If modify collation sequence is specified as the data format of the compared field or comparing field, field operations cannot be specified in *opt*.
- If character form two digit years, external decimal form two digit years, packed decimal form two digit years or decimal form two digit years is specified as the data format of the compared field or the comparing field, *opt* is ignored.

self-def

This specifies the literal values. The description form of *self-def* is shown below.

- **Format 1 of *self-def***

```
'Character string'
```

Specifies the character string for the format 1 of *self-def*.

- **Format 2 of *self-def***

```
xHexadecimal number
```

Specifies the hexadecimal number for the character string for the format 2 of *self-def*.

- **Format 3 of *self-def***

```
dDecimal number
```

Specifies the decimal number for numeric or number for the format 3 of *self-def*.

- The description method when a literal value is specified is different according to the shell used. The description example in a typical shell is shown below. When a literal value is specified for the argument file (argument file option (-a)), it is same as the method of describing the Bourne shell.
  - Bourne Shell
    - To specify a character string with a operand, enclose the entire operand as shown below.

      Example) When you specify character string "ABC"

      ```
      include="0.3asc.eq.'ABC'"
      ```

    - To specify a character string including a double quotation (") with a operand, specify a backslash (\) immediately before the double quotation (").The character with a special meaning as a shell is similar. (example: $, \, `)

      Example) When you specify character string ""ABC""

      ```
      include="0.5asc.eq.'\"ABC\"'"
      ```

- To insert a quotation (') into a character string, specify two quotations in succession.

  Example) When you specify character string "'ABC'"

```
include="0.5asc.eq.'''ABC'''"
```

- C shell
  - The backslash (\) is specified before quotation (') before and behind the character string when the character string of a literal value is specified.

    Example) When you specify character string "ABC"

```
include=0.3asc.eq.\'ABC\'
```

  - The backslash (\) is specified before double quotation (") when the character string of a literal value which contains double quotation (") is specified. The character with a special meaning as a shell is similar. (example: $, &, (,), |, \, `, {, }, ;, *, <, >, ?, blank)

    Example) When you specify character string ""ABC""

```
include=0.5asc.eq.\'\"ABC\"\'
```

  - When the character string of a literal value which contains quotation (') is specified, it is continued two specification of the backslash (\) before quotation (').

    Example) When you specify character string "'ABC'"

```
include=0.5asc.eq.\'\'\'ABC\'\'\'
```

- When the character string of a literal value is specified, the character string should be one character or more.
- When you specify a decimal number of a literal value, signs can be specified.
- Literal value is compared after adjusting to the data format of the compared field.
- When you specify a text file fixed field, literal values cannot contain record separation character.
- When you specify a text file floating field, literal values cannot contain field separation character or record separation character.
- For text file CSV format and text file TSV format, double quotation marks (") do not need to enclose literal values.
- When specifying literal value of fixed-point binary numbers, unsigned fixed-point binary numbers, fixed-point binary numbers - little endian, unsigned fixed-point binary numbers - little endian, fixed-point binary numbers - system dependent, and unsigned fixed-point binary numbers - system dependent as decimal numbers, a literal value can specify even the value expressible by 16 bytes.

cmp

This specifies the comparison operator.
The following shows comparison operators and their meanings.

cmp	Meaning(True condition)
eq	Compared field = Comparing field or literal value
ne	Compared field != Comparing field or literal value
gt	Compared field > Comparing field or literal value
ge	Compared field >= Comparing field or literal value
lt	Compared field < Comparing field or literal value
le	Compared field <= Comparing field or literal value

Note

When the compared field is EUC, EUC 2 byte process codes and EUC 4 byte process codes, the data is compared in the collating sequence according to the environment variable LC_COLLATE.

RECNUM

This specifies the selection in the number of records.
For the description format 3 of *condition-definition*, specify the fixed string 'RECNUM'.

num

This specifies the number of records.
For input, specify the top record of the input file as the first record. When two or more input files are specified, select the top record of each file as the first record. For output, specify the first record after summation or suppression as the first record.

Example

1. To select records that the field of packed decimal number in 4 bytes from the 32nd byte and the field of the external decimal number in 4 bytes from the 12th byte are equal, specify as follows.

```
include=11.4zdl.eq.31.4pdl
```

2. To select records that have more than 30 fields of the fixed-point binary number in 4 bytes from the 12th byte, specify as follows.

```
include=11.4fbi.ge.d30
```

3. To select records that have less than -1000 fields of the external decimal number in 4 bytes from the 10th byte, specify as follows.

```
include=9.4zdl.eq.d-1000
```

4. When the first 50 records from the head of the file are to be processed, specify as follows.

```
include=RECNUM.le.50
```

5. To select the record in which the character string 'abcde' exists in the field of ASCII code in 5 bytes from the 10th byte, specify as follows.

```
For Bourne shell
```

```
include="9.5asc.eq.'abcde'"
```

```
For C shell
```

```
include=9.5asc.eq.\'abcde\'
```

6. To select the record in which the character string 'ab"cd' exists in the field of ASCII code in 5 bytes from the 10th byte, specify as follows.

```
For Bourne shell
```

```
include="9.5asc.eq.'ab\"cd'"
```

```
For C shell
```

```
include=9.5asc.eq.\'ab\"cd\'
```

7. To select the record in which the character string 'abc'd' exists in the field of ASCII code in 5 bytes from the 10th byte, specify as follows.

```
For Bourne shell
```

```
include="9.5asc.eq.'abc''d'"
```

For C shell

```
include=9.5asc.eq.\'abc\'\'d\'
```

8. To select the record in which the character string 'abc' exists in the field of ASCII code in 3 bytes from the 4th byte and character string 'abc' in the field of ASCII code in 3 bytes from the 10th byte, the record in which the character strings 'abcd' in the field of ASCII code in 5 bytes from 4th byte, specify as follows.

   For Bourne shell

```
include="3.3asc.eq.'abc'.and.9.3asc.eq.'abc'.or.3.5asc.eq.'abcde'"
```

   For C shell

```
include=3.3asc.eq.\'abc\'.and.9.3asc.eq.\'abc\'.or.3.5asc.eq.\'abcde\'
```

4.2.5.5 omit operand

To use the record selection option for the input file, specify the selection field (selection condition).
This excludes the records that meet the condition from the object of processing. The omit operand is in the exclusive relationship with include operand.

Format

```
omit=condition-expression
```

condition-expression

This specifies the selection field (selection condition).
For information about the *condition-expression*, refer to include operand of input file information option (-input).

4.2.5.6 reconst operand

To use the record reconstruction option for the input file, specify the reconstruction field.
The field specified for the reconstruction field is output from right after the left of the output record sequentially. When you specified the record reconstruction option for the input file, the field position after the reconstruction of the input field in the key field, summation field, selection field of the output and reconstruction field of the output are specified.

Format

```
reconst=reconst-definition
```

reconst-definition

This specifies the reconstruction field.
Two or more *reconst-definition* can be specified by delimiting the data by comma (,).

- **Format 1 of *reconst-definition***

```
pos.len
```

  Format 1 specifies a field in the input record.
  When the specified field does not exist on the record of the input, it is an error.

- **Format 2 of *reconst-definition***

```
self-def.len typ [ opt ]
```

  Format 2 specifies literal values.

- **Format 3 of *reconst-definition***

```
pos.END
```

Format 3 specifies from the position of the input record specification to the end of the reconstruction field.
If the specified field does not exist in the input record, an error occurs.

- **Format 4 of *reconst-definition***

```
EMPTY [ opt ]
```

Format 4 specifies an empty field.
Format 4 can be specified for the text file CSV format and the text file TSV format.

pos

This specifies positions of the reconstruction field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

len

This specifies a length of the reconstruction field in the number of bytes.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.
In the specification of description format 1, the length is not limited. For the information about each data format that can be specified in the description format 2, refer to Data formats that can be specified in the literal value of the reconstruction field.

Note

In the text file CSV format and text file TSV format specification, the length of the reconstruction field does not contain double quotation marks (“). When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

Example: Reconstruction field and length in text file CSV format.

Reconstruction field	Characters effective as reconstruction field	Length of reconstruction field
ABC	ABC	3bytes
"ABC"	ABC	3bytes
"A""B""C"	A"B"C	5byte
"A,B,C"	A,B,C	5byte

self-def

This specifies literal values. The description form of *self-def* is shown below.

- **Format 1 of *self-def***

```
'Character string'
```

- **Format 2 of *self-def***

```
xHexadecimal number
```

- **Format 3 of *self-def***

```
dDecimal number
```

Note

- The description method when a literal value is specified is different according to the shell used. The description example in a typical shell is shown below. When a literal value is specified for the argument file (argument file option (-a)), it is same as the method of describing the Bourne shell.
  - Bourne Shell
    - To specify a character string with a operand, enclose the entire operand as shown below.

      Example) When you specify character string "ABC"

      ```
      reconst="'ABC'.3asc"
      ```

    - To specify a character string including a double quotation with a operand, specify a backslash (\) immediately before the double quotation (").The character with a special meaning as a shell is similar. (example: $, \, `)

      Example) When you specify character string ""ABC""

      ```
      reconst="'\"ABC\"'.5asc"
      ```

    - To insert a quotation (') into a character string, specify two quotations in succession.

      Example) When you specify character string "'ABC'"

      ```
      reconst="''''ABC''''.5asc"
      ```

  - C shell
    - The backslash(\) is specified before quotation (') before and behind the character string when the character string of a literal value is specified.

      Example) When you specify character string "ABC"

      ```
      reconst=\'ABC\'.3asc
      ```

    - The backslash (\) is specified before double quotation (") when the character string of a literal value which contains double quotation (") is specified. The character with a special meaning as a shell is similar. (example: $, &, (,), |, \, `, {, }, ;, *, <, >, ?, blank)

      Example) When you specify character string ""ABC""

      ```
      reconst=\'\"ABC\"\'.5asc
      ```

    - When the character string of a literal value which contains quotation (') is specified, it is continued two specification of the backslash (\) before quotation (').

      Example) When you specify character string "'ABC'"

      ```
      reconst=\'\'\'ABC\'\'\'.5asc
      ```

- When the character string of a literal value is specified, the character string should be one character or more.
- When you specify a decimal number of a literal value, signs can be specified.
- When you specify a text file fixed field, literal values cannot contain record separation character.
- When you specify a text file floating field, literal values cannot contain field separation character or record separation character.
- Output text file CSV format and text file TSV format with literal values enclosed in double quotation marks (") when the literal values contains the field separation character, the record separation character (Note) or double quotation marks ("). In this case, a literal value double quotation mark (") is output when two double quotation marks (") are consecutive.

  Note: Enclose literal values with double quotation marks (") when it is included by not only the record separator of the input file but also one of CRLF, CR, and LF permitted as a record separator.

```
Example: The data output to the reconstruction field when literal values contain the field
separation character (comma) or the double quotation marks (") is as follows:
```

```
Specification of literal values    Data output to reconstruction field
FIELD"2"                           "FIELD""2"""
FIELD2,3                           "FIELD2,3"
FIELD"2",3                         "FIELD""2"",3"
```

- When a value specified with a literal value and the length specified with *len* are different, process them as shown below.
  - When the literal value is character strings, the literal value is specified to start at the left and any blank spaces to the right is filled with the appropriate number of spaces if the length of the character strings specified in the literal value is shorter than the length specified in *len*. If the length of the character strings specified in the literal value is longer than the length specified in *len*, it is an error.
  - If the literal value is number, convert the value specified in the literal value into the data format and the length specified in *typ* and *len*, and embed them. If the conversion result is more than len, it is an error.
- When specifying literal value of fixed-point binary numbers, unsigned fixed-point binary numbers, fixed-point binary numbers - little endian, unsigned fixed-point binary numbers - little endian, fixed-point binary numbers - system dependent, and unsigned fixed-point binary numbers - system dependent as decimal numbers, a literal value can specify even the value expressible by the length specified with *len*.

Example

Length	Signed Data Format	Unsigned Data Format
1 byte	-128 - 127	0 - 255
2 byte	-32768 - 32767	0 - 65535
3 byte	-8388608 - 8388607	0 - 16777215
4 byte	-2147483648 - 2147483647	0 - 4294967295
:	:	:

typ

This specifies the data format of literal values.
For the information about each data format that can be specified, refer to Data formats that can be specified in the literal value of the reconstruction field.

Note

- In the binary file processing, when input file code system (icode operand) is excluding EBCDIC, ASCII code can be specified.
- In the text file processing, when input file code system (icode operand) is ASCII, ASCII code can be specified.
- When the input file code system (icode operand) is EBCDIC, EBCDIC code can be specified.
- When the input file code system (icode operand) is ASCII, and C or EUC locale is defined in the LANG environment variable, EUC can be specified.
- Unicode UCS-2 form can be specified, except when the input file code system (icode operand) is EBCDIC at the binary file. When the input file code system (icode operand) is Unicode (UCS-2 form), the Unicode UCS-2 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UCS-2 form can be specified.
- Unicode UTF-32 form can be specified, except when the input file code system (icode operand) is EBCDIC at the binary file. When the input file code system (icode operand) is Unicode (UTF-32 form), the Unicode UTF-32 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UTF-32 form can be specified.

- Unicode UTF-8 form can be specified, except when the input file code system (icode operand) is EBCDIC at the binary file. When the input file code system (icode operand) is Unicode (UTF-8 form), the Unicode UTF-8 form can be specified in case of the text file. In both cases, when C or UTF-8 locale is defined in the LANG environment variable, the Unicode UTF-8 form can be specified.

END

Specify the fixed string 'END' at format 3 of *reconst-definition*.

EMPTY

Specify the fixed string 'EMPTY' at format 4 of *reconst-definition*.

opt

This specifies the operation of the reconstruction field.
It can be specified in text file CSV format or text file TSV format, at format 2 of *reconst-definition* or format 4 of *reconst-definition*. When this specification is omitted, it operates assuming that L is specified.

opt	Meaning
A	Enclose the reconstruction field with double quotation marks (").
L	Do not enclose the reconstruction field with double quotation marks ("). However, enclose it with double quotation marks (") when a field separation character, a record separation character (Note) or double quotation marks (") is included in the reconstruction field.

NOTE

Enclose literal values with double quotation marks (") when it is included by not only the record separator of the input file but also one of CRLF, CR, and LF permitted as a record separator.

Information

- In text file CSV format and text file TSV format, when format 1 of *reconst-definition* or format 3 of *reconst-definition* is specified, enclosing the reconstruction field with double quotation marks (") depends on the condition. When the field on the input record is enclosed with double quotation marks ("), the field after the record is reconstructed is enclosed with double quotation marks (").

  Example: Specify "reconst=1.3,2.2" for the reconstruction field.

```
Input record                Output record
"001","ABC",60              "ABC",60
"002","ABCDE",50            "ABC",50
"003","AB,CDE",40           "AB,",40
"004","AB"CDE,30 (Note)     "AB"C,30

NOTE
"AB" is enclosed by double quotation marks (") in the second field and "CDE" continues afterwards.
```

Example

1. To reconstruct a 10 byte field from the 5th byte of the input record, specify as follows.

```
reconst=4.10
```

2. To specify the character strings of literal value (abc), specify as follows.

   For Bourne shell

```
reconst="'abc'.3asc"
```

For C shell

```
reconst=\'abc\'.3asc
```

3. To specify the character strings of literal value (ab"cd) that contain double quotation mark ("), specify as follows.

For Bourne shell

```
reconst="'ab\"cd'.5asc"
```

For C shell

```
reconst=\'ab\"cd\'.5asc
```

4. To specify the character strings of a literal value (abc'd) that contain quotation mark ('), specify as follows.

For Bourne shell

```
reconst="'abc''d'.5asc"
```

For C shell

```
reconst=\'abc\'\'d\'.5asc
```

5. To specify hexadecimal number of a literal value (abc), specify as follows.

```
reconst=x616263.3asc
```

6. To specify external decimal number of a literal value (-32) by 4 bytes, specify as follows.

```
reconst=d-32.4zdl
```

7. To reconstruct the field from the 5th byte of the input record to the end of the record, specify as follows.

```
reconst=4.END
```

8. To reconstruct a 8 byte field from the 3rd byte of the input record, and character strings of a literal value (,), and 5 byte field from the 20th byte of the input record, specify as follows.

For Bourne shell

```
reconst="2.8,',.1asc,19.5"
```

For C shell

```
reconst=2.8,\',\'.1asc,19.5
```

9. Enclose with double quotation marks ("), and specify the field in the text file CSV format and text file TSV format to specify the character string of literal value (abc).

For Bourne shell

```
reconst="'abc'.3ascA"
```

For C shell

```
reconst=\'abc\'.3ascA
```

10. In text file CSV format and text file TSV format, specify the empty field enclosed with double quotation marks (") between the first field in the input record and the second field.

```
reconst=0.1,EMPTYA,1.END
```

4.2.5.7 eof operand

When the input file is a text file, this specifies how the EOF control character is handled.

Format

```
eof={ effect | ignore }
```

effect

This inputs data up to the first EOF control character, recognizing the first EOF control character as the end of the file.

ignore

This continues to input data even after the EOF control character (that is, the EOF control character is not recognized as the end of the file).

Note

The default value of the eof operand is **ignore**.

4.2.5.8 overwrite operand

This specifies to continue processing when one of the input files is the same as the output files.
Because the processing result is overwritten to the input file when overwrite operand is specified, the disk space can be saved. When this this operand is omitted, and the same file as the output file is specified for one of the input files, it is an error. Only when the sort option is specified, it is enabled.

Format

```
overwrite
```

Note

Note that the data of the input files might not be secured when an error occurs in the sort processing.

4.2.6 Merge option (-merge)

This specifies the merge option.
The Merge option (-merge), Sort option (-sort), and Copy option (-copy) are mutually in the exclusive relationship.

Format

```
-merge [ key=key-definition [ ,key-definition ...] [ key= ...] ]
```

4.2.6.1 key operand

This specifies the key fields.
A key field is composed of position, length, data format, and operation. When there are two or more key fields, they are compared in the specified order. When two or more key fields are specified, the key field is delimited by the comma (,). Moreover, two or more key operands

can be specified. If the specification of key fields is omitted, the whole records is considered to be a key field and sorted in ascending order of the code.

The key field cannot be omitted for text file CSV format and text file TSV format.

Format

```
key=key-definition [ ,key-definition ...] [ key= ...]
```

key-definition

This specifies the key field.

- **Format 1 of *key-definition***

  ```
  pos.len typ [ opt ]
  ```

  In description format 1 of *key-definition*, the field in the input record is specified for a key field.

- **Format 2 of *key-definition***

  ```
  ALL [ opt ]
  ```

  In description format 2 of *key-definition*, the entire record is specified for a key field.
  It is not possible to specify it for text file CSV format and the text file TSV format.

pos

This specifies the position of the key field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

len

This specifies the length of the key field in the number of bytes.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.
To specify an unsigned binary number for *typ*, specify mask value for *len* in decimal number. The logical products of the field value and mask value are considered to be the key value.

The length of the key field does not contain double quotation marks (") that enclose the field for text file CSV format and text file TSV format. When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

Example: Key field and length in text file CSV format.

```
Key field        Characters effective as key field        Length of key field
ABC              ABC                                      3bytes
"ABC"            ABC                                      3bytes
"A""B""C"        A"B"C                                    5bytes
"A,B,C"          A,B,C                                    5bytes
```

typ

This specifies the data format of the key field.
For more information, refer to the Data formats that can be specified in the key field.

opt

This specifies operation of the key field.
To specify two or more operations, describe them successively. "a", "l", and "r" can be specified for binary files. All the operations can be specified for the text file. When both "a" and "r" are omitted, it operates assuming that "a" is specified. The operations that can be specified are shown below.

opt	Meaning
a	It sorts data in ascending order. It is in the exclusive relationship with "r".
b	Blanks and tabs in the head of the key field are disregarded.
d	Only the blank and the alphanumeric character are compared.
i	Control character codes are disregarded. Single shifts 2 (SS2) 0x8e and 3 (SS3) 0x8f of EUC are regarded as control characters when ASCII code is specified but regarded as characters when EUC is specified.
j	Lowercase letters are compared as uppercase letters.
l	Compares according to the collating sequence defined in LC_COLLATE environmental variable. Only when the data format is ASCII code, EUC, EUC 2 byte process codes, EUC 4 byte process codes or Unicode and the input file code system (icode operand) is ASCII code system or Unicode system, this option can be specified. It is in the exclusive relationship with "n" and "N".
n	Character strings of the numbers that contains signs are compared with arithmetic value. It can be specified only for text files. The result is not guaranteed when characters other than the numbers exist in the character string. When the data format is ASCII code, EBCDIC code or Unicode, it can be specified. It is in the exclusive relationship with "l" and "N".
N	Connected data of alphabet and numbers (for instance, data123) are evaluated and sorted separately for the alphabet and the numerical value. After only alphabets are targeted for the comparison and compared, character strings exclusively with numbers are compared with arithmetic value. Data is evaluated from the left of the specified field, and the data that appeared after effective data is disregarded. It can be specified when the data format is ASCII code, EBCDIC code or Unicode. It is in the exclusive relationship with "l" and "n".
r	It sorts the data in descending order. It is in the exclusive relationship with "a".

- When modify collation sequence is specified for a data form of the key field, the operations other than "a" or "r" cannot be specified.
- When character form two digit years, packed decimal form two digit years, external decimal form two digit years and decimal form two digit years are specified for a data form of the key field, the operations other than "a" or "r" are disregarded.

ALL

A fixed character string of 'ALL' is specified for description format 2 of *key-definition*.
It is used when the key field is omitted and the entire record is assumed to be a key field. If the operation of the key field is not specified, the key operand can be omitted.

Example

1. To specify to sort the fields of ASCII code in 10 bytes length from the head of the record in ascending order:

```
key=0.10asca
```

2. To specify an external decimal number in 8 bytes length from the 5th byte of the input record, arranged in descending order:

```
key=4.8zdlr
```

3. To specify ASCII code in 20 bytes length, arranged from the 10th byte of the input record in ascending order at the text file fixed field specification:

```
key=9.20ascabj
```

   The first blank and tab of the key field are disregarded, and lower case letters are treated as upper case letters.

4. To arrange in descending order as a key of the entire record:

```
key=ALLr
```

Information

For the binary file variable-length record form or the text file, when the record where the key field does not exist is input, the value of the part where the key field does not exist is processed as 0.

Example 1) binary file variable-length record form or the text file fixed field:

```
key field:6.4asca

  0123456789 : Record where key field exists
  012345     : Record where key field doesn't exist
```

Example 2) the text file floating field, the text file CSV format or the text file TSV format:

```
key field:2.1asca

  fld0,fld1,fld2,fld3 : Record where key field exists
  fld0,fld1           : Record where key field doesn't exist
  fld0,fld1,,fld3     : Record where key field doesn't exist
```

4.2.7 Execution environment option (-option)

This option specifies each execution environment option.

Format

```
-option [ colseq=col-def [ ,col-def ...] [ colseq= ...] ]
        [ fifo ]
        [ icode=input-code ]
        [ iconv=input-conversion ]
        [ memsize=memory-size ]
        [ msgfile=message-file ]
        [ msglevel=message-level ]
        [ supfile=startup-file ]
        [ tmpdir=temp-directory [ ,temp-directory ...] [ tmpdir= ...] ]
```

4.2.7.1 colseq operand

Specifies information required to modify the collation sequence.
If col is specified in the key field data format, selection field (compared field or comparison field) data format, with the modify collation sequence function, comparison is made in the collation sequence specified in the colseq operand. It is possible to specify information to modify multiple byte collating sequence values by separating the *col-def*s with commas (,).

Format

```
colseq=col-def [ ,col-def ...] [ colseq= ...]
```

col-def

This specifies information required to modify the collation sequence.
When describing in the format *cctt* and comparing records, if a field contains a byte equivalent to the hexadecimal number *cc*, that byte is evaluated as the hexadecimal number *tt* and compared. A hexadecimal number that is not specified in *cc* is not replaced and is compared as is.

Example

1. To compare the hexadecimal number 0x20 as the hexadecimal number 0x23, specify as follows:

   ```
   colseq=2023
   ```

2. If a *col-def* is specified more than once for the same hexadecimal number, the last specification is valid. In the example below, the hexadecimal number 0x20 is compared as the hexadecimal number 0x3A.

   ```
   colseq=2023,2040,203A
   ```

3. Specification when figures of ASCII code (0x30-0x39) are compared as a figure of the EBCDIC code (0xF0-0xF9) is as follows:

   ```
   colseq=30F0,31F1,32F2,33F3,34F4,35F5,36F6,37F7,38F8,39F9
   ```

4.2.7.2 fifo operand

This specifies the first-in first-out (FIFO) option.
When the vales of the specified key fields are equal, the firstly input records are output first. This option is effective in the Sort option (-sort). When omitted, the output order is not necessarily first-in first-out. When specified at the same time as the Merge option (-merge), the Copy option (-copy), the Summation option (field operand), and the Suppression option (suppress operand), the specification of the first-in first-out (FIFO) option is disregarded.

Format

```
fifo
```

Note

A priority level specified of the FIFO option is as follows.

1. Environment variable BSORT_FIFO
2. fifo operand of -option option
3. BSORT_FIFO of startup file

4.2.7.3 icode operand

This specifies input code system options.

Format

```
icode=input-code
```

input-code

Specify input code system.
Select and specify one of the following.

input-code	Meaning
au	ASCII code system (NetCOBOL form) **(Default value)**
ac	ASCII code system (Micro Focus COBOL form)
eb	EBCDIC code system (Note)
u2	Unicode system (UCS-2 form)
u3	Unicode system (UTF-32 form)
u8	Unicode system (UTF-8 form)

NOTE

The EBCDIC code system can be specified for binary file.

4.2.7.4 iconv operand

This specifies conversion methods between ASCII code and EBCDIC code.

Information

In the following cases, the code conversion between ASCII code and the EBCDIC code is done according to the iconv operand.

- Comparison of key field when data form of key field is EBCDIC code and when input file code system (icode operand) is ASCII code system.
- Comparison of key field when data form of key field is ASCII code and when input file code system (icode operand) is EBCDIC code system.
- Conversion of character strings of a literal value, when data form of selection field is EBCDIC code.
- Conversion of character strings of a literal value, when data form of reconstruction field is EBCDIC code.

Format

```
iconv=input-conversion
```

input-conversion

Specify conversion methods between ASCII code and EBCDIC code.
Select and specify one of the following.

input-conversion	Meaning
0	It executes the code conversion between the EBCDIC code and US ASCII code. **(Default value)**
1	It executes the code conversion between the EBCDIC (Japanese kana) code and ASCII (JIS8) code.
2	It executes the code conversion between the EBCDIC (Lowercase letters) code and ASCII (JIS7) code.

4.2.7.5 memsize operand

Specifies the size of the work area (memory size) that PowerBSORT uses.
When the memsize operand is omitted or 0 is specified, if the BSORT_MEMSIZE of startup file is specified, the work area of the size is used and if it has not been specified, the work area is automatically set.

Format

```
memsize=memory-size
```

memory-size

This specifies the size of the work area that PowerBSORT uses in kilobytes.
Specify within the range from 64 to 2097151 kilobytes. When the value of 2097152 or more is specified, it is considered that 2097151 were specified.

The memsize operand takes priority over BSORT_MEMSIZE of startup file.

4.2.7.6 msgfile operand

This specifies file path name to output the messages.
By default, if the BSORT_MSGFILE of startup file is specified, it is output to the file. If it is not specified, output the message to the standard error output.

Format

```
msgfile=message-file
```

message-file

This specifies file path name to output the messages.
To specify a file name that contains a space, enclose the whole name with double quotation marks (").

- The msgfile operand takes priority over BSORT_MSGFILE of startup file.
- When the message file is specified, the information message is output to the message file. The warning message and the error message are output to the message file and the standard error output.

4.2.7.7 msglevel operand

Specify messages to be output.
When msglevel operand is omitted, BSORT_MSGLEVEL of startup file is used. When msglevel operand and the BSORT_MSGLEVEL of startup file specification are omitted, W (error message and warning message are output) is assumed for *message-level*. The output destination of the message is decided by specifying msgfile operand or the BSORT_MSGFILE of startup file.

Format

```
msglevel=message-level
```

message-level

The type of the output message is specified. The message level that can be specified are shown below.

message-level	Meaning
N	Nothing is output.
E	The error message is output.
W	The error message and the warning message are output. **(Default value)**
I	The error message, the warning message, and the information message are output.

The type of the output message used in previous versions can also be specified.

message-level	Meaning
0	Nothing is output.
1	The error message and the warning message are output. **(Default value)**
2	The error message, the warning message, and the information message are output.

- The errors of the command syntax and command help are output to the standard error output even when specifying that nothing is output.
- The msglevel operand takes priority over BSORT_MSGLEVEL of startup file.

4.2.7.8 supfile operand

This specifies file path name of startup file.
By default, use "$HOME/.bsortrc" as a startup file. "$HOME" means PowerBSORT user's home directory.

Format

```
supfile=startup-file
```

startup-file

This specifies file path name of startup file.
To specify a file name that contains any blanks, enclose the whole name with double quotation marks (").

4.2.7.9 tmpdir operand

This specifies directory names of the temporary files.
Two or more directory names can be specified by delimiting with comma (,). Moreover, two or more tmpdir operands can be specified.
When two or more directories are specified, the temporary file is distributed to the specified directory.

Format

```
tmpdir=temp-directory [ ,temp-directory ...] [ tmpdir= ...]
```

temp-directory

This specifies directory names of the temporary file.
When the directory names contain any blanks, enclose it with double quotation marks (").

- Refer to "Priority level of the directory that creates the temporary file" for the priority level of the directory specification that creates the temporary file.

- When a wrong directory name is specified, the file not intended might be deleted.

4.2.8 Output file information option (-output)

This specifies output file information.
Two or more output file information options (-output) can be specified. When a standard output is specified for the output file, the file operand and the filesys operand are omitted. In this case, two or more output file information options (-output) cannot be specified.

Format

```
-output [ file=file-name [ ,file-name ...] [ file= ...]
            [ filesys=file-system ] ]
        [ maxfilesize=max-filesize ]
        [ maxrecnum=max-recordnumber ]
        [ { include=condition-expression [ ,condition-expression ] [ include= ...]
            | omit=condition-expression [ ,condition-expression ] [ omit= ...]
            | case= { condition-expression [ ,condition-expression ] | other } [ case= ...] } ]
        [ reconst=reconst-definition [ ,reconst-definition ...] [ reconst= ...] ]
        [ idxflag=index-flag ]
        [ idxkey=index-key [ ,index-key ...] [ idxkey= ...] ]
        [ linedlmt=line-delimiter ]
        [ removeeof ]
        [ -output ...]
```

4.2.8.1 file operand

This specifies the output file.
Two or more output files can be specified by delimiting with comma (,). When two or more output files are specified and maxfilesize operand or maxrecnum operand is specified, the data is output after being divided into two or more files according to the specification of maxfilesize operand or maxrecnum operand. When two or more output files are specified and maxfilesize operand or maxrecnum operand is not specified, the data is output after being divided into two or more files as recovery processing for the file system's blowout. The recovery processing can be specified at the following files.

- System standard file (binary file or text file)
- NetCOBOL file uses the high speed access library for a NetCOBOL sequential file (libbscblfast64.so)
- NetCOBOL file uses the high speed access library for a NetCOBOL physical sequential file (libbscblpsfast64.so)

- When the record that cannot be output even if all specified output files are used exists, it is an error.
- When the number of specified output file is 1, maxfilesize operand and maxrecnum operand cannot be specified.

Format

```
file=file-name
```

file-name

This specifies output file path names.
For notes on describing other *file-name*, refer to file operand of the input file information option (-input).

4.2.8.2 filesys operand

This specifies the file system of the output file specified in file operand.

Format

```
filesys=file-system
```

file-system

This specifies the file system of the output file.
When omitted, the standard file system of the system is assumed to be specified. For information about the file systems that can be specified, refer tofilesys operand of input file information option (-input).

4.2.8.3 maxfilesize operand

This specifies the maximam file size that can be output to a file.

Format

```
maxfilesize=max-filesize [ K | M | G ]
```

max-filesize [K | M | G]

This specifies the maximum file size that can be output to a file.
When the size of the file exceeds *max-filesize*, the data is continuously output to the next output file that is specified in file operand. K, M, and G are units of the maximum file size. K shows unit of kilobyte, M shows Mega-Byte and G shows Giga-Byte. When neither K, M nor G is specified, the unit is byte. When the data is specified at the same time as maxrecnum operand, it is output to the next output file when either one first reaches to the limit. Also, when the file system abnormally terminates before reaching to the limit specified, the data is output to the next output file at the point the file system abnormally terminates. The maxfilesize operand can be specified at the following files.

- System standard file (binary file or text file)
- NetCOBOL file uses the high speed access library for a NetCOBOL sequential file (libbscblfast64.so)
- NetCOBOL file uses the high speed access library for a NetCOBOL physical sequential file (libbscblpsfast64.so)

When the number of the output file specified in file operand is 1, maxfilesize operand cannot be specified.

4.2.8.4 maxrecnum operand

This specifies the maximum number of records that can be output to a file.

Format

```
maxrecnum=max-recordnumber
```

max-recordnumber

This specifies the maximum number of records that can be output to a file.
When the number of records exceeds *max-recordnumber*, it is continuously output to the next output file specified in file operand. When the data is specified at the same time as maxfilesize operand, it is output to the next output file when either one first reaches to the limit. Also, when the file system abnormally terminates before reaching to the limit specified, the data is output to the next output file at the point the file system abnormally terminates.

When the number of the output file specified in file operand is 1, maxrecnum operand cannot be specified.

4.2.8.5 include operand

To use the record selection option for the output file, specify the selection field (selection condition).
The records that meet the condition are to be processed. The include operand is in the exclusive relationship with omit operand and case operand.

Format

```
include=condition-expression
```

condition-expression

This specifies the selection field (selection condition).
When you specify two or more *condition-expression* delimiting with comma (,), it makes the logical product of the selection conditions.
When two or more include operands are specified, it makes logical sum of the selection condition. For information about the *condition-expression*, refer to include operand of input file information option (-input).

4.2.8.6 omit operand

To use the record selection option for the output file, specify the selection field (selection condition).
This excludes the records that meet the condition from the object of processing. The omit operand is in the exclusive relationship with include operand and case operand.

Format

```
omit=condition-expression
```

condition-expression

This specifies the selection field (selection condition).
When you specify two or more *condition-expression* delimiting with comma (,), it makes the logical product of the selection conditions.
When two or more omit operands are specified, it makes logical sum of the selection condition. For information about the *condition-expression*, refer to include operand of input file information option (-input).

4.2.8.7 case operand

To use the record selection option for the output file, specify the selection field (selection condition).
The records that meet the condition are to be processed. The case operand is in the exclusive relationship with include operand and omit operand.

Format

```
case={ condition-expression | other }
```

condition-expression

This specifies the selection field (selection condition). However, the records that are output in case operand of the output file information option (-output) specified before this description are not to be output.
When you specify two or more *condition-expression* delimiting with comma (,), it makes the logical product of the selection conditions.
When two or more case operands are specified, it makes logical sum of the selection condition. For information about the *condition-expression*, refer to include operand of input file information option (-input).

other

Specify the fixed string 'other'.
The records that are not output in case operand of the output file information option (-output) specified are to be output.

Example

The specification example of distributing the record according to the condition is as follows.

- The selection field outputs the record of less than 100 to sortout1.
- The selection field outputs the record from 100 to less than 200 to sortout2.
- The selection field outputs the record from 200 to less than 500 to sortout3.
- The selection field outputs the record of 500 or more to sortout4.

```
-output file=sortout1 case=49.4zdl.lt.d100
-output file=sortout2 case=49.4zdl.lt.d200
-output file=sortout3 case=49.4zdl.lt.d500
-output file=sortout4 case=other
```

4.2.8.8 reconst operand

To use the record reconstruction option for the output file, specify the reconstruction field.

Format

```
reconst=reconst-definition
```

reconst-definition

This specifies the reconstruction field.
Two or more *reconst-definition* can be specified by delimiting the data by comma (,). For information about the *reconst-definition*, refer to reconst operand of the input file information option (-input).

4.2.8.9 idxflag operand

This specifies the method for creating the index in the indexed file of COBOL file system.

Format

```
idxflag=index-flag
```

index-flag

This selects and specifies from the following. When two or more are specified, specify options continuously.

index-flag	Meaning
c	The index is a compressed key.
r	The record is a compressed data.

4.2.8.10 idxkey operand

This specifies index fields in the indexed file of COBOL file system.
It cannot be omitted for the indexed file of COBOL file system.

Format

```
idxkey=index-key [ ,index-key ...]
```

index-key

This specifies the main key first, and then the sub-key.

- **Format of *index-key***

```
pos.len typ opt
```

pos

This specifies the position of the index field in a byte position.

len

This specifies the length of the index field in byte.

typ

This specifies the data format of the index field.
The data formats and the lengths of the index field are shown as follows.

Type	Data format	*typ*	Length (byte)	Meaning
Character	ASCII code (Note 1)	asc	1-254	CHARTYPE of NetCOBOL
	Unicode UCS-2 form (Note 2)(Note 3)	uc2	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL
	Big endian	u2b	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL
	Little endian	u2l	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL
	Unicode UTF-32 form (Note 2)(Note 4)	u32	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
	Big endian	u3b	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
	Little endian	u3l	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
	Unicode UTF-8 form (Note 2)	ut8	1-254	CHARTYPE of NetCOBOL

NOTES

1. Can be specified if the input file code system (icode operand) is other than the EBCDIC.
2. Can be specified if the input file code system (icode operand) is other than the EBCDIC, and C or UTF-8 locale is defined in the LANG environment variable.
3. Byte order follows the specification of the environment variable BSORT_UCS2TYPE or the BSORT_UCS2TYPE of startup file.
4. Byte order follows the specification of the environment variable BSORT_UTF32TYPE or the BSORT_UTF32TYPE of startup file.

opt

This specifies the operation of the index field.
The operations that can be specified are shown below. When both "d" and "n" are omitted, it operates assuming that "d" is specified. Always specify "e" for the final segment.

opt	Meaning
a	It sorts data in ascending order.
d	The index can have duplicate key values. It is in an exclusive relation with "n".
e	It shows the end of segments.
n	The index is a unique key. It is in an exclusive relation with "d".

4.2.8.11 linedlmt operand

This specifies record separators in the output files.
It is effective in text files.

Format

```
linedlmt=line-delimiter
```

line-delimiter

This specifies the record separators in the output file.
The record separators that can be specified are shown below. When omitting the specification, it is the same as the record separators of the input file.

line-delimiter	Meaning
crlf	CRLF is treated as a record separator.
cr	CR is treated as a record separator.
lf	LF is treated as a record separator.

Information

The record separator in the input file is automatically detected from among the input file.

Note

When two or more record separator exists in the input file, it is not possible to operate normally.

4.2.8.12 removeeof operand

When the output file is a text file, EOF control character is removed.

Format

```
removeeof
```

4.2.9 Input record information option (-record)

This specifies the input record information.

Format

```
-record [ recform=record-format ]
        { [ fldsep=field-separate ] | [ fldsepx=field-separate-hex ] }
```

4.2.9.1 recform operand

This specifies the record format.

Format

```
recform=record-format
```

record-format

This specifies the record format.
When the record format is omitted, it operates assuming that the binary file fixed-length record format is specified in the standard file system of the system. When other files are processed, this specification cannot be omitted. Refer to PowerBSORT Input Output Environment and Specifying fields for the record format that can be specified. Select one of the following.

record-format	Meaning
fix	Binary file fixed-length record format
var	Binary file variable-length record format
txtfix	Text file fixed field specification
txtflt	Text file floating field specification
txtcsv	Text file CSV format
txttsv	Text file TSV format

Note

The binary file variable-length record format cannot be specified at the file of the standard file system of the system.

4.2.9.2 fldsep operand

This specifies field separation character string for the text file floating field specification.
The fldsep operand has an exclusive relationship with the fldsepx operand. When either specification of fldsep operand and fldsepx operand is omitted, tabs or spaces are treated as separation characters of the field.

Format

```
fldsep=field-separate
```

field-separate

The field separation character string is specified.
The backslash (\) is specified before the character when the character with a special meaning as a shell is contained in the field separating character string. (example: blank, $, &, ', ", (,), \, |, `, *, <, >, ?)

Example

1. Characters "SEPARATE" are specified as field separation character string.

   ```
   fldsep=SEPARATE
   ```

2. Characters "F S" are specified as field separation character string.

   ```
   fldsep=F\ S
   ```

3. Characters "'" are specified as field separation character string.

   ```
   fldsep=\'
   ```

4. Characters """ are specified as field separation character string.

   ```
   fldsep=\"
   ```

5. Characters "\" are specified as field separation character string.

   ```
   fldsep=\\
   ```

6. Characters "**" are specified as field separation character string.

```
fldsep=\*\*
```

7. Characters "* *" are specified as field separation character string.

```
fldsep=\*\ \*
```

4.2.9.3 fldsepx operand

This specifies field separation character string by the hexadecimal number for the text file floating field specification.
The fldsepx operand has an exclusive relationship with the fldsep operand. When either specification of fldsepx operand and fldsep operand is omitted, tabs or spaces are treated as separation characters of the field.

Format

```
fldsepx=field-separate-hex
```

field-separate-hex

The field separation character string is specified by the hexadecimal number.

Example

1. Character string "AB" (ASCII code) is specified by the hexadecimal number for a field separation character string.

```
fldsepx=4142
```

2. Character string "<->" (UTF-32LE) is specified by the hexadecimal number for a field separation character string.

```
fldsepx=600000002d00000062000000
```

4.2.10 Sort option (-sort)

This specifies the sort option.
The Sort option (-sort), Merge option (-merge), and Copy option (-copy) are mutually in the exclusive relationship. When the Sort option (-sort), Merge option (-merge), and Copy option (-copy) are omitted, it is assumed that the Sort option (-sort) having the entire record as a key is specified. For information about each operand, refer to the descriptions in the Merge option (-merge).

Format

```
-sort [ key=key-definition [ ,key-definition ...] [ key= ...] ]
```

4.2.11 Record summation option (-summary)

This executes the record summation processing or suppression processing to all the output records ahead of the output processing.
The Record summation option (-summary) is effective in the Sort option (-sort) and Merge option (-merge). When specified at the same time as the first-in first-out (FIFO) option (environment variable BSORT_FIFO, fifo operand of Execution environment option (-option), or BSORT_FIFO of startup file), the specification of the first-in first-out (FIFO) option is disregarded.

Format

```
-summary { field=summary-definition [ ,summary-definition ...] [ field= ...]
             | suppress } [ { first | last } ]
```

4.2.11.1 field operand

To use the record summation option, specify the summation field.
When the values of the specified key fields are equal, add the summation fields to make 1 record. In the case of text files, only the numbers described in ASCII code and Unicode are to be processed. The record summation option (field operand) has an exclusive relationship with the suppress operand.

Format

```
field=summary-definition
```

- **Format of *summary-definition***

```
pos.len typ [ opt ]
```

pos

This specifies the position of the summation field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

len

This specifies the length of the summation field in the number of bytes.
For fields that are longer than the specified field, process the text file floating field, the text file CSV format, and the text file TSV format with the specified field length. When a field that is shorter than the specified field length appears, it enhances to the specified field length.

In the text file CSV format and text file TSV format specification, the length of the summation field does not contain the double quotation marks (") that enclose the field.

Example: Summation field and length in text file CSV format.

```
Summation field    Characters effective as    Length of summation field
                   summation field
123                123                        3bytes
"123"              123                        3bytes
```

typ

This specifies the data format of the summation field.
For more information, refer to the Data formats that can be specified in the summation field.

opt

This specifies the output format options of the summation field for a text file.
If these are specified for a binary file, an error occurs. The output format options that can be specified are shown below.

opt	Meaning
i	Appends a sign to the value in the summation field. This option and the u option are mutually exclusive.
u	Appends a minus sign (-) to the value in the summation field if that value is negative. This option and the i option are mutually exclusive.
z	If the length of the value in the summation field is less than the field length, the area to the left of the value is padded with zeroes (0). For example, if the value in the summation field is "1234", and the summation field length is six bytes, the value is recorded as "001234". This option, the b, and d options are mutually exclusive.

opt	Meaning
b	If the length of the value in the summation field is less than the field length, the area to the left of the value is padded with spaces. For example, if the value in the summation field is "1234", and the summation field length is six bytes, the value is recorded as " 1234". This option, the z, and d options are mutually exclusive.
d	Deletes any spaces, tabs, or zeroes that appear at the head of the summation field. If there are any spaces, tabs, or zeroes at the head of the summation field, the field is evaluated from the left and any spaces, tabs, or zeroes that appear before the first number (other than zero) are deleted. For example, if the value in the summation field is "00123", it is changed to "123". Note that, if the value in the summation field is equivalent to zero (such as "0", "0000" or "+000"), the last "0" is not deleted. This option can be specified for the text file floating field, text file CSV format, and text file TSV format. This option, the z, and b options are mutually exclusive.

Note

- opt d can be specified for the text file floating field, text file CSV format, and text file TSV format. If it is specified for a text file fixed field, an error occurs.
- When summating with opt i specified, if the summated value becomes zero, a plus sign (+) is appended.
- When opt b is specified and a plus or minus sign exists, the sign is positioned immediately before the number.
- When neither opt i nor u is specified, processing is as follows:
  - If the summation result is a negative value, a minus sign (-) is included in the summation result.
  - If the summation result of unsigned data and signed data is a positive value, no sign is included in the summation result.
  - If two unsigned data items are summated, no sign is included in the summation result.
  - If signed and unsigned data are summated and the result is zero, no sign is included.
  - If two signed data items are summated, a sign is included in the summation result. If the summation result is zero, a plus sign (+) is included.
- When a sign is to be included in the value in the summation field, if the value in the summation field is already as long as the specified summation field length, the sign cannot be included and this causes an overflow to occur.
- When neither z, b nor d are specified, processing is as follows:
  - When the summation field is summated under the conditions listed below, if the length of the result in the summation field is less than the field length, the area to the left of the value is padded with zeroes (0).
    - When both summated data items are padded to the left with zeroes
    - When one summated data item is padded to the left with zeroes and the other is padded to the left with spaces
    - When one summated data item is padded to the left with zeroes and the other is padded to the left with tabs
  - When the summation field is summated under the conditions listed below, if the length of the result in the summation field is less than the field length, the area to the left of the value is padded with spaces.
    - When both summated data items are padded to the left with spaces
    - When both summated data items are padded to the left with tabs
    - When one summated data item is padded to the left with spaces and the other is padded to the left with tabs
- Summation fields of records that are not targeted for summation (that is, records with identical key field values for which no other record exists) are processed as follows:
  - If *opt* is specified, results are output in the format specified by *opt*.
  - If *opt* is not specified, results are output in the input format.

Example

1. To specify the field of the packed decimal number in 8 bytes from the 7th byte of the input record for the summation field, specify as follows:

```
-summary field=6.8pdl
```

2. To specify the field of ASCII code in 8 bytes in length from the 4th byte of the input record for the summation field in the text file. Appends a sign to the value in the summation field. If the length of the value in the summation field is less than the field length, the area to the left of the value is padded with spaces.

```
-summary field=3.8ascib
```

Note

- Of the records that are objects of the summation process, it is not possible to predict which record is output with the summation results. With the bsortex command, it is possible to specify the record that is output with the summation result using the first operand or last operand.
- Specify so that the summation field does not overlap a key field or another summation field.
- Note also that the summation field must be completely included in the record.
- When using the record summation feature, the key field specification cannot be omitted.
- If an overflow occurs during addition of a summation field, subsequent behavior is determined by the specification of the BSORT_SUMOVERCONT of startup file.
- Numbers with decimal points cannot be summated.
- For text files, only single-byte numbers in ASCII code, Unicode UCS-2 format, Unicode UTF-32 format, or Unicode UTF-8 format are processed.
- With a text file floating field specification, if the summation field contains a field separation character and this causes the summation field position to change, correct processing cannot be guaranteed.
- With a text file floating field specification, summation results are processed to the specified field length. When a field that is longer than the specified field length appears, the part that exceeds the specified field length is output without changing the content. If the field is shorter than the specified field length, it is extended to the specified field length, and then processed.

  Example: Specify "1.5asc" for the summation field. Specify "0.3asca" for the key field. The field separation character string is a comma (,).

```
Input record                  Output record
001,12345ABC,OPQ (Note1)      001,12456ABC,OPQ (Note2)
001,111,RST                   002,00127,UVW (Note3)
002,15,UVW (Note1)
002,00112DEF,XYZ

NOTE
1. Assume that the record shown here is the one output by the record summation option.
2. Part ("ABC") that exceeds the specified field length is output without changing the content.
3. When the field is shorter than the specified field length, it is extended to the specified field
length.
```

- In the text file CSV format and text file TSV format, summation results are processed to the specified field length. When a field is longer than the specified field length, the part that exceeds the specified field length is not output. If the field is shorter than the specified field length, it is extended to the specified field length, and then processed.

  Example: Specify "1.5asc" for the summation field of the text file CSV format. Specify "0.3asca" for the key field.

```
Input record                    Output record
001,12345ABC,OPQ (Note1)        001,12468,OPQ (Note2)
001,123,RST                     002,00027,UVW (Note3)
002,15,UVW (Note1)
002,00012DEF,XYZ

NOTE
1. Assume that the record shown here is the one output by the record summation option.
2. Part ("ABC") that exceeds the specified field length is not output.
3. When the field is shorter than the specified field length, it is extended to the specified field
length.
```

- In the text file CSV format and the text file TSV format, whether the summation field is enclosed with double quotation marks (") or it doesn't enclose it is decided depending on the following conditions. When the field on the input record of the target for the output is enclosed with double quotation marks ("), the field after the record is summation is enclosed with double quotation marks ("). When the field on the input record of the target for the output is not enclosed with double quotation marks ("), the field after the record is summation is not enclosed with double quotation marks (").

  Example: Specify "1.5asc" for the summation field of the text file CSV format. Specify "0.3asca" for the key field.

```
Input record                    Output record
001,"12345",OPQ (Note)          001,"12468",OPQ
001,"123",RST                   002,"00027",UVW
002,"15",UVW (Note)             003,11900,GHI
002,00012,XYZ                   004,98769,MNO
003,11111,GHI (Note)
003,"00789",JKL
004,98765,MNO (Note)
004,4,PQR

NOTE
Assume that the record shown here is the one output by the record summation option.
```

- In the text file floating field specification, when a field that is shorter than the specified field length appears, it enhances to the specified field length. As a result, when the record length exceeds the specified maximum record length, it processes as an overflow.

4.2.11.2 suppress operand

This specifies the suppression option.
When the values of the specified key fields are equal, 1 record is left and others are deleted. The suppression option (suppress operand) has an exclusive relationship with the field operand.

Format

```
suppress
```

Example

The specification to use the suppression option is as follows.

```
-summary suppress
```

4.2.11.3 first operand

The first input record out of the records in the same key record is output.
It is in the exclusive relationship with last operand. When the first operand and the last operand are omitted, the output record is irregularly output from among the record with the same key field.

Information

When the first operand is specified by the merge option, one output record is chosen from among the record with the same key field. The record outputs the record in the input file specified first. The record in the same file outputs the first record in the file.

Format

```
first
```

Example

In the record summation option, specify it as follows when you want to output the first input record.

```
-summary field=9.4zdl first
```

4.2.11.4 last operand

The last input record out of the records in the same key record is output.
It is in the exclusive relationship with first operand. When the first operand and the last operand are omitted, the output record is irregularly output from among the record with the same key field.

Information

When the last operand is specified by the merge option, one output record is chosen from among the record with the same key field. The record outputs the record in the input file specified at the last. The record in the same file outputs the last record in the file.

Format

```
last
```

Example

In the suppression option, specify it as follows when you want to output the last input record.

```
-summary suppress last
```

Chapter 5 Using PowerBSORT with a COBOL program

PowerBSORT can be used from COBOL programs created by NetCOBOL.

- The runtime system of NetCOBOL automatically calls PowerBSORT once PowerBSORT has been installed. When a NetCOBOL program uses the SORT or MERGE statement, PowerBSORT is substituted for the intrinsic sort/merge function.
- Existing NetCOBOL programs will use PowerBSORT automatically and do not require source code changes, recompilation or special runtime settings.

Chapter 6 Using PowerBSORT with a C language program

PowerBSORT can be used by C language programs.
When PowerBSORT is used from C language program, the BSORT function is used. This section explains overview of each BSORT function, its usage and notes for developing C language applications using the BSORT function.

6.1 What is the BSORT function

The BSORT function is a generic name of the function to use PowerBSORT from the C language program. Five functions are provided.

BSORT function list

Function name	Overview
bsrtopen	This loads shared library of PowerBSORT into the virtual storage and requests the address of other functions. In addition, it prepares the execution environment of PowerBSORT.
bsrtclse	This deletes the execution environment of PowerBSORT. Also, it deletes shared library of PowerBSORT loaded into the virtual storage in bsrtopen function from the virtual storage.
bsrtput	This passes records from user applications to shared library of PowerBSORT in the sort option.
bsrtget	This receives the sorted record from shared library of PowerBSORT in the sort option.
bsrtmrge	This passes records from user application to shared library of PowerBSORT and receives the merged records from PowerBSORT in the merge option.

Combinations of functions

BSORT functions include five functions as shown in the BSORT function list.
Functions to be used depend on the options to use and forms of input and output. The combinations of BSORT functions that are related to the functions of PowerBSORT are shown below.

Table 6.1 Options of PowerBSORT for use and related BSORT function list

Function name	Sort option	Merge option	Copy option
bsrtopen	Always used	Always used	Always used
bsrtclse	Always used	Always used	Always used
bsrtput	Can be used	Not used	Not used
bsrtget	Can be used	Not used	Not used
bsrtmrge	Not used	Can be used	Not used

6.2 BSORT function usage

In this section, it explains the usage of the BSORT function when sort option, merge option, and copy option are used.

6.2.1 Sort option

When using the sort option, the following four types are available depending on the styles of input and output.

- Form that uses file (including standard input and standard output) for both input and output
- Form that uses user application (bsrtput function and bsrtget function) for both input and output
- Form that uses file (including standard input) for input and user application (bsrtget function) for output
- Form that uses user application (bsrtput function) for input and file (including standard output) for output

Form that uses file (including standard input and standard output) for both input and output

When the file (including standard input and standard output) is used for the input and the output, execute the following procedure.

1. Issue the bsrtopen function.
2. Issue the bsrtclse function.

Information

Specifying an input file path name (or standard input) and output file path name (or standard output) in parameters of the bsrtopen function makes this form.

Form that uses user application (bsrtput function and bsrtget function) for both input and output

When the user application (bsrtput function and bsrtget function) is used for the input and the output, execute the following procedure.

1. Issue the bsrtopen function.
2. Issue the bsrtput function.
   The bsrtput function is issued continuously until the record passed to PowerBSORT is lost.
3. Issue the bsrtput function and notify the end of the records.
4. Issue the bsrtget function.
   The bsrtget function is issued continuously from PowerBSORT to the notification of the end of the record.
5. Issue the bsrtclse function.

Information

Not specifying an input file path name (or standard input) and output file path name (or standard output) in parameters of the bsrtopen function makes this form.

Form that uses file (including standard input) for input and user application (bsrtget function) for output

When the file (including standard input) is used for the input, and the user application (bsrtget function) is used for the output, execute the following procedure.

1. Issue the bsrtopen function.
2. Issue the bsrtget function.
   The bsrtget function is issued continuously from PowerBSORT to the notification of the end of the record.
3. Issue the bsrtclse function.

Information

Specifying an input file path name (or standard input) and not specifying output file path name (or standard output) in parameters of the bsrtopen function makes this form.

Form that uses user application (bsrtput function) for input and file (including standard output) for output

When the user application (bsrtput function) is used for the input, and the file (including standard output) is used for the output, execute the following procedure.

1. Issue the bsrtopen function.
2. Issue the bsrtput function.
   The bsrtput function is issued continuously until the record passed to PowerBSORT is lost.

3. Issue the bsrtput function and notify the end of the records.
4. Issue the bsrtclse function.

Information

Not specifying an input file path name (or standard input) and specifying output file path name (or standard output) in parameters of the bsrtopen function makes this form.

6.2.2 Merge option

When using the merge option, the following four types are available depending on the styles of input and output.

- Form that uses file (including standard output) for both input and output
- Form that uses user application (bsrtmrge function) for both input and output
- Form that uses file for input and user application (bsrtmrge function) for output
- Form that uses user application (bsrtmrge function) for input and file (including standard output) for output

Form that uses file (including standard output) for both input and output

When the file (including standard output) is used for the input and the output, execute the following procedure.

1. Issue the bsrtopen function.
2. Issue the bsrtclse function.

Information

Specifying an input file path name and output file path name (or standard output) in parameters of the bsrtopen function makes this form.

Form that uses user application (bsrtmrge function) for both input and output

When the user application (bsrtmrge function) is used for the input and the output, execute the following procedure.

1. Issue the bsrtopen function.
2. Issue the bsrtmrge function.
   The bsrtmrge function is issued continuously from PowerBSORT to the notification of the end of the record. In the bsrtmrge function in this form, there are 2 functions: one passes records from the user application to PowerBSORT and the other one receives records from PowerBSORT. In issuing the bsrtmrge function, the record of individual string (record group that have been sorted) is passed and the records to be returned from PowerBSORT are received. When the end of the record is notified to all the strings in process, bsrtmrge function is issued continuously until the end of the record is notified even if there is no record to be passed and the record to be returned from PowerBSORT is received.
3. The bsrtclse function is issued at the end if the end of the record is notified from PowerBSORT.

Information

Not specifying an input file path name and output file path name (or standard output) in parameters of the bsrtopen function makes this form.

Form that uses file for input and user application (bsrtmrge function) for output

When the file is used for the input, and the user application (bsrtmrge function) is used for the output, execute the following procedure.

1. Issue the bsrtopen function.

2. Issue the bsrtmrge function.
   The bsrtmrge function is issued continuously from PowerBSORT to the notification of the end of the record. The bsrtmrge function in this form functions to receive records from PowerBSORT.
3. The bsrtclse function is issued at the end if the end of the record is notified from PowerBSORT.

Information

Specifying an input file path name and not specifying output file path name (or standard output) in parameters of the bsrtopen function makes this form.

Form that uses user application (bsrtmrge function) for input and file (including standard output) for output

When the user application (bsrtmrge function) is used for the input, and the file (including standard output) is used for the output, execute the following procedure.

1. Issue the bsrtopen function.
2. Issue the bsrtmrge function.
   The bsrtmrge function is issued continuously from PowerBSORT to the notification of the end of the record. The bsrtmrge function in this form functions to pass PowerBSORT the record. In issuing the bsrtmrge function, the record of individual string is passed and the end of the records is notified when the records of the string have been lost. When the end of the record is notified to all the strings in process, the function is continuously issued until PowerBSORT notifies the end of the record even if there is no record to be passed.
3. The bsrtclse function is issued at the end if the end of the record is notified from PowerBSORT.

Information

Not specifying an input file path name and specifying output file path name (or standard output) in parameters of the bsrtopen function makes this form.

6.2.3 Copy option

When you use the copy option, the form that uses file (including standard input and standard output) for input and output are available. The form that can pass records to PowerBSORT or receive records from PowerBSORT as in the sort option and merge option is not available.
To use the copy option, execute the following procedure.

1. Issue the bsrtopen function.
2. Issue the bsrtclse function.

6.3 Points of concern when C language program is developed

In this section, it explains points of concern when the C language program that uses the BSORT function is developed.

About the environment variables and startup file

The environment variable and the startup file influence not only the bsortex command, and the bsort command but also the executions of the C language program that uses the BSORT function. Therefore, when the C language program that uses the BSORT function is executed, it is necessary to set the appropriate environment variables and the startup file.

About a common environment variable to the system

LC_* represents system environment variables. To enable system environment variables, issue setlocale(3) before calling a BSORT function. The setlocale function can be omitted in English based systems in the USA. However, if the setlocale(3) function is not issued in a region other than the USA, features provided for your desired region (e.g.,messages in Japanese) cannot be used.

About the link with the library

An application program using BSORT functions must be linked with a BSORT supported library. To create an executable format program by the gcc command, link the library as shown below.

```
gcc source-file-name -I/opt/FJSVXbsrt/include -L/opt/FJSVXbsrt/lib -lbsrt -ldl
```

About the MultiThreading operation

The BSORT function doesn't correspond to the C language program that operates by the MultiThreading.

6.4 BSORT function types

This section provides additional information about each BSORT function.

6.4.1 bsrtopen function

The bsrtopen function loads the shared library of PowerBSORT into the virtual memory storage, and sets the address of each function in parameter func. In addition, the execution environment of PowerBSORT is prepared.

Format

```
#include <bsort/bsrt.h>
int bsrtopen (long int BSRTVL, BSPTR_BSRTFUNC func,
              BSPTR_BSRTPRIM prim, BSPTR_BSRTREC rec,
              BSPTR_BSRTKEY key, BSPTR_BSRTFILE file,
              BSPTR_BSRTOPT opt);
```

Function explanation

The function of the bsrtopen function is shown below.

- The execution environment of PowerBSORT is constructed.
- When using files for input (including standard input) and user applications for output, PowerBSORT executes sort processing in the sort option and PowerBSORT returns right before the result is output. In the merge option, records are input from each file little by little and it returns right before the merge processing is executed.
- When specifying files for input and output (including standard input and standard output), the sort processing, merge processing, and copy processing are executed and then it returns.

Parameter detail

In the following, it explains the parameter of the bsrtopen function.

BSRTVL

This specifies version levels (BSRTVL) of PowerBSORT.
BSRTVL is defined in header file (bsrt.h). This parameter enables PowerBSORT to be operated without changing the C language program even if there is a change in the interface of the BSORT function.

func

This sets each address of the BSORT function (bsrtput function, bsrtget function, and bsrtmrge function).

Information

BSPTR_BSRTFUNC is a pointer of BSRTFUNC structure.

prim

It is a parameter that specifies the sort option, merge option, copy option, record selection option, record reconstruction option, first in first out (FIFO) option, the record summation option, and the suppression option.

Information

BSPTR_BSRTPRIM is a pointer of BSRTPRIM structure.

rec

No information is specified in the bsrtopen function. The detail code is set when an error occurs.

Information

BSPTR_BSRTREC is a pointer of BSRTREC structure.

key

The key field in the sort option or merge option is specified.

Information

BSPTR_BSRTKEY is a pointer of BSRTKEY structure.

file

It is a parameter that specifies file information such as an input file, output file, and temporary file.

Information

BSPTR_BSRTFILE is a pointer of BSRTFILE structure.

opt

It is a parameter that specifies option options.
The pointer of each structure to specify the summation field, selection field, reconstruction field and the number of skipping records is specified. When neither of pointers of the above mentioned structures is specified, NULL is specified for opt parameter of the bsrtopen function. When specifying any, specify it after the area is secured respectively.

Information

BSPTR_BSRTOPT is a pointer of BSRTOPT structure.

Return value

The return values of the bsrtopen function are as follows.

Return value	Meaning
0	This shows that the bsrtopen function ended normally. Issue the bsrtput function, bsrtget function, bsrtmrge function or bsrtclse function continuously.

Return value	Meaning
-1	This shows that an error occurred in the bsrtopen function. Detailed information is set in errdetail of BSRTREC structure. Issue the bsrtclse function continuously.
-2	This shows that an error occurred in the bsrtopen function. Detailed information is set in errdetail of BSRTREC structure. The bsrtclse function need not be issued for this return value.

Note

- Other parameters than BSRTVL should be cleared to be 0 before the bsrtopen function is issued even when the value is not set. As for the parameter to which value is set, clear 0 in field before setting.
- Among parameters of the bsrtopen function, BSRTVL, func, prim, and rec cannot be omitted. The key, file, and opt can be omitted. Specify NULL for the pointer in the parameter of the bsrtopen function when omitting them.

  Example) Specification when key, file, and opt are omitted

```
bsrtopen(BSRTVL, func, prim, rec, (BSPTR_BSRTKEY)NULL, (BSPTR_BSRTFILE)NULL, (BSPTR_BSRTOPT)NULL);
```

6.4.2 bsrtclse function

The bsrtclse function deletes the execution environment of PowerBSORT that the bsrtopen function constructed.

Format

```
#include <bsort/bsrt.h>
int bsrtclse (long int BSRTVL, BSPTR_BSRTFUNC func,
              BSPTR_BSRTPRIM prim, BSPTR_BSRTREC rec);
```

Function explanation

The function of the bsrtclse function is shown below.

- The execution environment of PowerBSORT constructed by the bsrtopen function is deleted (releasing the work area and close of an I/O file and a temporary file and so on).
- Shared library of PowerBSORT loaded in the bsrtopen function is deleted from the virtual storage.
- When the bsrtclse function is called during the processing of the bsrtget function and bsrtput function, the processing is interrupted.

Parameter detail

In the following, it explains the parameter of the bsrtclse function.

BSRTVL

This specifies version levels (BSRTVL) of PowerBSORT.
BSRTVL is defined in header file (bsrt.h).

func

This specifies func specified in bsrtopen function.

Information

BSPTR_BSRTFUNC is a pointer of BSRTFUNC structure.

prim

This specifies prim specified in bsrtopen function.

Information

BSPTR_BSRTPRIM is a pointer of BSRTPRIM structure.

rec

This specifies rec specified in bsrtopen function.

Information

BSPTR_BSRTREC is a pointer of BSRTREC structure.

Return value

The return values of the bsrtclse function are as follows.

Return value	Meaning
0	This shows that the bsrtclse function ended normally.
1	This shows that the interruption of processing ended normally when the interruption of processing is directed. In the following situation, if the bsrtclse function is issued, processing will be interrupted. - When the user application does not notify the end of data in bsrtput function or bsrtmrge function. - When the bsrtget function or bsrtmrge function does not notify the end of data to the user application.
-1	This shows that an error occurred in the bsrtclse function. Detailed information is set in errdetail of BSRTREC structure.

Note

In the parameter of the bsrtclse function, there is no member that the user application should newly set.

6.4.3 bsrtput function

When the sort option is specified in the bsrtopen function and records are passed from user applications to PowerBSORT, the bsrtput function is used.

Format

```
#include <bsort/bsrt.h>
BSRTFUNC func; (Note)
int func.bsrtput(BSPTR_BSRTREC rec);
```

NOTE

func is a parameter specified in thebsrtopen function.

Function explanation

The function of the bsrtput function is shown below.

- Records or record groups are passed from the user applications to PowerBSORT.
- The end of the records or record groups is notified to PowerBSORT.

How to pass record group

When two or more records are passed to PowerBSORT at once, specify the length of the record group in rec_len of BSRTREC structure. The length of the record group is in multiple of the number of records of record length in the fixed-length record format. In the variable-length record format, area of the record length (long type of C language) is prefaced for each record. The length of record group is accumulated length of the record and record length area. The record length does not include the area of the record length. In text file, the record that includes line feed character is stored in a continuous area. The length of the record group is the length from the head of the record to the line feed character of the last record.

Parameter detail

In the following, it explains the parameter of the bsrtput function.

rec

This specifies rec specified in bsrtopen function.

Information

BSPTR_BSRTREC is a pointer of BSRTREC structure.

Return value

The return values of the bsrtput function are as follows.

Return value	Meaning
0	This shows that the bsrtput function ended normally.
1	This shows that the end of the record or record group was normally accepted.
-1	This shows that an error occurred in the bsrtput function. Detailed information is set in errdetail of BSRTREC structure.

6.4.4 bsrtget function

When the sort option is specified in the bsrtopen function and record of the sort result is received from PowerBSORT, the bsrtget function is used.

Format

```
#include <bsort/bsrt.h>
BSRTFUNC func; (Note)
int func.bsrtget(BSPTR_BSRTREC rec);
```

NOTE

func is a parameter specified in thebsrtopen function.

Function explanation

The function of the bsrtget function is shown below.

- Record or record group is received from PowerBSORT.
- The end of record or record group is notified from PowerBSORT.

Format of record group to be returned

Individual record is stored and returned to a continuous area.
For the fixed-length record format, the address on the first record is set in rec_addr of BSRTREC structure and the length of the record group (in multiple of the number of the records of the record length) is set in rec_len of BSRTREC structure. For the variable-length record format, the address on the first record is set in rec_addr of BSRTREC structure in the form with the area where the record length (long type of C language) was stored in the head of each record and length in which the area of the record and the record length are accumulated is set in rec_len of BSRTREC structure. For the record in text files, record that includes line feed characters is stored and returned to a continuous area. The address of the first record is set in rec_addr of BSRTREC structure and the length in which the records to be returned are accumulated is set in rec_len of BSRTREC structure.

Parameter detail

In the following, it explains the parameter of the bsrtget function.

rec

This specifies rec specified in bsrtopen function.

Information

BSPTR_BSRTREC is a pointer of BSRTREC structure.

Return value

The return values of the bsrtget function are as follows.

Return value	Meaning
0	This shows that the bsrtget function ended normally.
1	This shows that the bsrtget function ended normally. Finished passing all the records, NULL was set in rec_addr of BSRTREC structure and 0 was set in rec_len of BSRTREC structure. In a normal processing, issue the bsrtget function repeatedly until the return value of the bsrtget function becomes 1.
-1	This shows that an error occurred in the bsrtget function. Detailed information is set in errdetail of BSRTREC structure.

6.4.5 bsrtmrge function

When the merge option is specified in the bsrtopen function and records to be merged are passed to PowerBSORT and when the records from the merge result are received from PowerBSORT, the bsrtmrge function is used.

Format

```
#include <bsort/bsrt.h>
BSRTFUNC func; (Note)
int func.bsrtmrge(BSPTR_BSRTREC rec);
```

NOTE

func is a parameter specified in thebsrtopen function.

Function explanation

The function of the bsrtmrge function is shown below.

- Records to be merged are passed from user applications to PowerBSORT.
- The records from the merge result are returned from PowerBSORT to the user application.
- The end of the string is notified to PowerBSORT. A group of records that have been sorted is called a string.

- The end of the record from merge result is notified from PowerBSORT to the user application.

Parameter detail

In the following, it explains the parameter of the bsrtmrge function.

rec

This specifies rec specified in bsrtopen function.

Information

BSPTR_BSRTREC is a pointer of BSRTREC structure.

Return value

The return values of the bsrtmrge function are as follows.

Return value	Meaning
0	This shows that the bsrtmrge function ended normally.
1	This shows that the bsrtmrge function ended normally. Finished passing all the records, NULL was set in mgrec_addr of BSRTREC structure and 0 was set in mgrec_len of BSRTREC structure.
-1	This shows that an error occurred in the bsrtmrge function. Detailed information is set in errdetail of BSRTREC structure.

6.5 BSORT structures

This section explains structures used in the BSORT functions.

- Setting basic information
  - BSRTPRIM structure
- Setting record information
  - BSRTREC structure
- Setting key field information
  - BSRTKEY structure
  - BSKEY structure
  - BSCOL structure
- Setting file information
  - BSRTFILE structure
  - BSFILE structure
  - BSFSYS structure
  - BSIDX structure
  - BSIDXKEY structure
  - BSFILE_EXT structure
  - BSFILE_BASE structure

- Setting record option information
  - BSRTOPT structure
  - BSRTSUM structure
  - BSSUM structure
  - BSRTSELE structure
  - BSSELE structure
  - BSRTRCON structure
  - BSRCON structure
  - BSRTSKIP structure
- Other structure
  - BSRTFUNC structure

6.5.1 BSRTPRIM structure

The BSRTPRIM structure is a structure to specify primary basic information of the sort processing, the merge processing, the copy processing, and etc.

```
typedef struct {
    unsigned long    reserve1;      /* reserved */
    unsigned char    function;
    unsigned char    recform;
    unsigned char    optionfunc;
    unsigned char    fileoprat;
    unsigned char    recoprat;
    unsigned char    msglevel;
    unsigned char    cdmode;
    unsigned char    chklevel;
    unsigned long    rec_len;
    unsigned long    input_string;
    unsigned long    memory_size;
    BSPTR_VOID       memory_addr;
    unsigned long    input_recno;
    BSPTR_UCHAR      fldchar_addr;
    unsigned char    debuginfo;
    unsigned char    fieldmode;
    unsigned char    keyoption;
    unsigned char    keyoption2;
    unsigned char    linedlmt;
    unsigned char    error_happened;
    unsigned char    altmode;
    unsigned char    reserve4;      /* reserved */
    BSPTR_UCHAR      reserve5;      /* reserved */
} BSRTPRIM;
```

BSRTPRIM structure member

In the following, it explains about the member of the BSRTPRIM structure.

function

This specifies the main options in PowerBSORT.
Select one out of the following. This specification cannot be omitted.

Define value	Meaning
BS_SORT	Sort option

Define value	Meaning
BS_MERGE	Merge option
BS_COPY	Copy option

recform

This specifies record format.
Select one out of the following. In the text file, recform need not be specified.

Define value	Meaning
BS_FREC	Fixed-length record format
BS_VREC	Variable-length record format

optionfunc

This specifies option functions.
Specify the following according to the necessity. When selecting two or more values specify the logical sum.

Define value	Meaning
BS_FIFO	This shows the first-in first-out (FIFO) option is specified. When specified at the same time as the suppression option or the record summation option, the specification of the first-in first-out (FIFO) option is ignored. When omitted, the output order is not necessarily first-in first-out.
BS_SUPPRESS	This shows the suppression option is specified. It is in the exclusive relationship with the record summation option.
BS_SUM	This shows the record summation option is specified. It is in the exclusive relationship with the suppression option. When this specification is set, you need to set an address of the summation table (BSRTSUM structure) in BSRTOPT structure.
BS_SELECT	This shows the record selection option is specified. When this specification is set, you need to set an address of the selection table (BSRTSELE structure) in BSRTOPT structure.
BS_RECON	This shows the record reconstruction option is specified. When this specification is set, you need to set an address of the reconstruction table (BSRTRCON structure) in BSRTOPT structure.
BS_REVERSE	This shows the sort processing or merging processing is executed in reverse order. This is effective when the key field is omitted.
BSMSG_STDERR	This shows messages are output to the standard error output. (Note)

NOTE

The output message depends on msglevel of BSRTPRIM structure or BSORT_MSGLEVEL of startup file.

fileoprat

This specifies input output information.
Specify the following according to the necessity. When selecting two or more values specify the logical sum.

Define value	Meaning
BS_OVERWRITE	This shows the processing is continued when the sort option is specified and one of input files is the same as the output file. Note that the data of the I/O file might not be guaranteed when this option is specified and an error occurs in the sort processing.

Define value	Meaning
BS_STDIN	This shows record is input from standard input. This cannot be specified in the merge option. When the input file is specified in BSRTFILE structure, it is ignored.
BS_STDOUT	This shows result is output to standard output. When the output file is specified in BSRTFILE structure, it is ignored.

recoprat

This specifies method of handing over record with PowerBSORT.
When selecting two or more, select each one method from handling methods for input and ones for output and specify the logical sum.

- Method for handling over input

Define value	Meaning
BS_IS	This shows each record is passed to PowerBSORT by one record by using the bsrtput function or bsrtmrge function.
BS_IM	This shows two or more records are passed to PowerBSORT by using the bsrtput function. This option cannot be specified for the merge option.

- Method for handling over output

Define value	Meaning
BS_OS	This shows each record is received from PowerBSORT by one record by using the bsrtget function or bsrtmrge function.
BS_OM	This shows two or more records are received from PowerBSORT by using the bsrtget function. This option cannot be specified for the merge option.

msglevel

This specifies level of the message that PowerBSORT outputs.
Select one out of the following.

Define value	Meaning
BSMSG_LEVEL_N	Nothing is output.
BSMSG_LEVEL_E	Error message is output.
BSMSG_LEVEL_W	Error message and warning message are output.
BSMSG_LEVEL_I	Error message, warning message, and information message are output.

The level of the message used in previous versions can also be specified.

Define value	Meaning
BSMSG_LEVEL0	Nothing is output.
BSMSG_LEVEL1	Error message and warning message are output.
BSMSG_LEVEL2	Error message, warning message, and information message are output.

To output the message to the standard error output, BSMSG_STDERR should be set in optionfunc of BSRTPRIM structure.

cdmode

This specifies code system of the input file.
Select one out of the following.

Define value	Meaning
BSZD_AU	ASCII code system (NetCOBOL form)
BSZD_AC	ASCII code system (Micro Focus COBOL form)
BSZD_EBCDIC	EBCDIC code system (Note)
BSZD_UCS2	Unicode system (UCS-2 form) (Byte order of system standard)
BSZD_UCS2B	Unicode system (UCS-2 form) (Byte order of big endian)
BSZD_UCS2L	Unicode system (UCS-2 form) (Byte order of little endian)
BSZD_UTF32	Unicode system (UTF-32 form) (Byte order of system standard)
BSZD_UTF32B	Unicode system (UTF-32 form) (Byte order of big endian)
BSZD_UTF32L	Unicode system (UTF-32 form) (Byte order of little endian)
BSZD_UTF8	Unicode系(UTF-8形式)

NOTE

The EBCDIC code system can be specified for binary file.

chklevel

This specifies check mechanism of the BSORT function.
Select one out of the following.

Define value	Meaning
BSCHK_LEVEL0	Nothing is checked.
BSCHK_LEVEL1	The interface check validates parameters in the bsrtput function, the bsrtget function, and the bsrtmrge function.

rec_len

This specifies length of the input record.
For binary files, the record length is specified when the record format is a fixed-length record form. When the record format is a variable-length record form, the maximum record length is specified. For text files, the maximum record length including the line feed code is specified.

Note

In the binary file (variable-length record form) and the text file, if the record length specified by *recsize* is longer than the actual maximum record length, it is possible to process. However, when the record length that is longer than the actual maximum record length is specified by *recsize*, it becomes the factor of the performance deterioration according to the difference between a specified record length and the actual maximum record length.

input_string

This specifies the number of input strings.
When the sort option or copy option is specified, set to 0. When the merge option is specified, set the number of strings.

Information

A group of records that have been sorted is called a string.

memory_size

This specifies size of the work area (memory size) that PowerBSORT uses in the number of bytes.
Specify it within the range from 65536 to 2147482624 bytes. 0 is specified when omitting it. When omitted, if the BSORT_MEMSIZE of startup file is specified, the work area of the size is used. If not, the work area is automatically set. When the value of 2147482625 or more is specified, 2147482624 is imposed.

memory_addr

This specifies address in the work area that PowerBSORT uses.
NULL is specified when omitting the specification. When memory_addr is specified, the size should be specified in memory_size.

Note

- When an invalid value is specified, the processing of PowerBSORT is not guaranteed.

input_recno

This is an unsupported member. Do not set anything.

fldchar_addr

This specifies address of field separation character string.
When text file floating field is specified, it is valid. The field character is a character string that ends with \0. When the specification of the field separation character string is omitted, NULL is specified. When the specification of the field separation character string is omitted, the tab and the blank are treated as a field separation character.

debuginfo

This is an unsupported member. Do not set anything.

fieldmode

This specifies how to decide field for text files.
Always specify for text files. Select one out of the following.

Define value	Meaning
BS_FLTFLD	This shows text file floating field specification. Each field is specified in the number of the field delimited by the field separation character string. Because field position is decided by the position of the field separation character, field position depends on each record.
BS_FIXFLD	This shows text file fixed field specification. Each field is specified by byte position.
BS_CSVFORM	This shows the text file CSV format specification. Each field is specified in the number of the field delimited by the comma (,). Because field position is decided by the position of the comma, field position depends on each record.

Define value	Meaning
BS_TSVFORM	This shows the text file TSV format specification. Each field is specified in the number of the field delimited by the tab. Because field position is decided by the position of the tab, field position depends on each record.

keyoption

This specifies operation of the key field.
When the specification of the key field is omitted in the case of text file, it is effective. Specify the following according to the necessity.
When selecting two or more values specify the logical sum.

Define value	Meaning
BSOPT_BLANK	Blanks and tabs in the head of the key field are disregarded.
BSOPT_DICTIONARY	Only the blank and the alphanumeric character are compared.
BSOPT_IGNORE	Control character codes are disregarded.
BSOPT_JUMBO	Lowercase letters are compared as uppercase letters.
BSOPT_NUMERIC	Character strings of the numbers that contain signs are compared with arithmetic value. The result is not guaranteed when characters other than the numbers exist in the character strings. It is in the exclusive relationship with BSOPT_CHARNUM and the BSOPT2_LOCALE of keyoption2.
BSOPT_CHARNUM	Connected data of alphabet and numbers (for instance, data123) are evaluated and sorted separately for the alphabet and the numerical value. After only alphabets are targeted for the comparison and compared, character strings exclusively with numbers are compared with arithmetic values. Data is evaluated from the left of the specified field, and the data that appeared after effective data is disregarded. It is in the exclusive relationship with BSOPT_NUMERIC and the BSOPT2_LOCALE of keyoption2.

keyoption2

This specifies operation of the key field.
When the specification of the key field is omitted for text file, it is effective. This member is in the exclusive relationship with BSOPT_NUMERIC and BSOPT_CHARNUM of keyoption. Specify one out of the following according to the necessity.

Define value	Meaning
BSOPT2_LOCALE	Compares according to the collating sequence defined in LC_COLLATE environmental variable.

linedlmt

This specifies record separators.
It is effective for text file. Select one out of the following.

Define value	Meaning
BSLDM_LF	LF is treated as a record separator. **(Default value)**
BSLDM_CR	CR is treated as a record separator.
BSLDM_CL	CRLF is treated as a record separator.

Note

When two or more record separator exists in the input file, it is not possible to operate normally.

error_happened

This is an unsupported member.

altmode

This specifies code conversion method between the EBCDIC code and the ASCII code.
Select one out of the following.

Define value	Meaning
BSALT_ASCII	It executes the code conversion between the EBCDIC code and the US ASCII code. **(Default value)**
BSALT_JIS8	It executes the code conversion between the EBCDIC (Japanese kana) code and the ASCII (JIS8) code.
BSALT_JIS7	It executes the code conversion between the EBCDIC (Lowercase letters) code and the ASCII (JIS7) code.

Information

In the following cases, the code conversion between the EBCDIC code and the ASCII code is done.

- Comparison of key field when data form of key field is EBCDIC code and when input code system (cdmode of BSRTPRIM structure) is ASCII code system.
- Comparison of key field when data form of key field is ASCII code and when input code system (cdmode of BSRTPRIM structure) is EBCDIC code system.

6.5.2 BSRTREC structure

The BSRTREC structure is a structure to receive and to pass the record between PowerBSORT and the user application.
In the BSRTREC structure, there is no member that user application sets before issuing the bsrtopen function.

```
typedef struct {
    BSPTR_VOID       sort_table;
    unsigned long    errdetail;
    unsigned long    in_count;
    unsigned long    out_count;
    BSPTR_VOID       rec_addr;
    unsigned long    rec_len;
    unsigned char    rec_status;
    unsigned char    mgrec_status;
    unsigned short   mgrec_string;
    BSPTR_VOID       mgrec_addr;
    unsigned long    mgrec_len;
    unsigned long    hProcFile;
    unsigned long    sub_error_code;
} BSRTREC;
```

BSRTREC structure member

In the following, it explains about the member of the BSRTREC structure.

sort_table

This sets an address in work area that PowerBSORT uses when the bsrtopen function is processed.
This member is a specific area that PowerBSORT uses and the user application should guarantee the content from the issue of the bsrtopen function to the completion of the issue of the bsrtclse function. The user application need not do anything to this member.

errdetail

When an error is detected while PowerBSORT is being executed, error detail codes of BSORT function is set.
The user application can know occurrence of error by return value of each BSORT function. The user applications don't have to refer to the member except for the case of error occurrence.

in_count

This is an unsupported member.

out_count

This is an unsupported member.

rec_addr

- For bsrtput function

  This sets the address of record passed to PowerBSORT.
  When BS_IM is specified for recoprat of the BSRTPRIM structure specified in bsrtopen function, the address of the record group is set.

- For bsrtget function

  This sets the address of record returned to user application.
  When BS_OM is specified for recoprat of the BSRTPRIM structure specified in the bsrtopen function, the address of the record group is set. NULL is set when there is no record or record group to be returned.

- For bsrtmrge function

  This sets address of the record passed to PowerBSORT.

rec_len

- For bsrtput function

  This sets the length of the record passed to PowerBSORT.
  When BS_IM is specified for recoprat of the BSRTPRIM structure specified in bsrtopen function, the length of the record group is set. When the record of the text file is passed, the length of the record including line feed character is set.

- For bsrtget function

  This sets the length of the record returned to user application.
  When BS_OM is specified for recoprat of the BSRTPRIM structure specified in the bsrtopen function, the length of the record group is set. 0 is set when there is no record or record group to be returned.

- For bsrtmrge function

  This sets the length of the record passed to PowerBSORT.
  When the record of the text file is passed, the length of the record including line feed character is set.

rec_status

- For bsrtput function

  This notifies whether the records have been passed.
  Set either of the following.

Define value	Meaning
BS_CONT	Records to be passed to PowerBSORT are left.
BS_ENDDATA	Record to be passed to PowerBSORT is not left. Set NULL for rec_addr and set 0 for rec_len when the end of the record is notified.

- For bsrtget function

  This is not used.

- For bsrtmrge function

  This notifies whether it is the final record.
  Set either of the following.

Define value	Meaning
BS_CONT	Records in the string of mgrec_string are left.
BS_ENDDATA	Records in the string of mgrec_string are not left.

mgrec_status

This allows PowerBSORT to notify input request of the record.
This member is set when the bsrtmrge function is used. Either of the following is set.

Define value	Meaning
BS_REQ	This shows input request of the record is set. The record of the string that mgrec_string shows is passed in the following bsrtmrge function.
BS_NOREQ	There is no input request of the record. Set NULL for rec_addr and set 0 for rec_len, and issue the bsrtmrge function again.

mgrec_string

This sets the string number of record to be passed to PowerBSORT.
When the first record is passed in the bsrtmrge function set to 0. PowerBSORT sets the string number after the first call and should be let unaltered. User application should pass the string number back to PowerBSORT on subsequent calls to the bsrtmrge function. This member is set when the bsrtmrge function is used.

mgrec_addr

This sets address of the record to be returned to the user application.
NULL is set when there is no record to be returned. This member is set when the bsrtmrge function is used.

mgrec_len

This sets the length of the record to be returned to the user application.
0 is set when there is no record to be returned. This member is set when the bsrtmrge function is used.

hProcFile

This is an unsupported member.

sub_error_code

This is an unsupported member.

6.5.3 BSRTKEY structure

The BSRTKEY structure is a structure to define key field in the sort processing or the merging processing.
When the specification of the key field is omitted, the entire record is considered to be a key field.
The size of the area when the area of BSRTKEY structure is secured can be calculated by using BS_KEYTABSIZE(n). BS_KEYTABSIZE(n) is a macro to calculate the size of BSRTKEY structure with 'n' occurrences of BSKEY structure.

Note

The key field is not omissible for text file CSV format and text file TSV format.

```
typedef struct {
    unsigned long    entry_no;
    BSKEY            key_entry[1];
} BSRTKEY;
```

BSRTKEY structure member

In the following, it explains about the member of the BSRTKEY structure.

entry_no

This specifies the number of the key field (BSKEY structure).
When 0 is specified, this means that the key field is omitted.

key_entry[n]

This specifies the key field.
For more information, refer to BSKEY structure.

6.5.4 BSKEY structure

The BSKEY structure is a structure to define an individual key field.

```
typedef struct {
    unsigned long    key_position;
    unsigned long    key_length;
    unsigned char    key_type;
    unsigned char    key_order;
    unsigned char    key_option;
    unsigned char    key_option2;
    BSPTR_VOID       subfield_addr;
} BSKEY;
```

BSKEY structure member

In the following, it explains about the member of the BSKEY structure.

key_position

This specifies the position of the key field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

key_length

This specifies the length of the key field in the number of bytes.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.
To specify an unsigned binary number (BSKEY_BIT) for a data format (key_type), specify mask value for this member. The logical products of the field value and mask value are considered to be the key value.

The length of the key field does not contain double quotation marks (") that enclose the field for text file CSV format and text file TSV format. When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

```
Example: Key field and length in text file CSV format.
```

```
Key field        Characters effective as key field        Length of key field
ABC              ABC                                      3bytes
"ABC"            ABC                                      3bytes
"A""B""C"        A"B"C                                    5bytes
"A,B,C"          A,B,C                                    5bytes
```

key_type

This specifies the data format of the key field.
For more information, refer to the Data formats that can be specified in the key field.

key_order

This specifies sorting order of the key field.
Specify either of the following.

Define value	Meaning
BS_ASCND	This shows ascending order is set. **(Default value)**
BS_REVRS	This shows descending order is set.

key_option

This specifies operation of the key field.
It is effective only for the text file. Specify the following according to the necessity. When selecting two or more values specify the logical sum.

Define value	Meaning
BSOPT_BLANK	Blanks and tabs in the head of the key field are disregarded.
BSOPT_DICTIONARY	Only the blank and the alphanumeric character are compared.
BSOPT_IGNORE	Control character codes are disregarded.
BSOPT_JUMBO	Lowercase letters are compared as uppercase letters.
BSOPT_NUMERIC	Character strings of the numbers that contains signs are compared with arithmetic value. The result is not guaranteed when characters other than the numbers exist in the character string. When the data format is ASCII code, EBCDIC code or Unicode, it can be specified. It is in the exclusive relationship with BSOPT_CHARNUM and the BSOPT2_LOCALE of key_option2.
BSOPT_CHARNUM	Connected data of alphabet and numbers (for instance, data123) are evaluated and sorted separately for the alphabet and the numerical value. After only alphabets are targeted for the comparison and compared, character strings exclusively with numbers are compared with arithmetic value. Data is evaluated from the left of the specified field, and the data that appeared after effective data is disregarded. It can be specified when the data format is ASCII code, EBCDIC code or Unicode. It is in the exclusive relationship with BSOPT_NUMERIC and the BSOPT2_LOCALE of key_option2.

- If modify collation sequence (BSKEY_COL) is specified as the data format of the key field, key_option cannot be specified.
- When Character form two digit years is specified for a data format of the key field, the specification of the key option is ignored.

key_option2

This specifies operation of the key field.
This member is in the exclusive relationship with BSOPT_NUMERIC of key_option and BSOPT_CHARNUM of key_option. Specify the following according to the necessity.

Define value	Meaning
BSOPT2_LOCALE	Compares according to the collating sequence defined in LC_COLLATE environmental variable. Only when the data format is ASCII code, EUC, EUC 2 byte process codes, EUC 4 byte process codes or Unicode and the input code system (cdmode of BSRTPRIM structure) is ASCII code system or Unicode system, this option can be specified.

Note

- If modify collation sequence (BSKEY_COL) is specified as the data format of the key field, key_option2 cannot be specified.
- When character form two digit years, packed decimal form two digit years, external decimal form two digit years and decimal form two digit years are specified for a data form of the key field, the specification of the key option2 is ignored.

subfield_addr

This specifies the modify collation sequence function.

- When using the modify collation sequence function, specify the address of the BSCOL structure that defined the modify collation sequence. The modify collation sequence function can be used when BSKEY_COL is specified for key_type of BSKEY structure.

Information

For the binary file variable-length record form or the text file, when the record where the key field does not exist is input, the value of the part where the key field does not exist is processed as 0.

Example 1) binary file variable-length record form or the text file fixed field:

```
key field:6.4asca

  0123456789 : Record where key field exists
  012345     : Record where key field doesn't exist
```

Example 2) the text file floating field, the text file CSV format or the text file TSV format:

```
key field:2.1asca

  fld0,fld1,fld2,fld3 : Record where key field exists
  fld0,fld1           : Record where key field doesn't exist
  fld0,fld1,,fld3     : Record where key field doesn't exist
```

6.5.5 BSCOL structure

The BSCOL structure is a structure for defining the modify collation sequence function.

```
typedef struct {
    unsigned char    col_char[256];
} BSCOL;
```

BSCOL structure member

In the following, it explains about the member of the BSCOL structure.

col_char[256]

This specifies the 256-byte table for the modify collation sequence.
Set the code number to be changed in the position of the code number of this table's offset value. Note that the whole of the 256 bytes must be set in the code. Set code that is not to be changed in the same code as the offset value.

6.5.6 BSRTFILE structure

The BSRTFILE structure specifies information in input file, output file, message file and temporary file.
When all file information to be specified in BSRTFILE structure is omitted, NULL is specified for the parameter file of the bsrtopen function. To specify file information, execute it after securing an area.

```
typedef struct {
    BSPTR_BSFILE        inpfile_tbl;
    BSPTR_UCHAR         outfile_addr;
    BSPTR_BSFILE        tmpfile_tbl;
    BSPTR_UCHAR         msgfile_addr;
    BSPTR_BSFSYS        inpfsys_tbl;
    unsigned char       outfsys;
    unsigned char       recovery;
    unsigned char       reserve[2];    /* reserved */
    BSPTR_BSIDX         idx_addr;
    BSPTR_BSFILE_EXT    outfile_tbl;
    BSPTR_VOID          reserve2;      /* reserved */
} BSRTFILE;
```

BSRTFILE structure member

In the following, it explains about the member of the BSRTFILE structure.

inpfile_tbl

To specify input file path name, set address of the BSFILE structure to which input file path name is set.
To input from the standard input or hand over the record from user application (bsrtput function or bsrtmrge function), set NULL.

outfile_addr

To specify output file path name, set the address of the file path name.
The file path name is a character string that ends with '\0'. To specify two or more output files, set them in outfile_tbl of BSRTFILE structure. To output to the standard output or return the record to user application (bsrtget function or bsrtmrge function) or specify two or more output files, set NULL.

tmpfile_tbl

To specify temporary file path name, set the address of BSFILE structure to which temporary file path name is set.
When temporary file path name is omitted, set NULL. In this case, PowerBSORT decides the directory that makes the temporary file according to the priority level. Refer to "Priority level of the directory that creates the temporary file" for the priority level.

msgfile_addr

To specify message file path name, set the address of the file path name.
The file path name is a character string that ends with '\0'. When message file path name is omitted, set NULL. In this case, the message is output to the specified file if BSORT_MSGFILE of startup file is specified.

inpfsys_tbl

This sets address of BSFSYS structure.

When the input file path name is not specified or when only the standard file system of the system is used, specification can be omitted.

outfsys

This sets file system of output file.
For information about the file system, refer to BSFSYS structure. When the output file path name is not specified or when the standard file system of the system is used, specification can be omitted.

recovery

This sets recovery processing when the output file abnormally terminates.
When the recovery processing is set, two or more output files are specified. The output processing ends without opening another output file if the output file abnormally terminates when this specification does not exist. The recovery processing can be specified at the following files.

- System standard file (binary file or text file)
- NetCOBOL file uses the high speed access library for a NetCOBOL sequential file (libbscblfast64.so)
- NetCOBOL file uses the high speed access library for a NetCOBOL physical sequential file (libbscblpsfast64.so)

When the recovery processing is specified, specify the following.

Define value	Meaning
BSRC_FILE	Recovery processing is executed.

idx_addr

This sets address of BSIDX structure.
It is effective for COBOL file system. It cannot be omitted for the indexed file of COBOL file system.

outfile_tbl

This sets address of BSFILE_EXT structure.
Information on the output file for the recovery processing is set.

6.5.7 BSFILE structure

The BSFILE structure is a structure to specify input file path name and temporary file path name.
The size of the area when the area of BSFILE structure is secured can be calculated by using BS_FILESIZE(n). BS_FILESIZE(n) is a macro to calculate the size of BSFILE structure with n pieces of file path name pointer.

```
typedef struct {
    unsigned long    entry_no;
    BSPTR_UCHAR      file_addr[1];
} BSFILE;
```

BSFILE structure member

In the following, it explains about the member of the BSFILE structure.

entry_no

This specifies the number of input file path names or temporary file path names.
When 0 is specified, this means that the input file path name or temporary file path name is omitted. When the input file path name is specified in the merge option, the same value should be specified in the number of input file path name and input_string of BSRTPRIM structure.

file_addr[n]

This specifies the address of input file path name or temporary file path name.
The file path name is a character string that ends with '\0'.

6.5.8 BSFSYS structure

The BSFSYS structure is a structure to specify the input file system.
The size of the area when the area of BSFSYS structure is secured can be calculated by using BS_INPFSYSSIZE(n). BS_INPFSYSSIZE(n) is a macro to calculate the size of BSFSYS structure with n pieces of input files system information.

```
typedef struct {
    unsigned long    entry_no;
    unsigned char    filesys[1];
} BSFSYS;
```

BSFSYS structure member

In the following, it explains about the member of the BSFSYS structure.

entry_no

This specifies the number of input file systems.

filesys[n]

This specifies the file system of input files.
A specification of input file system sets 1 input file as a file system of a file specified for file_addr[i] of BSFILE structure should be set in filesys[i]. When the number of input file systems is less than the number of input file path names, the file system specified at the end is applied to the file that remains. The file system that can be specified is shown below. When BSFS_COB1, BSFS_COB2, and BSFS_COB3 are specified for an definition value, it is necessary to define BSORT_FILESYS_fs of a startup file to associate the definition value with an actual file system. NetCOBOL file system can be specified for COBOL file system.

Define value	File system
BSFS_UFS	Native file system of the system **(Default)**
BSFS_COBS64	NetCOBOL file system (sequential file)
BSFS_COBP64	NetCOBOL file system (physical sequential file)
BSFS_COBR64	NetCOBOL file system (relative file)
BSFS_COBI64	NetCOBOL file system (indexed file)
BSFS_COB1	COBOL file system
BSFS_COB2	COBOL file system
BSFS_COB3	COBOL file system

6.5.9 BSIDX structure

The BSIDX structure is a structure to create the index when the output file is NetCOBOL indexed file.
The size of the area when the area of BSIDX structure is secured can be calculated by using BS_IDXTABSIZE(n). BS_IDXTABSIZE(n) is a macro to calculate the size of BSIDX structure with n pieces of index information.

```
typedef struct {
    unsigned char    idx_flag;
    unsigned char    reserve[3];    /* reserved */
    unsigned long    entry_no;
    BSIDXKEY         idx_entry[1];
} BSIDX;
```

BSIDX structure member

In the following, it explains about the member of the BSIDX structure.

idx_flag

This specifies the method for creating indexes.
Specify the following. When selecting two or more values specify the logical sum.

Define value	Meaning
BSIF_COMP	The index is a compressed key.
BSIF_CMPR	The record is a compressed data.

entry_no

This specifies number of indexes (BSIDXKEY structure).
The maximum value complies with the maximum value of the COBOL file system.

idx_entry[n]

This specifies the index field.
For more information, refer to BSIDXKEY structure.

6.5.10 BSIDXKEY structure

The BSIDXKEY structure is a structure to specify the index field when output file is the NetCOBOL indexed file.

```
typedef struct {
    unsigned long    idx_position;
    unsigned long    idx_length;
    unsigned char    idx_type;
    unsigned char    idx_order;
    unsigned char    idx_msub;
    unsigned char    idx_comp;
} BSIDXKEY;
```

BSIDXKEY structure member

In the following, it explains about the member of the BSIDXKEY structure.

idx_position

This specifies the position of the index field in a byte position.

idx_length

This specifies the length of the index field in byte.

idx_type

This specifies the data format of the index field.
The data formats and the lengths of the index field are shown as follows.

Type	Data format		Define value	Length (byte)	Meaning
Character	ASCII code (Note 1)		BSKEY_ASC	1-254	CHARTYPE of NetCOBOL
	Unicode UCS-2 form (Note 2) (Byte order of system standard)		BSKEY_UCS2	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL
		Big endian	BSKEY_UCS2B	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL
		Little endian	BSKEY_UCS2L	2-254 (Multiples of 2)	CHARTYPE of NetCOBOL

Type	Data format		Define value	Length (byte)	Meaning
	Unicode UTF-32 form (Note 2) (Byte order of system standard)		BSKEY_UTF32	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
		Big endian	BSKEY_UTF32B	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
		Little endian	BSKEY_UTF32L	4-252 (Multiples of 4)	CHARTYPE of NetCOBOL
	Unicode UTF-8 form (Note 2)		BSKEY_UTF8	1～254	CHARTYPE of NetCOBOL

NOTES

1. Can be specified if input code system (cdmode of BSRTPRIM structure) is other than the EBCDIC.
2. Can be specified if input code system (cdmode of BSRTPRIM structure) is other than the EBCDIC, and C or UTF-8 locale is defined in the LANG environment variable.

idx_order

This specifies sorting order of index field.
The ascending order is specified for the NetCOBOL indexed file.

Define value	Sorting order
BS_ASCND	Ascending order

idx_msub

This specifies method for creating index fields.
Specify the following according to the necessity. Specify BS_ISEGEND at the end of a main-key and the sub-key.

Define value	Meaning
BSIF_DUPS	This shows duplicate key specification is set in index. It is in an exclusive relation with BSIF_NDUP.
BSIF_NDUP	This shows unique key specification is set in index. It is in an exclusive relation with BSIF_DUPS.
BS_IMAIN	This shows main-key is specified. It is in an exclusive relation with BS_ISUB.
BS_ISUB	This shows sub-key is specified. It is in an exclusive relation with BS_IMAIN.
BS_ISEGEND	This shows end of the main-key and the sub-key.

idx_comp

This is an unsupported member.
Set '\0' for this version.

6.5.11 BSFILE_EXT structure

The BSFILE_EXT structure is a structure to specify two or more output files.
Files can be divided and output to two or more files as a recovery processing by specifying two or more output files when the output file abnormally terminates. Therefore, set on another file system when two or more files are specified. The recovery processing can be specified at the following files.

- System standard file (binary file or text file)

- NetCOBOL file uses the high speed access library for a NetCOBOL sequential file (libbscblfast64.so)
- NetCOBOL file uses the high speed access library for a NetCOBOL physical sequential file (libbscblpsfast64.so)

The size of the area when the area of BSFILE_EXT structure is secured can be calculated by using BS_OUTTABSIZE(n). BS_OUTTABSIZE(n) is a macro to calculate the size of BSFILE_EXT structure with n pieces of information on output file.

```
typedef struct {
    unsigned long            entry_no;
    union {
        BSPTR_BSFILE_BASE    ufs;
        BSPTR_BSFILE_BASE    text;
        BSPTR_BSFILE_BASE    cob;
        BSPTR_BSFILE_BASE    cobr;
        BSPTR_BSFILE_INDEX   cobi;
        BSPTR_BSFILE_INDEX   cisam;
        BSPTR_BSFILE_INDEX   rdm;
        BSPTR_BSFILE_BASE    dosfs;
        BSPTR_BSFILE_BASE    ntfs;
        BSPTR_BSFILE_BTRV    btrieve;
        BSPTR_BSFILE_BASE    sfs;
        BSPTR_BSFILE_BASE    mtfs;
        BSPTR_BSFILE_BASE    mtfr;
        BSPTR_BSFILE_BASE    mtfi;
    } outfile_opt[1];
} BSFILE_EXT;
```

BSFILE_EXT structure member

In the following, it explains about the member of the BSFILE_EXT structure.

entry_no

This specifies number of output file path names.
It is considered that the specification of the output file is omitted when 0 is set.

Member that sets file extension information

These members are defined in a union.

outfile_opt[n].ufs

This specifies the address of BSFILE_BASE structure in binary file of the standard file system of the system.
Specify the number of addresses specified in entry_no.

outfile_opt[n].text

This specifies the address of BSFILE_BASE structure in text file of the standard file system of the system.
Specify the number of addresses specified in entry_no.

outfile_opt[n].cob

This specifies the address of BSFILE_BASE structure in NetCOBOL sequential file or NetCOBOL physical sequential file.
Specify the number of addresses specified in entry_no.

outfile_opt[n].cobr

This is an unsupported member.

outfile_opt[n].cobi

This is an unsupported member.

outfile_opt[n].cisam

This is an unsupported member.

outfile_opt[n].rdm

This is an unsupported member.

outfile_opt[n].dosfs

This is an unsupported member.

outfile_opt[n].ntfs

This is an unsupported member.

outfile_opt[n].btrieve

This is an unsupported member.

outfile_opt[n].sfs

This is an unsupported member.

outfile_opt[n].mtfs

This is an unsupported member.

outfile_opt[n].mtfr

This is an unsupported member.

outfile_opt[n].mtfi

This is an unsupported member.

6.5.12 BSFILE_BASE structure

The BSFILE_BASE structure is a structure to specify specific file information to each output file.

```
typedef struct {
    BSPTR_UCHAR         outfile_addr;
    unsigned char       file_kind;
    unsigned char       reserve[3];    /* reserved */
    BSPTR_BSOUT_COND    out_cond;
} BSFILE_BASE;
```

BSFILE_BASE structure member

In the following, it explains about the member of the BSFILE_BASE structure.

outfile_addr

This specifies the address of output file path name.
The file path name is a character string that ends with '\0'.

file_kind

This specifies the file system of output file.
Two or more file systems cannot be specified. For information about the file systems, refer to BSFSYS structure.

out_cond

This is an unsupported member. Do not set anything.

6.5.13 BSRTOPT structure

The BSRTOPT structure is a structure to specify the summation field, the selection field, the reconstruction field, and the number of skipping records.

```
typedef struct {
    BSPTR_BSRTSUM     sum_addr;
    BSPTR_BSRTSELE    sele_addr;
    BSPTR_BSRTRCON    rcon_addr;
    BSPTR_BSRTSKIP    skip_addr;
} BSRTOPT;
```

BSRTOPT structure member

In the following, it explains about the member of the BSRTOPT structure.

sum_addr

This specifies the address of BSRTSUM structure.
BSRTSUM structure defines summation field. When BS_SUM is specified for optionfunc of BSRTPRIM structure, BSRTSUM structure is effective. When BS_SUM is specified for optionfunc of BSRTPRIM structure, and the summation field is omitted, it will make an error.

sele_addr

This specifies the address ofBSRTSELE structure.
BSRTSELE structure defines selection field. When BS_SELECT is specified for optionfunc of BSRTPRIM structure, BSRTSELE structure is effective. When BS_SELECT is specified for optionfunc of BSRTPRIM structure, and the selection field is omitted, it will make an error.

rcon_addr

This specifies the address of BSRTRCON structure.
BSRTRCON structure defines reconstruction field. When BS_RECON is specified for optionfunc of BSRTPRIM structure, BSRTRCON structure is effective. When BS_RECON is specified for optionfunc of BSRTPRIM structure, and reconstruction field is omitted, it will make an error.

skip_addr

This specifies the address of BSRTSKIP structure.
BSRTSKIP structure defines the skipped number of records. When the input file name is specified in BSRTFILE structure, BSRTSKIP structure is effective.

6.5.14 BSRTSUM structure

The BSRTSUM structure is a structure to define the summation field when the record summation option is used.
When the values of the specified key fields are equal, add the summation fields to make 1 record. In the case of text files, only the numbers described in ASCII code and Unicode are to be processed. This option is effective in the sort option and the merge option. The record summation option has an exclusive relationship with the suppression option. When specified at the same time as the first-in first-out (FIFO) option, the specification of the first-in first-out (FIFO) option is disregarded.
The size of the area when the area of BSRTSUM structure is secured can be calculated by using BS_SUMTABSIZE(n). BS_SUMTABSIZE(n) is a macro to calculate the size of n pieces of BSRTSUM structures.

Information

If an overflow occurs during record summation, short records (excluding the summation field) are input, or an invalid code is found in the summation field during text record summation, the return value of the bsrtopen function is zero, and BSERR_OVERFLOW, BSERR_SUMSHRTREC, or BSERR_SUMFIELD is set in the errdetail of the BSRTREC structure.

```
typedef struct {
    unsigned long    entry_no;
    BSSUM            sum_entry[1];
} BSRTSUM;
```

BSRTSUM structure member

In the following, it explains about the member of the BSRTSUM structure.

entry_no

This specifies the number of the summation field (BSSUM structure).
When 0 is specified, this means that the summation field is omitted.

sum_entry[n]

This specifies the summation field.
For more information, refer to BSSUM structure.

6.5.15 BSSUM structure

The BSSUM structure is a structure to define an individual summation field.

```
typedef struct {
    unsigned long    sum_position;
    unsigned long    sum_length;
    unsigned char    sum_type;
    unsigned char    sum_resultsign;
    unsigned char    sum_resultnumber;
    unsigned char    reserve;            /* reserved */
} BSSUM;
```

BSSUM structure member

In the following, it explains about the member of the BSSUM structure.

sum_position

This specifies the position of the summation field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

sum_length

This specifies the length of the summation field in the number of bytes.
For fields that are longer than the specified field, process the text file floating field, the text file CSV format, and the text file TSV format with the specified field length. When a field that is shorter than the specified field length appears, it enhances to the specified field length.

Note

In the text file CSV format and text file TSV format specification, the length of the summation field does not contain the double quotation marks (") that enclose the field.

Example: Summation field and length in text file CSV format.

```
Summation field    Characters effective as    Length of summation field
                   summation field
123                123                        3bytes
"123"              123                        3bytes
```

sum_type

This specifies the data format of the summation field.
For more information, refer to the Data formats that can be specified in the summation field.

sum_resultsign

This specifies whether or not signs are used for text files.
Specify one of the following, as appropriate.

Define value	Meaning
BS_SIGNED	Appends a sign to the value in the summation field. This value and the BS_PLUSUNSIGNED are mutually exclusive.
BS_PLUSUNSIGNED	Appends a minus sign (-) to the value in the summation field if that value is negative. This value and the BS_SIGNED are mutually exclusive.

Information

If omitted, behavior is determined by the input data, as described below.

- If the summation result is a negative value, a minus sign (-) is included in the summation result.
- If the summation result of unsigned data and signed data is a positive value, no sign is included in the summation result.
- If two unsigned data items are summated, no sign is included in the summation result.
- If signed and unsigned data are summated and the result is zero, no sign is included.
- If two signed data items are summated, a sign is included in the summation result. If the summation result is zero, a plus sign (+) is included.

sum_resultnumber

This specifies the behavior when the length of the value in the summation field is less than the field length for text files.
Specify one of the following, as appropriate.

Define value	Meaning
BS_ZEROPADDING	If the length of the value in the summation field is less than the field length, the area to the left of the value is padded with zeroes (0). For example, if the value in the summation field is "1234", and the summation field length is six bytes, the value is recorded as "001234". This value, BS_BLANKPADDING, and BS_ZBDELETE are mutually exclusive.
BS_BLANKPADDING	If the length of the value in the summation field is less than the field length, the area to the left of the value is padded with spaces. For example, if the value in the summation field is "1234", and the summation field length is six bytes, the value is recorded as " 1234". This value, BS_ZEROPADDING, and BS_ZBDELETE are mutually exclusive.
BS_ZBDELETE	Deletes any spaces, tabs, or zeroes that appear at the head of the summation field. If there are any spaces, tabs, or zeroes at the head of the summation field, the field is evaluated from the left and any spaces, tabs, or zeroes that appear before the first number (other than zero) are deleted. For example, if the value in the summation field is "00123", it is changed to "123". Note that, if the value in the summation field is equivalent to zero (such as "0", "0000" or "+000"), the last "0" is not deleted. This option can be specified for the text file floating field, text file CSV format, and text file TSV format. This value, BS_ZEROPADDING, and BS_BLANKPADDING are mutually exclusive.

Information

If omitted, behavior is determined by the input data, as described below.

- When the summation field is summated under the conditions listed below, if the length of the result in the summation field is less than the field length, the area to the left of the value is padded with zeroes (0).
  - When both summated data items are padded to the left with zeroes
  - When one summated data item is padded to the left with zeroes and the other is padded to the left with spaces
  - When one summated data item is padded to the left with zeroes and the other is padded to the left with tabs
- When the summation field is summated under the conditions listed below, if the length of the result in the summation field is less than the field length, the area to the left of the value is padded with spaces.
  - When both summated data items are padded to the left with spaces
  - When both summated data items are padded to the left with tabs
  - When one summated data item is padded to the left with spaces and the other is padded to the left with tabs

Note

- Of the records that are objects of the summation process, it is not possible to predict which record is output with the summation results.
- Specify so that the summation field does not overlap a key field or another summation field.
- Note also that the summation field must be completely included in the record.
- When using the record summation feature, the key field specification cannot be omitted.
- If an overflow occurs during addition of a summation field, subsequent behavior is determined by the specification of the BSORT_SUMOVERCONT of startup file.
- Numbers with decimal points cannot be summated.
- For text files, only single-byte numbers in ASCII code, Unicode UCS-2 format, Unicode UTF-32 format, or Unicode UTF-8 format are processed.
- With a text file floating field specification, if the summation field contains a field separation character and this causes the summation field position to change, correct processing cannot be guaranteed.
- With a text file floating field specification, summation results are processed to the specified field length. When a field that is longer than the specified field length appears, the part that exceeds the specified field length is output without changing the content. If the field is shorter than the specified field length, it is extended to the specified field length, and then processed.

  Example: Specify "1.5asc" for the summation field. Specify "0.3asca" for the key field. The field separation character string is a comma (,).

```
Input record                  Output record
001,12345ABC,OPQ (Note1)      001,12456ABC,OPQ (Note2)
001,111,RST                   002,00127,UVW (Note3)
002,15,UVW (Note1)
002,00112DEF,XYZ

NOTE
1. Assume that the record shown here is the one output by the record summation option.
2. Part ("ABC") that exceeds the specified field length is output without changing the content.
3. When the field is shorter than the specified field length, it is extended to the specified field
length.
```

- In the text file CSV format and text file TSV format, summation results are processed to the specified field length. When a field is longer than the specified field length, the part that exceeds the specified field length is not output. If the field is shorter than the specified field length, it is extended to the specified field length, and then processed.

  Example: Specify "1.5asc" for the summation field of the text file CSV format. Specify "0.3asca" for the key field.

```
Input record                  Output record
001,12345ABC,OPQ (Note1)      001,12468,OPQ (Note2)
001,123,RST                   002,00027,UVW (Note3)
002,15,UVW (Note1)
002,00012DEF,XYZ

NOTE
1. Assume that the record shown here is the one output by the record summation option.
2. Part ("ABC") that exceeds the specified field length is not output.
3. When the field is shorter than the specified field length, it is extended to the specified field
length.
```

- In the text file CSV format and the text file TSV format, whether the summation field is enclosed with double quotation marks (") or it doesn't enclose it is decided depending on the following conditions. When the field on the input record of the target for the output is enclosed with double quotation marks ("), the field after the record is summation is enclosed with double quotation marks ("). When the field on the input record of the target for the output is not enclosed with double quotation marks ("), the field after the record is summation is not enclosed with double quotation marks (").

  Example: Specify "1.5asc" for the summation field of the text file CSV format. Specify "0.3asca" for the key field.

```
Input record                  Output record
001,"12345",OPQ (Note)        001,"12468",OPQ
001,"123",RST                 002,"00027",UVW
002,"15",UVW (Note)           003,11900,GHI
002,00012,XYZ                 004,98769,MNO
003,11111,GHI (Note)
003,"00789",JKL
004,98765,MNO (Note)
004,4,PQR

NOTE
Assume that the record shown here is the one output by the record summation option.
```

- In the text file floating field specification, when a field that is shorter than the specified field length appears, it enhances to the specified field length. As a result, when the record length exceeds the specified maximum record length, it processes as an overflow.

6.5.16 BSRTSELE structure

The BSRTSELE structure is a structure to define the selection field when the record selection option is used.
The size of the area when the area of BSRTSELE structure is secured can be calculated by using BS_SELETABSIZE(n). BS_SELETABSIZE(n) is a macro to calculate the size of n pieces of BSRTSELE structures.

```
typedef struct {
    unsigned long    entry_no;
    BSSELE           sele_entry[1];
} BSRTSELE;
```

BSRTSELE structure member

In the following, it explains about the member of the BSRTSELE structure.

entry_no

This specifies the number of the selection field (BSSELE structure).
When 0 is specified, this means that the selection field is omitted.

sele_entry[n]

This specifies the selection field.
For more information, refer to BSSELE structure.

6.5.17 BSSELE structure

The BSSELE structure is a structure to define an individual selection field.

```
typedef struct {
    unsigned char        sele_cmpoprat;
    unsigned char        sele_type1;
    unsigned char        sele_type2;
    unsigned char        sele_option;
    unsigned long        sele_position1;
    unsigned long        sele_length1;
    union {
        unsigned long    sele_position2;
        BSPTR_UCHAR      sele_literal;
    } pos2;
    unsigned long        sele_length2;
    unsigned char        sele_trueop;
    unsigned char        sele_falseop;
    unsigned char        sele_mask;
    unsigned char        sele_option2;
    unsigned long        sele_truejmp;
    unsigned long        sele_falsejmp;
    BSPTR_VOID           subfield_addr;
} BSSELE;
```

BSSELE structure member

In the following, it explains about the member of the BSSELE structure.

sele_cmpoprat

This specifies the comparison operator.
The following table shows comparison operators and their true conditions.

Comparison operation	Meaning(True condition)
BSCOND_EQ	Compared field = Comparing field or literal value
BSCOND_NE	Compared field != Comparing field or literal value
BSCOND_GT	Compared field > Comparing field or literal value
BSCOND_GE	Compared field >= Comparing field or literal value
BSCOND_LT	Compared field < Comparing field or literal value
BSCOND_LE	Compared field <= Comparing field or literal value

When the compared field is EUC, EUC 2 byte process codes and EUC 4 byte process codes, the data is compared in the collating sequence according to the environment variable LC_COLLATE.

sele_type1

This specifies the data format of the compared field.
For more information, refer to the Data formats that can be specified in the compared field and comparing field.

sele_type2

This specifies the data format of the comparing field.
For more information, refer to the Data formats that can be specified in the compared field and comparing field. When a literal value is specified for the comparing field, BS_LITERAL is specified for sele_type2.

- When the data format of the compared field and comparing field are different, it compares the data after adjusting them to the data format of the compared field.
- When the data format of the compared field is character, it compares data in the shorter length of the compared field and comparing field.
- When the data format of the compared field is numeric or number, the shorter field of the compared field and comparing field are compared with the longer one after moving to the right end.

sele_option

This specifies operation of compared field.
It is effective only for text files. Specify the following according to the necessity. When selecting two or more values specify the logical sum.

Define value	Meaning
BSOPT_BLANK	Blanks and tabs in the head of the field are disregarded.
BSOPT_DICTIONARY	Only the blank and the alphanumeric character are compared.
BSOPT_IGNORE	Control character codes are disregarded.
BSOPT_JUMBO	Lowercase letters are compared as uppercase letters.
BSOPT_NUMERIC	Character strings of the numbers that contains signs are compared with arithmetic value. The result is not guaranteed when characters other than the numbers exist in the character string. When the data format is ASCII code or Unicode, it can be specified. It is in the exclusive relationship with BSOPT2_LOCALE of sele_option2.

If modify collation sequence (BSKEY_COL) is specified as the data format of the compared field, compared field operations cannot be specified.

sele_position1

This specifies the position of the compared field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.

sele_length1

This specifies the length of the compared field.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.

Note

In text file CSV format and text file TSV format specification, the length of the compared field does not contain double quotation marks (") that enclose the field. When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

```
Example: Compared field and length in text file CSV format
```

```
Compared field    Characters effective as compared field    Length of compared field
ABC               ABC                                      3bytes
"ABC"             ABC                                      3bytes
"A""B""C"         A"B"C                                    5bytes
"A,B,C"           A,B,C                                    5bytes
```

pos2.sele_position2

This specifies the position of the comparing field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0.
pos2.sele_literal is defined in a union. When a literal value is specified, set in pos2.sele_literal.

pos2.sele_literal

This specifies the address of a literal value.
pos2.sele_position2 is defined in a union. When specifying other values than literal value, set in pos2.sele_position2.

Note

- When you specify a text file fixed field, literal values cannot contain record separation character.
- When you specify a text file floating field, literal values cannot contain field separation character or record separation character.
- For text file CSV format and text file TSV format, double quotation marks (") do not need to enclose literal values.

sele_length2

This specifies the length of the comparing field or a literal value.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed.

Note

In text file CSV format and text file TSV format specification, the length of the comparing field does not contain double quotation marks (") that enclose the field. When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

```
Example: Comparing field and length in text file CSV format
```

```
Comparing field    Characters effective as comparing field    Length of comparing field
ABC                ABC                                       3bytes
"ABC"              ABC                                       3bytes
"A""B""C"          A"B"C                                     5bytes
"A,B,C"            A,B,C                                     5bytes
```

sele_trueop

This specifies processing when a comparison result is true.
Specify the definition value that shows the end of the comparison processing in sele_trueop and sele_falseop of the last entry. For information about the processing of the comparison result, refer to the following table.

Table 6.2 Method of processing comparison result

Type of processing	Define value	Meaning
INCLUDE	BSSEL_INCLUDE	This shows the end of the comparison processing. And, it is shown to process the record.
OMIT	BSSEL_OMIT	This shows the end of the comparison processing. And, it is shown not to process the record.
Processing next field	BSSEL_NEXT	This shows to move to the next comparison processing.
Skipping to specified selection field	BSSEL_JUMP	This shows to move to the specified comparison processing.

sele_falseop

This specifies processing when a comparison result is false.
Specify the definition value that shows the end of the comparison processing in sele_trueop and sele_falseop of the last entry. For information about the processing of the comparison result, refer to the Method of processing comparison result.

sele_mask

This is an unsupported member. Do not set anything.

sele_option2

This specifies operation of the compared field.
This member is in the exclusive relationship with BSOPT_NUMERIC of sele_option. Specify the following according to the necessity.

定義値	意味
BSOPT2_LOCALE	Compares according to the collating sequence defined in LC_COLLATE environmental variable. Only when the data format is ASCII code, EUC, EUC 2 byte process codes, EUC 4 byte process codes or Unicode and the input code system (cdmode of BSRTPRIM structure) is ASCII code system or Unicode system, this option can be specified.

If modify collation sequence (BSKEY_COL) is specified as the data format of the compared field, sele_option2 cannot be specified.

sele_truejmp

This specifies the jumping destination when BSSEL_JUMP is specified for sele_trueop.
If i is specified for sele_truejmp, processing moves to the selection field of sele_entry[i] when the comparison result is a true. The value within the range of entry number from +1 to entry_no-1 is set in sele_truejmp.

sele_falsejmp

This specifies the jumping destination when BSSEL_JUMP is specified for sele_falseop.
If i is specified for sele_falsejmp, processing moves to the selection field of sele_entry[i] when the comparison result is a false. The value within the range of entry number from +1 to entry_no-1 is set in sele_falsejmp.

subfield_addr

When using the modify collation sequence function, this specifies the address of the BSCOL structure that defined the modify collation sequence.
The modify collation sequence function can be used when BSKEY_COL is specified in sele_type1 or sele_type2.

Information

One selection field expresses one logical expression. Therefore, the compound condition can be specified by specifying two or more selection fields. The specification method when the record where it meets the compound condition is selected is shown as follows.

Compound condition

```
(Logical expression 1 AND Logical expression 2) OR Logical expression 3
    AND:Logical product OR:Logical sum
```

Specification method

```
entry_no = 3
[0] Logical expression 1  true BSSEL_NEXT     false BSSEL_JUMP 2
[1] Logical expression 2  true BSSEL_INCLUDE  false BSSEL_NEXT
[2] Logical expression 3  true BSSEL_INCLUDE  false BSSEL_OMIT
```

6.5.18 BSRTRCON structure

The BSRTRCON structure is a structure to define the reconstruction field when the record reconstruction option is used.
The size of the area when the area of BSRTRCON structure is secured can be calculated by using BS_RCONTABSIZE(n). BS_RCONTABSIZE(n) is a macro to calculate the size of n pieces of BSRTRCON structures.

Note

When the reconstruction field is specified, specify the field positions after the reconstruction for the key field and the summation field.

```
typedef struct {
    unsigned long    entry_no;
    BSRCON           rcon_entry[1];
} BSRTRCON;
```

BSRTRCON structure member

In the following, it explains about the member of the BSRTRCON structure.

entry_no

This specifies the number of the reconstruction field (BSRCON structure).
When 0 is specified, this means that the reconstruction field is omitted.

rcon_entry[n]

This specifies the reconstruction field.
For more information, refer to BSRCON structure.

6.5.19 BSRCON structure

The BSRCON structure is a structure to define an individual reconstruction field.

```
typedef struct {
    union {
        unsigned long    rcon_position;
```

```
        BSPTR_UCHAR      rcon_literal;
    } pos;
    unsigned long        rcon_length;
    unsigned char        rcon_consinf;
    unsigned char        rcon_option;
    unsigned char        reserve[2];        /* reserved */
} BSRCON;
```

BSRCON structure member

In the following, it explains about the member of the BSRCON structure.

pos.rcon_position

This specifies the position of the reconstruction field.
To specify the binary files and text file fixed fields, specify the byte position where the head of the record was assumed to be 0. For the specification of the text file floating field, text file CSV format, and text file TSV format, specify the field number counted from 0. Specify 0 when you specify BS_EMPFLD for rcon_consinf. pos.rcon_literal is defined in a union. When a literal value is specified, set in pos.rcon_literal.

pos.rcon_literal

This specifies the address of a literal value.
pos.rcon_literal is defined in a union. When the field in the input record is specified, set in pos.rcon_position.

- When you specify a text file fixed field, literal values cannot contain record separation character.
- When you specify a text file floating field, literal values cannot contain field separation character or record separation character.
- Output text file CSV format and text file TSV format with literal values enclosed in double quotation marks (") when the literal values contains the field separation character, the record separation character (Note) or double quotation marks ("). In this case, a literal value double quotation mark (") is output when two double quotation marks (") are consecutive.

  Note: Enclose literal values with double quotation marks (") when it is included by not only the record separator of the input file but also one of CRLF, CR, and LF permitted as a record separator.

  Example: The data output to the reconstruction field when literal values contain the field separation character (comma) or the double quotation marks (") is as follows:

```
Specification of literal values    Data output to reconstruction field
FIELD"2"                           "FIELD""2"""
FIELD2,3                           "FIELD2,3"
FIELD"2",3                         "FIELD""2"",3"
```

rcon_length

This specifies the length of the reconstruction field or a literal value.
For the text file floating field specification, text file CSV format, and text file TSV format, process with the length specified here when the actual fields are longer than the specified field length. When the actual fields are shorter than the specified field length, the actual field length is processed. Specify 0 when the BS_RECEND or the BS_EMPFLD is specified in rcon_consinf.

In the text file CSV format and text file TSV format specification, the length of the reconstruction field does not contain double quotation marks (“). When two consecutive double quotation marks ("") are included in the field, they are interpreted as one double quotation mark (").

Example: Reconstruction field and length in text file CSV format.

```
Reconstruction field    Characters effective as    Length of reconstruction field
                        reconstruction field
ABC                     ABC                        3bytes
"ABC"                   ABC                        3bytes
"A""B""C"               A"B"C                      5byte
"A,B,C"                 A,B,C                      5byte
```

rcon_consinf

This specifies whether to set a field in an input record to be a reconstruction field or literal value to be a reconstruction field.
When the text file floating field specification, text file CSV format, and text file TSV format, PowerBSORT adds and reconstructs the field separation character strings.

Define value	Meaning
BS_FIELD	This specifies field in input record for the reconstruction field.
BS_LITERAL	This specifies literal value for reconstruction field.
BS_RECEND	This specifies from the specified position to the record end for the reconstruction field.
BS_EMPFLD	This specifies the empty field for the reconstruction field. BS_EMPFLD can be specified for text file CSV format and text file TSV format.

rcon_option

This specifies the operation of the reconstruction field.
In text file CSV format or text file TSV format, when the literal values or the empty field is treated in the reconstruction field, this operation can be specified. When this specification is omitted, BSFE_L is assumed.

Define value	Meaning
BSFE_A	Enclose the reconstruction field with double quotation marks (").
BSFE_L	Do not enclose the reconstruction field with double quotation marks ("). However, enclose it with double quotation marks (") when a field separation character, a record separation character (Note) or double quotation marks (") is included in the reconstruction field.

NOTE

Enclose literal values with double quotation marks (") when it is included by not only the record separator of the input file but also one of CRLF, CR, and LF permitted as a record separator.

Information

- In text file CSV format and text file TSV format, when format 1 of *recon-def* or format 3 of *recon-def* is specified, enclosing the reconstruction field with double quotation marks (") depends on the condition. When the field on the input record is enclosed with double quotation marks ("), the field after the record is reconstructed is enclosed with double quotation marks (").

  Example: Specify the second field for three bytes and the third field for two bytes for a reconstruction field.

```
Input record                Output record
"001","ABC",60              "ABC",60
"002","ABCDE",50            "ABC",50
"003","AB,CDE",40           "AB,",40
"004","AB"CDE,30 (Note)     "AB"C,30

NOTE
"AB" is enclosed by double quotation marks (") in the second field and "CDE" continues afterwards.
```

6.5.20 BSRTSKIP structure

The BSRTSKIP structure is a structure to define the number of records to be skipped when skipping unnecessary records from the head of the input files before processing.
The size of the area when the area of BSRTSKIP structure is secured can be calculated by using BS_SKIPTABSIZE(n). BS_SKIPTABSIZE(n) is a macro to calculate the size of n pieces of BSRTSKIP structures.

```
typedef struct {
    unsigned long    entry_no;
    unsigned long    skip_recnum[1];
} BSRTSKIP;
```

BSRTSKIP structure member

In the following, it explains about the member of the BSRTSKIP structure.

entry_no

This specifies the number of entries of skipped numbers of records.
When 0 is specified, this means that the specification of the skipped number of records was omitted.

skip_recnum[n]

This specifies the number of records to be skipped.
When 0 is specified, all the records are processed. Specify the number of the records for each input file according to the order of specification of the input files. When the specified number of records to be skipped is less than the input files, all the records in the input files that have not been specified are processed. When the specified number of records to be skipped is more than the number of input files, the specification that exceeds the number of the files are ignored.

6.5.21 BSRTFUNC structure

The BSRTFUNC structure is a structure to which each address of the BSORT function is set.

```
typedef struct {
  int (BSPTR_BSRT bsrtput) (BSPTR_BSRTREC);
  int (BSPTR_BSRT bsrtget) (BSPTR_BSRTREC);
  int (BSPTR_BSRT bsrtmrge) (BSPTR_BSRTREC);
  int (BSPTR_BSRT bsrtinit) (long int, struct _BSRTFUNC BSPTR_BSRT, BSPTR_BSRTPRIM, BSPTR_BSRTREC,
BSPTR_BSRTKEY, BSPTR_BSRTFILE, BSPTR_BSRTOPT);
  int (BSPTR_BSRT bsrtterm) (BSPTR_BSRTPRIM, BSPTR_BSRTREC);
  unsigned long             reserve1;    /* reserved */
  unsigned long             reserve2;    /* reserved */
  BSPTR_VOID                handle;
} BSRTFUNC;
```

BSRTFUNC structure member

In the following, it explains about the member of the BSRTFUNC structure.

bsrtput

The bsrtopen function sets address of bsrtput function. Do not update this member.

bsrtget

The bsrtopen function sets address of bsrtget function. Do not update this member.

bsrtmrge

The bsrtopen function sets address of bsrtmrge function. Do not update this member.

bsrtinit

PowerBSORT uses this member while running. This member need not be referred and updated.

bsrtterm

PowerBSORT uses this member while running. This member need not be referred and updated.

handle

PowerBSORT uses this member while running. This member need not be referred and updated.

Chapter 7 Messages and Error codes

This section explains the output messages of the bsort command, the bsortex command, and the BSORT function, and the error detail codes of the BSORT function.

7.1 Messages

Messages include the information messages, warning messages, and error messages.
The language of the message changes into Japanese and English according to a regional setting of the system.

Output destination of the message

The message is output to the standard error output.
When the messages are output to the file, it is specified by one of the following:

- BSORT_MSGFILE of startup file
- msgfile operand of -option option in bsortex command
- -G option in bsort command
- msgfile_addr of BSRTFILE structure in BSORT function

When the message file is specified, the information messages are output to the message file. The warning messages and the error messages are output to the message file and the standard error output.
Moreover, when E, W or I is specified for BSORT_SYSLOG of startup file, the message corresponding to a specified value is output to the syslog.

When the BSORT function is used, it is necessary to specify BSMSG_STDERR for optionfunc of the BSRTPRIM structure specified by the bsrtopen function to output the message to the standard error output.

Output form of the message

The output form of the message is specified by BSORT_MSGSTYLE of startup file.
When 1 is specified for BSORT_MSGSTYLE of startup file, the message is output in the form of the following.

```
Header: Text
```

```
Header: Header is output. ("bsort")
Text:   Message text is output.
```

When BSORT_MSGSTYLE of startup file specification is omitted or 2 is specified, the message is output in the form of the following.

```
Header: Type: Date: Number Text
```

```
Header: Header is output. ("PSORT")
Type:   Message type is output. ("INFO", "WARNING", or "ERROR")
Date:   Date and time are output. ("YYYY-MM-DD HH:MM:SS")
Number: Message number is output. (Four digits)
Text:   Message text is output.
```

7.1.1 Information messages

Information messages are messages that notify regarding the processing situation and the processing result of PowerBSORT.
When I is specified for the following specification, the error message is output.

- BSORT_MSGLEVEL of startup file
- msglevel operand of -option option in bsortex command

- -l option in bsort command
- msglevel of BSRTPRIM structure in BSORT function

7.1.1.1 Explanation form of information messages

The explanation form of the information messages is shown as follows.

Message number

```
Message text
```

Embedded variable

Embedded variable in the information message is shown as follows.
%s: Meaning of character string shown at position of %s

%llu: Meaning of numeric shown at position of %llu

Note

When two or more embedded variable exists, the serial number is added to %s.

Example) %s1, %s2

Action

The operation of PowerBSORT is shown.

7.1.1.2 Information messages

The information messages are shown as follows.

1044

```
Number of input records is %llu.
```

Embedded variable

%llu: The total number of records loaded from input file or user application.

Action

Process will continue.

1045

```
Number of output records is %llu.
```

Embedded variable

%llu: The total number of records read out to output file or those returned to user application.

Action

Process will continue.

1046

```
Number of suppress records is %llu.
```

Embedded variable

%llu: The number of records ignored during the process of the record selection, the record summation and the suppression option.

Action

Process will continue.

1048

```
Number of output records for %s is %llu.
```

Embedded variable

%s : Output file path name.
%llu: Number of output records.

Action

Process will continue.

1049

```
Number of suppress records for SELECT is %llu.
```

Embedded variable

%llu: Number of records suppressed by record selection option.

Action

Process will continue.

1050

```
Number of suppress records for SUM is %llu.
```

Embedded variable

%llu: Number of records suppressed by record summation option.

Action

Process will continue.

1051

```
Number of suppress records for SUPPRESS is %llu.
```

Embedded variable

%llu: Number of records suppressed by suppression option.

Action

Process will continue.

1052

```
BSORT starts at %s.
```

Embedded variable

%s: Start time

Action

Process will continue.

1053

```
BSORT stopped at %s.
```

Embedded variable

%s: Finish time

Action

Process will continue.

7.1.2 Warning messages

Warning messages are messages that notify regarding an abnormality that PowerBSORT has detected. After the abnormality is detected, processing is continued.
When W or I is specified for the following specification, the error message is output.

- BSORT_MSGLEVEL of startup file
- msglevel operand of -option option in bsortex command
- -l option in bsort command
- msglevel of BSRTPRIM structure in BSORT function

7.1.2.1 Explanation form of warning messages

The explanation form of the warning messages is shown as follows.

Message number

```
Message text
```

Embedded variable

Embedded variable in the warning message is shown as follows.
%s: Meaning of character string shown at position of %s
%d: Meaning of numerical value shown at position of %d

Note

When embedded variable of two or more same types exists, the serial number is added to %s or %d.

Example) %s1, %s2, %d1, %d2

Cause of error

The cause of error is shown.

Action

The operation of PowerBSORT is shown.

Action required by user

The user's action is shown.

7.1.2.2 Warning messages

The warning messages are shown as follows.

0024

```
The message file capacity was insufficient. The output to the message file was interrupted.
```

Embedded variable

None

Cause of error

Unable to extend space due to insufficient disk capacity.

Action

Abort outputting to message file and switch to standard error output (including output of this message) to proceed.

Action required by user

Do one of the following.

- Specify message file to other disk.
- Try the operation again after creating space by deleting unnecessary file.

0028

```
Same file(%s) is specified.
```

Embedded variable

%s: File path name

Cause of error

This is due to either of the following.

- Same file is specified between temporary file and input file, temporary file and output file, temporary file and message file, or temporary file and other temporary file.
- Failed to create a unique path name when creating temporary file path name.

Action

Processing is continued without using the temporary file of the error.

Action required by user

Specify a different file path name.

0029

```
Invalid format of file (%s).
```

Embedded variable

%s: File path name with error

Cause of error

The specified temporary file is not a usual file. Or, the specification of the directory that makes the temporary file is not a directory.

Action

Processing is continued without using the directory that makes the temporary file and the temporary file of the error.

Action required by user

Confirm the form of the temporary file or the specification of the directory that makes the temporary file.

0030

```
Invalid attribute of file (%s).
```

Embedded variable

%s: File path name with error

Cause of error

The attribute of the temporary file is different. Or, the directory that makes the temporary file cannot be used.

Action

Processing is continued without using the directory that makes the temporary file and the temporary file of the error.

Action required by user

Confirm the attribute of the temporary file or the specification of the directory that makes the temporary file.

0032

```
Error occurred during file (%s1) open.(%s2)
```

Embedded variable

%s1: Temporary file path name
%s2: Error number or error detail

Cause of error

The error occurred by open processing of the temporary file.

Action

Processing is continued without using the temporary file of the error.

Action required by user

Remove the cause of the error referring to the manual of the system.

0039

```
There is no reference or writing permission on the file(%s).
```

Embedded variable

%s: Temporary file path name

Cause of error

Temporary file without read or write permission is specified.

Action

Processing is continued without using the temporary file of the error.

Action required by user

Confirm the specification of the temporary file and try the operation again.

0041

```
The overflow occurred at the summation process.
```

Embedded variable

None

Cause of error

An overflow occurred during the summation process.

Action

Either of the following processing is executed.

- If the BSORT_SUMOVERCONT of startup file is not set or is set to OFF, the current summation processing is aborted and subsequent summation processes for records with the same key field value are not started.
- If the BSORT_SUMOVERCONT of startup file is set to ON, the record summated immediately before the overflow occurred is output and subsequent records with the same key field are summated as separate records.

The sort processing and the merge processing continue.
If a value of BSORT_SUMOVER of startup file is set, the specified value will be returned as termination status of bsort command or bsortex command.

Action required by user

Extend the field specified for the summation field.

0042

```
Summation field did not exist on the input record. Therefore, the summation process was
discontinued.
```

Embedded variable

None

Cause of error

This message is output for the following either.

- When entering variable length record, short record that does not include summation field was entered.
- When entering text record, a record that does not include summation field was entered.

Action

Summation processing is aborted but the sort processing and the merge processing continue.
If a value of BSORT_SUMSHRT of startup file is set, the specified value will be returned as termination status of bsort command or bsortex command.

Action required by user

Confirm the specification of the summation field.

0083

```
Error occurred during system call or library function (%s)(%d).
```

Embedded variable

%s: System call or library function name
%d: Error number (errno)

Cause of error

Error occurred during system call or library function.

Action

Processing is continued.

Action required by user

When you can find out the cause by function name or error number, resolve the cause and try the operation again. If you cannot find out the cause, notify your technical support staff (SE) the function name and error number.

0084

```
Error occurred during system call or library function (%s1)(%d) (path name:%s2).
```

Embedded variable

%s1: System call or library function name
%d: Error number (errno)
%s2: Path name

Cause of error

Error occurred during system call or library function.

Action

Processing is continued.

Action required by user

When you can find out the cause by function name, error number or path name, resolve the cause and try the operation again. If you cannot find out the cause, notify your technical support staff (SE) the function name, error number and path name.

0089

```
Unreasonable code is found in the summation-field(%d) specification. The summation process was
interrupted.
```

Embedded variable

%d: Field number

Cause of error

In summation field on the text record, characters other than number, sign, blank, and tab were detected.

Action

Summation processing is aborted but the sort processing and the merge processing continue.
If a value of BSORT_SUMFILD of startup file is set, the specified value will be returned as termination status of bsort command or bsortex command.

Action required by user

Confirm the specification of the summation field.

7.1.3 Error messages

Error messages are messages that notify regarding an abnormality that PowerBSORT has detected. After the abnormality is detected, processing is ended.
When E, W or I is specified for the following specification, the error message is output.

- BSORT_MSGLEVEL of startup file
- msglevel operand of -option option in bsortex command
- -l option in bsort command
- msglevel of BSRTPRIM structure in BSORT function

7.1.3.1 Explanation form of error messages

The explanation form of the error messages is shown as follows.

Message number

```
Message text
```

Embedded variable

Embedded variable in the error message is shown as follows.
%s: Meaning of character string shown at position of %s
%d: Meaning of numerical value shown at position of %d

Note

When embedded variable of two or more same types exists, the serial number is added to %s or %d.

Example) %s1, %s2, %d1, %d2

Cause of error

The cause of error is shown.

Action

The operation of PowerBSORT is shown.

Action required by user

The user's action is shown.

7.1.3.2 Error messages

The error messages are shown as follows.

0001

```
Option (%s) is not specified.
```

Embedded variable

%s: Required option

Cause of error

This message is output in the following cases.

- Although there is no Standard output option (-w), Output file option (-o) is not specified.
- When COBOL file are specified, it is necessary to specify the record format (recform operand or Record format option (-Z)).

Action

Processing is aborted.

Action required by user

Specify necessary option and try the operation again.

0002

```
Exclusive options (%s1), (%s2) are specified.
```

Embedded variable

%s1: Option 1 that is mutually exclusive
%s2: Option 2 that is mutually exclusive

Cause of error

This message is output in the following cases.

- You are specifying the sort option (-s), the merge option (-m), or the copy option (-c) at the same time.
- You are specifying the suppression option (suppress operand or -u) and the record summation option (field operand of -summary or -g) at the same time.
- You are specifying the copy option (-copy or -c) and other options (-summary, overwrite operand of -input, -u, -g, -r or -v) at the same time.
- You are specifying the merge option (-merge or -m) and overwrite option (overwrite operand or -v) at the same time.

Action

Processing is aborted.

Action required by user

Correct the specification of the option in an exclusive relation, and try the operation again.

0003

```
Duplicate options (%s) are specified.
```

Embedded variable

%s: Option that duplicates

Cause of error

Duplicate options are specified besides those for fields.

Action

Processing is aborted.

Action required by user

Correct the duplicate option and try the operation again.

0004

```
Invalid option (%s) is specified.
```

Embedded variable

%s: Invalid option

Cause of error

This message is output in the following cases.

- You are specifying a wrong option.
- You are specifying the merge option (-m) without input file.
- The mistake is found in the specification of the file system (-F).

Action

Processing is aborted.

Action required by user

Correct the wrong option and try the operation again.

0005

```
Invalid value (%s) is specified.
```

Embedded variable

%s: Invalid option

Cause of error

This message is output in the following cases.

- Specified record length (-z) is not a numerical value.
- The value specified are those that are not permitted for this option.

Action

Processing is aborted.

Action required by user

Correct the wrong option and try the operation again.

0006

```
Syntax error (%s).
```

Embedded variable

%s: Character string that detected grammatical mistake

Cause of error

This message is output in the following cases.

- Even though argument is required, none is specified.
- You are specifying no argument option (e.g. -s, -m, -f, -u, -v) and option with an argument (e.g. -y, -z, -o) consecutively.

Action

Processing is aborted.

Action required by user

Correct the format of specified option or operand and try the operation again.

0007

```
The mistake is found in the key-field(%d) specification.
```

Embedded variable

%d: Incorrect field number

Cause of error

This message is output in the following cases.

- Format is invalid.
- You have incorrectly specified the key field (key_entry[%d] of BSRTKEY structure).
- It exceeds the field length that is permitted in data format. Or the value is incorrect.

- The field length specified exceeds the record length. Or no record length is specified.
- Operation of the key field is incorrectly specified.
- The data form that cannot be specified at the text file is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0008

```
The mistake is found in the summation-field(%d) specification.
```

Embedded variable

%d: Incorrect field number

Cause of error

This message is output in the following cases.

- Format is invalid.
- You have incorrectly specified the summation field (sum_entry[%d] of BSRTSUM structure).
- Although BS_SUM is specified in optionfunc of BSRTPRIM structure, BSRTSUM structure is not specified.
- It exceeds the field length that is permitted in data format. Or the value is incorrect.
- The specified field length exceeds the record length. Or the record length is not specified.
- The data form that cannot be specified at the binary file is specified.
- The data form that cannot be specified at the text file is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0009

```
The mistake is found in the reconstruction-field(%d) specification.
```

Embedded variable

%d: Incorrect field number

Cause of error

This message is output in the following cases.

- Format is invalid.
- Although BS_RECON is specified in optionfunc of BSRTPRIM structure, BSRTRCON structure is not specified.
- Unable to recognize construction information (rcon_consinf) specified. Or it is not specified.
- You have specified a literal value (self) that is longer than the literal value (len) of bsort command argument (-e self.len typ).
- You have specified a literal value (self-def) that is longer than the literal value (len) of bsortex command argument (-input reconst=self-def.len typ).

- The data form that cannot be specified at the text file is specified for a literal value.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0010

```
The mistake is found in the selection-field(%d) specification.
```

Embedded variable

%d: Incorrect field number

Cause of error

This message is output in the following cases.

- Format is invalid.
- You have incorrectly specified the selection field (sele_entry[%d] of BSRTSELE structure).
- Although BS_RECON is specified in optionfunc of BSRTPRIM structure, BSRTSELE structure is not specified.
- It exceeds the field length that is permitted in data format. Or the value is incorrect.
- The specified field length exceeds the record length. Or the record length is not specified.
- The true operation or the false operation or the link destination is incorrect.
- Operation of the compared field is incorrectly specified.
- The data form that cannot be specified at the text file is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0011

```
Invalid environment variable is specified.
```

Embedded variable

None

Cause of error

Invalid environment variable is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0012

```
Invalid startup file is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- The mistake is found in the description form of a startup file.
- A startup file is not found.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0013

```
The mistake is found in startup file.
```

Embedded variable

None

Cause of error

- The mistake is found in the key word or the value specified with a startup file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0014

```
The mistake is found in BSRTPRIM(%s) specified by the bsrtopen function.
```

Embedded variable

%s: Field name where error was detected

Cause of error

This message is output in the following cases.

- Unable to recognize main option (function). Or it is not specified.
- Unable to recognize record format (recform). Or it is not specified.
- Unable to recognize option function (optionfunc).
- Unable to recognize input output information (fileoprat).
- Unable to recognize method of handing over record (recoprat).
- Unable to recognize level of the message (msglevel).
- Unable to recognize code system of the input file (cdmode).
- Unable to recognize check mechanism (chklevel).
- Unable to recognize operation of the key field (keyoption or keyoption2).
- The numbers of files or strings are not specified for merge option.

- When merge option, it is specified that the record is input from a standard input.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0015

```
The mistake is found in BSRTFILE(%s) specified by the bsrtopen function.
```

Embedded variable

%s: Field name where error was detected

Cause of error

This message is output in the following cases.

- The entry in input file (inpfile_tbl) is invalid.
- The entry in output file (outfile_tbl) is invalid.
- The entry in temporary file (tmpfile_tbl) is invalid.
- The entry in message file (msgfile_addr) is invalid.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0016

```
Key field and summation field positions, or summation fields positions are identical.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- Key field and summation field are overlapping.
- Summation fields are overlapping.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0018

```
Invalid use for bsrtput function.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- Although you specified to input from file, bsrtput function is issued.
- After specifying BS_ENDDATA (end passing record) with bsrtput function, bsrtput function is issued.
- Although the merge option is specified in bsrtopen function, bsrtput function only for the sort option is issued.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0019

```
Invalid use for bsrtget function.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- Although you specified the output to file, bsrtget function is issued.
- Although BS_ENDDATA (end passing record) is not specified in bsrtput function (while passing record), bsrtget function is issued.
- After PowerBSORT notifies the end of the record, bsrtget function is issued.
- Although the merge function is specified in bsrtopen function, bsrtget function only for the sort option is issued.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0020

```
Invalid use for bsrtmrge function.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- Although bsrtopen function is specifying input and output file, bsrtmrge function is issued.
- After PowerBSORT notifies the end of the record, bsrtmrge function is issued.
- Although the sort function is specified in bsrtopen function, bsrtmrge function only for the merge option is issued.
- A string number that is larger than the value specified in bsrtopen function is set for that of bsrtmrge function.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0021

```
Record address is not specified in %s function.
```

Embedded variable

%s: bsrtput or bsrtmrge

Cause of error

This message is output in the following cases.

- Record pointer is not specified in bsrtput function.
- Although bsrtmrge function has not completed passing records, record pointer is not specified.

Action

Processing is aborted.

Action required by user

Correct user application so that the record pointer is set. If all records are passed, revise user application so that BS_ENDDATA is specified.

0022

```
Invalid record length is specified in %s function.
```

Embedded variable

%s: bsrtopen, bsrtput or bsrtmrge

Cause of error

This message is output in the following cases.

- When fixed length record format, a record length that is different from the length specified by bsrtopen function is specified.
- When variable length record format, a record length longer than maximum record length specified by bsrtopen function is specified.
- Record length is not specified in bsrtopen function.
- Record length for a fixed length records group was not integral multiples of the record length specified by bsrtopen function.

Action

Processing is aborted.

Action required by user

Do one of the following.

- Correct the user application so that the record length would be correct.
- Confirm the maximum record length specified by the bsrtopen function is more than an actual maximum record length. If incorrect, correct the user application.
- When detected by bsrtopen function, correct the user application to specify record length.

0023

```
A necessary BSORT working area cannot be secured.
```

Embedded variable

None

Cause of error

A necessary memory cannot be secured.

Action

Processing is aborted.

Action required by user

Do one of the following.

- Increase a specified value when you specify the memory size. If the amount of physical memory is small, you have to decrease the amount specified.
- Confirm if there is enough available memory for PowerBSORT if calling from user application.
- At least 64KB of memory is necessary for PowerBSORT.

 See

Refer to the following for the specification of the memory size.

- 1.7.1 Estimate of memory (work area) size that PowerBSORT uses

0025

```
Not enough temporary file space.
```

Embedded variable

None

Cause of error

Disk space is detected to be insufficient while outputting data to temporary file.

Action

Processing is aborted.
Message number 0175 or message number 0176 is output following this message.

Action required by user

Do one of the following.

- If there is free disk space, define directory with free space as temporary directory.
- If there is free disk space and temporary file path name is specified by bsrtopen function, increase the number of specified temporary file.
- If there is no free space on disk, create free space by deleting unnecessary files.

 See

Refer to the following for the specification of the temporary file.

- Environment variable TMPDIR
- BSORT_TMPDIR of startup file
- tmpdir operand of -option option in bsortex command

- tmpfile_tbl of BSRTFILE structure in bsrtopen function

0026

```
The temporary file cannot be created.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- The mistake is found in the specification of the temporary file.
- Virtual storage space or disk space needed to create temporary file is insufficient to automatically procure temporary file space.

Action

Processing is aborted.

Action required by user

Do one of the following.

- Confirm the specification of the temporary file.
- Decrease the amount of virtual storage space used by PowerBSORT or user application.
- Try the operation again when the load against the system is low.
- Delete unnecessary files and create free disk space.

0027

```
File(%s) does not exist.
```

Embedded variable

%s: Input file path name or output file path name

Cause of error

File path name for input or output file does not exist.

Action

Processing is aborted.

Action required by user

Confirm the input or output file name.

0028

```
Same file(%s) is specified.
```

Embedded variable

%s: File path name

Cause of error

This message is output in the following cases.

- Although the same file is specified for input and output file, overwrite permission (overwrite operand of -input option in bsortex command, -v option in bsort command or BS_OVERWRITE of fileoprat in bsrtopen function) is not specified.

- Same file is specified for input and output in the merge option and the copy option.
- Same file is specified for input file and message file.
- Same file is specified for output file and message file.

Action

Processing is aborted.

Action required by user

Note that same file path name should not be specified.
Specify the overwrite permission (overwrite operand of -input option in bsortex command, -v option in bsort command or BS_OVERWRITE of fileoprat in bsrtopen function) when you specify the same file as input and output in the sort option. If you specify same file for input and output and encounter error during operation, you may not be able to restore file.

0029

```
Invalid format of file (%s).
```

Embedded variable

%s: File path name with error

Cause of error

This message is output in the following cases.

- The specified input file is not a usual file.
- The specified output file is not a usual file.
- The specified message file is not a usual file.
- The specified argument file is not a usual file.
- Variable length record format is specified but file system is omitted or standard file system of system is specified.

Action

Processing is aborted.

Action required by user

Confirm file format and try the operation again.

0030

```
Invalid attribute of file (%s).
```

Embedded variable

%s: File path name with error

Cause of error

This message is output in the following cases.

- Both the variable length format and fixed length format are specified for input file in mixture.
- The file specified for output has different format from input records.
- The specified record format is different from the record format of the file.
- The specified record length is different from the record length of the file.
- The input file is making or updating it.

Action

Processing is aborted.

Action required by user

Confirm file attribute and try the operation again.

0031

```
Record length is not specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- Record length is not specified.
- The rec_len of BSRTPRIM structure is not specified.

Action

Processing is aborted.

Action required by user

Do one of the following.

- Specify the record length.
- Specify the record length in rec_len of BSRTPRIM structure.

0032

```
Error occurred during file (%s1) open.(%s2)
```

Embedded variable

%s1: File path name
%s2: Error code (Note) or error detail

Note

To investigate the cause of the error, the following values might be set to the error code.
Refer to the manual of the file system for details of the error code.

file system	error code
Standard file system of system	Error code (errno) that the open system call sets
NetCOBOL file system	Error code that the NetCOBOL file system sets

Cause of error

This message is output in the following cases.

- Ran out of memory when attempting to process open file in system.
- Unable to extend space for parent directory.
- Component for file path name is not a directory.
- The length of the path name exceeds PATH_MAX.

- The length of the element of the path name exceeds NAME_MAX.
- Unable to secure space for file.
- File is in use by other process.
- Data corruption occurred due to media or software failure.
- Disk which file resides is not available.
- If it is an indexed file, error exists in specifying index.

Action

Processing is aborted.

Action required by user

Do one of the following.

- If it is due to insufficient memory, reduce the amount of virtual storage space used by PowerBSORT or user application and try the operation again.
- If it is due to insufficient disk space, create free space and try the operation again.
- If the component of file path is not a directory, specify a correct file path name.
- If the file is used in another process, try the operation again when the file is not used.
- If the path name or the component exceeds the limit, specify a correct name and try the operation again.

0033

```
There are too many files open in the system or in the process.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- Too many files are open within process or system so necessary files cannot be opened.
- The number of temporary files needed for sort option is too many.

Action

Processing is aborted.

Action required by user

Do one of the following.

- Reduce the number of input file if bsort command or bsortex command is in use.
- If using BSORT function, close open files before issuing BSORT function.
- If the number of files opened is too many, try the operation again when system load is low.
- If the number of temporary file required for sort option is too many, expand the disk space specified for temporary file. If you are specifying temporary file using BSORT_TMPDIR of startup file or environment variable TMPDIR, shorten the length of each folder path name.

0034

```
Error occurred during file (%s1) close.(%s2)
```

Embedded variable

%s1: File path name
%s2: Error code (Note) or error detail

Note

To investigate the cause of the error, the following values might be set to the error code.
Refer to the manual of the file system for details of the error code.

file system	error code
Standard file system of system	Error code (errno) that the close system call sets
NetCOBOL file system	Error code that the NetCOBOL file system sets

Cause of error

This message is output in the following cases.

- Memory is insufficient.
- Data corruption occurred due to media or software failure.
- Disk which file resides is not available.

Action

Processing is aborted.

Action required by user

Reduce the amount of virtual storage space for PowerBSORT or user application and try the operation again.

0035

```
Error occurred during the file (%s1) reading.(%s2)
```

Embedded variable

%s1: File path name
%s2: Error code (Note) or error detail

Note

To investigate the cause of the error, the following values might be set to the error code.
Refer to the manual of the file system for details of the error code.

file system	error code
Standard file system of system	Error code (errno) that the read system call sets
NetCOBOL file system	Error code that the NetCOBOL file system sets

Cause of error

This message is output in the following cases.

- Ran out of memory while inputting record from file.
- Data corruption occurred due to media or software failure.
- Disk which file resides is not available.

Action

Processing is aborted.

Action required by user

Reduce the amount of virtual storage space for PowerBSORT or user application and try the operation again.

0036

```
Error occurred during the file (%s1) writing.(%s2)
```

Embedded variable

%s1: File path name
%s2: Error code (Note) or error detail

Note

To investigate the cause of the error, the following values might be set to the error code.
Refer to the manual of the file system for details of the error code.

file system	error code
Standard file system of system	Error code (errno) that the write system call sets
NetCOBOL file system	Error code that the NetCOBOL file system sets

Cause of error

This message is output in the following cases.

- Ran out of memory while outputting record to file.
- Unable to expand file space.
- Data corruption occurred due to media or software failure.
- Disk which file resides is not available.

Action

Processing is aborted.

Action required by user

Do one of the following.

- If it is due to insufficient memory, reduce the amount of virtual storage space used by PowerBSORT or user application and try the operation again.
- If it cannot expand disk space for output file, create free disk space and try the operation again.

0037

```
There is no reference permission in file(%s).
```

Embedded variable

%s: File path name

Cause of error

File without permission to read is specified.

Action

Processing is aborted.

Action required by user

Do one of the following.

- If file path name is incorrect, try the operation again after specifying the correct path name.
- If no error exits in what you have specified, inquire administrator for permission to read.

0038

```
There is no writing permission in the file(%s).
```

Embedded variable

%s: File path name

Cause of error

File without permission to write is specified.

Action

Processing is aborted.

Action required by user

Do one of the following.

- If file path name is incorrect, try the operation again after specifying the correct path name.
- If no error exits in what you have specified, inquire administrator for permission to write.

0040

```
The string was not sorted.
```

Embedded variable

None

Cause of error

The file or the string specified for the input of the merge option is not sorted in order of the specified key field.

Action

Processing is aborted.

Action required by user

In the merge option, specify sorted file or sorted string.

0043

```
Logical error occurred at BSORT.(%s-%d)
```

Embedded variable

%s: Detected module name
%d: Error identification number

Cause of error

Logical error was detected.

Action

Processing is aborted.

Action required by user

Contact technical support staff (SE) along with the error identification number.

Information

If the detection module is "qha7tmpp" and the error identification number is "162", there is a possibility that the input file has been updated in other processes. Confirm whether other processes had updated the input file while processing PowerBSORT.

0072

```
Selection field did not exist on the input record.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- When entering variable length record, short record that does not include selection field was entered.
- When entering text record, a record that does not include selection field was entered.

Action

Processing is aborted.

Action required by user

Confirm the specification of the selection field.

0073

```
Reconstruction field did not exist on the input record.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- When entering variable length record, short record that does not include reconstruction field was entered.
- When entering text record, a record that does not include reconstruction field was entered.

Action

Processing is aborted.

Action required by user

Confirm the specification of the reconstruction field.

0074

```
Unreasonable code is found in the key field.
```

Embedded variable

None

Cause of error

Undefined code is detected in the key field.

Action

Processing is aborted.

Action required by user

Confirm the specification of the key field.

0075

```
Error occurred in file system(%s).
```

Embedded variable

%s: Specified file system

Cause of error

This message is output in the following cases.

- The specified file system is not installed.
- The library of the file system is not specified by environment variable LD_LIBRARY_PATH.

Action

Processing is aborted.

Action required by user

Do one of the following.

- Install the necessary file system.
- Specify the library of the file system by environment variable LD_LIBRARY_PATH.

0076

```
The mistake is found in argument file.
```

Embedded variable

None

Cause of error

Quotation mark (') or double quotation mark (") is not used in pairs.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0077

```
Number of symbolic links encountered during path name(%s) traversal exceeds MAXSYMLINKS.
```

Embedded variable

%s: File path name

Cause of error

The number of symbolic links found while checking the input file, the output file or the argument file exceeds MAXSYMLINKS.

Action

Processing is aborted.

Action required by user

Decrease the number of symbolic links to MAXSYMLINKS or less, and try the operation again.

0078

```
File path name(%s) is too long.
```

Embedded variable

%s: File path name

Cause of error

The length of the path name of the input file, the output file, and the argument file exceeds the value of PATH_MAX defined by the system.

Action

Processing is aborted.

Action required by user

Correct the length of the file path name to PATH_MAX or less, and execute it again.

0079

```
A path prefix component of %s is not a directory.
```

Embedded variable

%s: File path name

Cause of error

The path prefixes for input file, output file or argument file contain those that are not a directory.

Action

Processing is aborted.

Action required by user

Correct the specified of file path name and try the operation again.

0080

```
Specified %s is a directory.
```

Embedded variable

%s: path name

Cause of error

Directory is specified for output file.

Action

Processing is aborted.

Action required by user

Correct the specified of output file and try the operation again.

0081

```
No space left on the output device.
```

Embedded variable

None

Cause of error

The free space on the device was insufficient while writing to the output file or the temporary file.

Action

Processing is aborted.

Action required by user

Allocate device that has sufficient free disk space and try the operation again.

0082

```
File(%s) size exceeds the process's file size limit or the maximum file size.
```

Embedded variable

%s: File name

Cause of error

The file size of output file or temporary file exceeded the limit of file size specified for process in system or exceeded the maximum file size set by user.

Action

Processing is aborted.

Action required by user

Increase the file size limit or maximum file size of process, and try the operation again. Only the super-user can increase the file size limit of process. The maximum file size cannot exceed the file size limit of process specified for the system.

0083

```
Error occurred during system call or library function (%s)(%d).
```

Embedded variable

%s: System call or library function name
%d: Error number (errno)

Cause of error

Error occurred during system call or library function.

Action

Processing is aborted.

Action required by user

When you can find out the cause by function name or error number, resolve the cause and try the operation again. If you cannot find out the cause, notify your technical support staff (SE) the function name and error number.

0084

```
Error occurred during system call or library function (%s1)(%d) (path name:%s2).
```

Embedded variable

%s1: System call or library function name
%d : Error number (errno)
%s2: Path name

Cause of error

Error occurred during system call or calling library function.

Action

Processing is aborted.

Action required by user

When you can find out the cause by function name, error number or path name, resolve the cause and try the operation again. If you cannot find out the cause, notify your technical support staff (SE) the function name, error number and path name.

0085

```
Error occurred during BSORT loading.
```

Embedded variable

None

Cause of error

The installation directory is not set to environment variable LD_LIBRARY_PATH or the mistake is found in the setting.

Action

Processing is aborted.

Action required by user

Confirm environment variable LD_LIBRARY_PATH setting.

0086

```
Line feed did not exist on the record specified in %s function.
```

Embedded variable

%s: bsrtput or bsrtmrge

Cause of error

Line feed does not exist on the specified text record.

Action

Processing is aborted.

Action required by user

Do one of the following.

- Confirm the input record is a text form.
- Confirm the record length specified by bsrtput function or bsrtmrge function.

0088

```
Invalid record length is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- When entering variable length record, a record that exceeds the specified record length has been entered.
- When entering text record, a record that exceeds the specified record length has been entered.

Action

Processing is aborted.

Action required by user

Confirm the specification of the record length.

0093

```
Invalid environment variable (%s) is specified.
```

Embedded variable

%s: Invalid environment variable

Cause of error

Invalid environment variable is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0094

```
The mistake (%s) is found in startup file.
```

Embedded variable

%s: Keyword in the startup file with error

Cause of error

The keyword in the startup file is invalid, or an incorrect value is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0095

```
The mistake is found in BSRTOPT(%s) specified by the bsrtopen function.
```

Embedded variable

%s: Field name where error was detected

Cause of error

This message is output in the following cases.

- The option function (optfunc) cannot be recognized.
- The output record position (outrecpos) cannot be recognized.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0096

```
The mistake is found in record length after the input process.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- When input reconstruction is not specified, the record length specified by input file groups is different.
- When input reconstruction is specified, the record length after input reconstruction specified by the input file groups is different.

Action

Processing is aborted.

Action required by user

Do one of the following.

- If using input reconstruction option, try the operation again after specifying input file groups to use same record length after input reconstruction.
- If not using input reconstruction option, try the operation again after specifying input file groups to use same record length.

0097

```
License checkout of %s failed (%d).
```

Embedded variable

%s: Feature name

%d: Error number

Cause of error

License checkout failed.

Action

Processing is aborted.

Action required by user

Confirm the license and try the operation again.

0098

```
Synchronous option (%s1), (%s2) are not specified.
```

Embedded variable

%s1: Required option 1
%s2: Required option 2

Cause of error

Either of the two options required is missing.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0099

```
Startup file(%s) does not exist.
```

Embedded variable

%s: Path name of the startup file

Cause of error

The file path name specified for the startup file does not exist.

Action

Processing is aborted.

Action required by user

Confirm the startup file path name.

0100

```
Invalid conversion method between ASCII code and EBCDIC code is specified.
```

Embedded variable

None

Cause of error

Error exists in the specification of conversion method between ASCII code and EBCDIC code.

See

- iconv operand of -option option in bsortex command
- Character code system conversion option (-Q) in bsort command
- altmode of BSRTPRIM structure in BSORT function

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0102

```
Invalid index creation method is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- Neither main nor sub key is specified.
- The duplicate and unique keys of the index are both specified at the same time.
- Invalid index creation method is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0103

```
Exclusive index options, duplicate key and unique key are specified.
```

Embedded variable

None

Cause of error

The mutually exclusive index options duplicate key and unique key are both specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0104

```
Index data type is not specified.
```

Embedded variable

None

Cause of error

Index data format is not specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0105

```
Index creation method is not specified.
```

Embedded variable

None

Cause of error

Index creation method is not specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0106

```
Invalid index length is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- The length of the index is 0 byte.
- The length of the index exceeds the length that can be specified by the specified data format.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0107

```
Invalid index order is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- In NetCOBOL indexed file, descending order is specified.
- Invalid index order is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0108

```
The mistake is found in the operation of the key field when the key field is omitted.
```

Embedded variable

None

Cause of error

When the entire record is assumed to be a key field, the operation of the key field in an exclusive relation is specified at the same time.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0109

```
Invalid key type is specified.
```

Embedded variable

None

Cause of error

Invalid data format of the key field is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0110

```
Exclusive options, copy and summation/suppress/reverse are specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- Record summation option is specified in copy option.
- Suppression option is specified in copy option.
- Descending order option is specified in copy option.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0111

```
Invalid string number is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- The number of input files or strings is different in merge option.
- Zero is specified for number of string in merge option.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0112

```
Invalid check level is specified.
```

Embedded variable

None

Cause of error

Invalid check level is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0113

```
Invalid field separating character-string in the text file is specified.
```

Embedded variable

None

Cause of error

Record delimiter character is included in field separating character string.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0114

```
Text field format is not specified.
```

Embedded variable

None

Cause of error

The method of specifying the field is not specified for the text file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0116

```
Invalid file system is specified.
```

Embedded variable

None

Cause of error

Invalid file system is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0117

```
Same file(%s) is specified for message file and startup file.
```

Embedded variable

%s: File path name

Cause of error

Same file is specified for message file and startup file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0118

```
Invalid message level is specified.
```

Embedded variable

None

Cause of error

Invalid message level is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0119

```
Invalid output line delimiter is specified.
```

Embedded variable

None

Cause of error

Invalid output line delimiter is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0120

```
Invalid record format is specified.
```

Embedded variable

None

Cause of error

Invalid record format is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0121

```
Record format is not specified.
```

Embedded variable

None

Cause of error

Record format is not specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0122

```
Record format or text field format is not specified.
```

Embedded variable

None

Cause of error

Record format or text field format is not specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0123

```
Invalid method of passing and receiving records is specified.
```

Embedded variable

None

Cause of error

Invalid method of passing and receiving records is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0124

```
Exclusive options, max output file size/max output record number and indexed file/C-ISAM file are
specified.
```

Embedded variable

None

Cause of error

The max output file size or the max output record number is specified in indexed file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0125

```
Exclusive options, summation and suppress are specified.
```

Embedded variable

None

Cause of error

The record summation option and the suppression option cannot be specified at the same time.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0126

```
First/last option is specified without summation/suppress function.
```

Embedded variable

None

Cause of error

The output of first or last record is specified without specifying record summation option or suppression option.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0127

```
Same file(%s) is specified for output file and startup file.
```

Embedded variable

%s: File path name

Cause of error

Same file is specified for output file and startup file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0128

```
Same file(%s) is specified for output file and message file.
```

Embedded variable

%s: File path name

Cause of error

Same file is specified for output file and message file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0129

```
Same file(%s) is specified for output files.
```

Embedded variable

%s: File path name

Cause of error

Same file is specified for output files.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0130

```
Invalid output file system is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- When the output file is text file, file systems other than the standard file system of the system are specified.
- When the output file is standard output, file systems other than the standard file system of the system are specified.
- Invalid output file system is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0131

```
Invalid input line delimiter is specified.
```

Embedded variable

None

Cause of error

Invalid input line delimiter is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0132

```
Invalid output file path name(%s) is specified.
```

Embedded variable

%s: File path name

Cause of error

Invalid output file path name is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0133

```
Invalid max output file size is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- The maximum file size specified is smaller that output record length (length after reconstruction).
- Invalid max output file size is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0134

```
Invalid max output file size or max output record number is specified.
```

Embedded variable

None

Cause of error

The max output file size or the max output record number is specified in standard output.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0135

```
Exclusive options, merge/copy and overwrite are specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- I/O overwrite option is specified in merge option.
- I/O overwrite option is specified in copy option.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0136

```
Invalid input data code is specified.
```

Embedded variable

None

Cause of error

Invalid input data code is specified.

See

- icode operand of -option option in bsortex command
- Input code system option (-q) in bsort command
- cdmode of BSRTPRIM structure in BSORT function

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0137

```
Same file(%s) is specified for input file and startup file.
```

Embedded variable

%s: File path name

Cause of error

Same file is specified for input file and startup file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0138

```
Same file(%s) is specified for input file and message file.
```

Embedded variable

%s: File path name

Cause of error

Same file is specified for input file and message file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0139

```
Same file(%s) is specified for input file and output file.
```

Embedded variable

%s: File path name

Cause of error

Same file is specified for input file and output file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0140

```
Invalid input file system is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- When the input file is text file, file systems other than the standard file system of the system are specified.
- When the input file is standard input, file systems other than the standard file system of the system are specified.
- Invalid input file system is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0141

```
Invalid input file path name(%s) is specified.
```

Embedded variable

%s: File path name

Cause of error

Invalid input file path name is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0142

```
Exclusive options, standard input and overwrite are specified.
```

Embedded variable

None

Cause of error

Standard input and I/O overwrite option are specified at the same time.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0143

```
Input file path name is not specified.
```

Embedded variable

None

Cause of error

Input file path name is not specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0144

```
Output file path name is not specified.
```

Embedded variable

None

Cause of error

Output file path name is not specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0145

```
The mistake is found in input file system specification.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- The file system is not defined in a startup file corresponding to the specification of the input file system.
- The mistake is found in input file system specification.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0146

```
The mistake is found in output file system specification.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- The file system is not defined in a startup file corresponding to the specification of the output file system.
- The mistake is found in output file system specification.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0147

```
Invalid index data type is specified.
```

Embedded variable

None

Cause of error

This message is output in the following cases.

- In NetCOBOL indexed file, data format besides ASCII code and Unicode are specified.
- Invalid index data type is specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0148

```
Exclusive options, standard file system and variable-length record format are specified.
```

Embedded variable

None

Cause of error

Variable length record format is specified in standard file system of system.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0149

```
Exclusive options, output record passing and indexed file/C-ISAM file are specified.
```

Embedded variable

None

Cause of error

Output record passing and indexed file are specified.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0150

```
BSRTPRIM is not specified by the bsrtopen function.
```

Embedded variable

None

Cause of error

BSRTPRIM structure is not specified in bsrtopen function.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0151

```
Key option is specified for binary file.
```

Embedded variable

None

Cause of error

The operation of key field for the text file is specified at the binary file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0152

```
Exclusive options, text file and EBCDIC input data code are specified.
```

Embedded variable

None

Cause of error

EBCDIC code system is specified for the code system of the input file at the text file.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0153

```
The mistake is found in %s1(%s2) specified by the bsrtopen function.
```

Embedded variable

%s1: Structure name of which error was detected
%s2: Field name of which error was detected

Cause of error

Mistake is found in idx_flag of BSFILE_IDX structure.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0154

```
Division of output file cannot be specified for one output file.
```

Embedded variable

None

Cause of error

When the division output of the output file by size of the maximum file or number of maximum records is specified, only one output file is specified.

Action

Processing is aborted.

Action required by user

Do one of the following.

- Specify multiple output files and try the operation again.
- Delete the specification of the division output of the output file, and try the operation again

0155

```
Exclusive options, -sort, -merge or -copy are specified.
```

Embedded variable

None

Cause of error

Either the sort option, the merge option or the copy option are specified at the same time.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0156

```
Invalid operand (%s) is specified.
```

Embedded variable

%s: Invalid operand

Cause of error

This message is output in the following cases.

- Strings that cannot be recognized as operand is specified.
- Invalid operation of the key field is specified in key=ALL of bsortex command.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0157

```
Not enough output file after %s specified for the output division.
```

Embedded variable

%s: Last file path name specified in output division

Cause of error

There is a record that cannot be output because the output file specified by the division output is few.

Action

Processing is aborted.

Action required by user

Increase the number of files for output division and try the operation again.

0158

```
Max output file size is specified with the file system which does not support it.
```

Embedded variable

None

Cause of error

Max output file size is specified, except for the following files.

- System standard file (binary file or text file)
- NetCOBOL file uses the high speed access library for a NetCOBOL sequential file (libbscblfast64.so)
- NetCOBOL file uses the high speed access library for a NetCOBOL physical sequential file (libbscblpsfast64.so)

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0159

```
Operand (%s) is not specified.
```

Embedded variable

%s: Necessary operand

Cause of error

Record length is not specified in reclen operand.

Action

Processing is aborted.

Action required by user

Specify the necessary operand and try the operation again.

0161

```
The mistake is found in the index-field(%d) specification.
```

Embedded variable

%d: Incorrect field number

Cause of error

This message is output in the following cases.

- Error in position or length of the index field.
- Error in data format of the index field.
- Error in operation of the index field.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0173

```
Attribute acquisition error of file(%s1). (%s2)
```

Embedded variable

%s1: File path name
%s2: Error number

Cause of error

This message is output in the following cases.

- Error occurred by the attribute acquisition processing of the file.
- The file that has opened NetCOBOL application program in a sharing mode is specified for an input file of PowerBSORT, and the standard file system of the system is specified for the input file system.

Action

Processing is aborted.

Action required by user

Correct the mistake and try the operation again.

0174

```
BSORT is not correctly installed.
```

Embedded variable

None

Cause of error

PowerBSORT is not correctly installed.

Action

Processing is aborted.

Action required by user

Reinstall PowerBSORT.

0175

```
Temporary directory (%s) was used.
```

Embedded variable

%s: Temporary directory name

Cause of error

Disk space is detected to be insufficient while outputting data to temporary file.

Action

Processing is aborted.
This message is output as addition information on message number 0025 following message number 0025.

Action required by user

Do one of the following.

- If there is free disk space, define directory with free space as temporary directory.
- If there is no free space on disk, create free space by deleting unnecessary files.

0176

```
Temporary file (%s) was used.
```

Embedded variable

%s: Temporary file name

Cause of error

Disk space is detected to be insufficient while outputting data to temporary file.

Action

Processing is aborted.
This message is output as addition information on message number 0025 following message number 0025.

Action required by user

Do one of the following.

- If there is free disk space, increase the number of temporary files specified by the bsrtopen function.
- If there is no free space on disk, create free space by deleting unnecessary files.

0177

```
Key field is not specified.
```

Embedded variable

None

Cause of error

The specification of the key field is not omissible for text file CSV format or text file TSV format.

Action

Processing is aborted.

Action required by user

Specify the key field and try the operation again.

7.2 Error Detail Codes of BSORT Function

Error detail codes that may be notified when you call PowerBSORT from C language program are set in errdetail of BSRTREC structure. The following section explains the detailed codes and its meaning set in errdetail.

Error detail Code	Definition value of error detail code	Meaning
1	BSERR_VL	Error in bsrtopen or bsrtclse function's first argument.
2	BSERR_LOAD	Error when loading PowerBSORT.
3	BSERR_DELETE	Error in an attempt to delete PowerBSORT from virtual storage.
4	BSERR_LMFAIL	License checkout failed.
5	BSERR_PRODUCT	PowerBSORT is not correctly installed.
50	BSERR_MEMSIZE	The specified memory size by bsrtopen function is too small.
51	BSERR_PASSMEM	Memory address range specified by bsrtopen function is not cleared by bsrtclse function.
52	BSERR_ALCMEM	Not enough PowerBSORT virtual storage working area is secured.
53	BSERR_MEMZERO	Bsrtopen function is not specifying the amount of virtual storage while specifying the address.
54	BSERR_MANYFILE	Too many input file paths specified by bsrtopen function.
55	BSERR_INFILETAB	Input file name table specified by bsrtopen function is invalid.
56	BSERR_BSRTPRIM	Error in BSRTPRIM information specified by bsrtopen function. - While bsrtopen function is specifying merge option, the number of files and strings are not specified. - The number of strings entered does not match the one specified by BSRTFILE. - Record formats specified by input file and BSRTPRIM do not match. - A regular file has been specified as the input file, but the regular file size is not a multiple of the record length.
57	BSERR_EXCLUSIVE	Functions mutually exclusive are specified. - Suppression option and record summation option - Sort option, merge option and copy option
59	BSERR_KEYTAB	Error in key field. - Key field is outside the range of record. - Unexpected data format of key field. - Key length is not supported by data format. - Error in specifying key operation.
60	BSERR_SUMTAB	Error in summation field - The number of summation field is not specified. - Summation field is outside the range of record. - Unexpected data format of summation field. - Summation length is not supported by data format.
61	BSERR_DUPKEY	Key and summation field or the two summation fields are identical.

Error detail Code	Definition value of error detail code	Meaning
62	BSERR_UNSUPPORT	Unsupported option specified.
63	BSERR_SELTAB	Error in the selection field.
64	BSERR_RECONTAB	Error in the reconstruction field.
65	BSERR_FILESYS	The file system specified is not valid.
66	BSERR_SELSHRTREC	The selection field specified is not valid. - The record entered is shorter than the selection field. - The text record entered does not include selection field.
67	BSERR_RCONSHRTREC	The reconstruction field specified is not valid. - The record entered is shorter than the reconstruction field. - The text record entered does not include reconstruction field.
68	BSERR_OUTFILETAB	Error in output file table.
69	BSERR_BSRTOPT	BSRTOPT structure specified by bsrtopen function is incorrect.
100	BSERR_NOOPEN	Bsrtput, bsrtget or bsrtmrge function is in use before calling bsrtopen function.
101	BSERR_BSRTPUT	Bad usage in bsrtput function. - Although records are specified to be loaded from file by bsrtopen function, bsrtput function is in use. - Although record has already been passed, bsrtput function still in use. - Although merge option is specified, bsrtput function is in use.
102	BSERR_BSRTGET	Bad usage in bsrtget function. - Although records are specified to be output to file by bsrtopen function, bsrtget function is in use. - Although records are not yet passed completely by bsrtput function, bsrtget function is in use. - Although all records are outputted, bsrtget function is in use. - Although merge option is specified, bsrtget function is in use.
103	BSERR_BSRTMRGE	Bad usage in bsrtmrge function. - Although bsrtopen function is specified to input/output record to file, bsrtmrge function is in use. - Although all record is outputted by bsrtmrge function, bsrtmrge function is still in use. - Although sort option is specified, bsrtmrge function is in use. - String number that is larger than those specified by bsrtopen function is specified by bsrtmrge function.
104	BSERR_BSRTCLSE	Bsrtclse function is called in the middle of output process.
110	BSERR_NORECADR	There are no record address settings.
111	BSERR_RECLEN	Bad record length.
114	BSERR_NOTORDER	Input string for the merge option is not in line.
115	BSERR_OVERFLOW	The summation operation aborted due to overflow.

Error detail Code	Definition value of error detail code	Meaning
116	BSERR_SUMSHRTREC	The summation field specified is invalid. - Summation process will abort due to entering insufficient variable length record that lacks summation field. - Summation process will abort due to entering insufficient text record that lacks summation field.
117	BSERR_NONLF	Line feed is missing at the end of record in text file.
118	BSERR_EXTSHRTREC	Record without key field is entered during inputting text record.
119	BSERR_SUMFIELD	Summation process will abort because invalid code was detected in field during text record summation.
200	BSERR_READFILE	Error occurred while reading record from file.
201	BSERR_WRITEFILE	Error occurred while writing record to file.
202	BSERR_FILEATTR	Bad file attribute. Failed to get file attribute.
203	BSERR_TEMPATTR	Temporary file attribute is invalid. Failed to get file attribute.
204	BSERR_FILEFORMAT	Bad file format. - Ordinary and sequential files are specified in mixture. - Relative or indexed file are specified. - File besides ordinary file is specified for message file.
205	BSERR_SAMEFILE	Duplicate file path name. - Same file has been specified for input and output file. - Same file has been specified for input and message file. - Same file has been specified for output and message file.
206	BSERR_RECFORM	Invalid record format. - Fixed-length and variable-length file is specified in a mixture. - Input/output record format are different.
207	BSERR_FILERECLEN	Bad record length. - When variable-length file is specified, this occurs when record length of file attribute is longer than the length specified by bsrtopen function. - When fix-length file is specified, this occurs when length of file attribute and length specified by bsrtopen function differ.
208	BSERR_FILEOPEN	Error occurred when opening file.
209	BSERR_OPENMAX	Too many files are opened within process or system. Or the number of required temporary file exceeds the maximum.
210	BSERR_FILECLOSE	Error occurred when closing file.
211	BSERR_FILEIO	Data corruption due to media or software failure.
212	BSERR_TEMPSPACE	Not enough temporary file space.
213	BSERR_INCORE	Unable to perform in-core sort.
214	BSERR_TEMPCREAT	Unable to create temporary file.
215	BSERR_FILENONE	File specified for input or output does not exist.
216	BSERR_PERMIFILE	No permission to read the specified file for input.

Error detail Code	Definition value of error detail code	Meaning
217	BSERR_PERMOFILE	No permission to write to the specified file for output.
218	BSERR_PERMMFILE	No permission to write to the specified message file.
219	BSERR_PERMTFILE	No permission to write or read the specified temporary file.
220	BSERR_MSGSPACE	The message file size is insufficient. Output to the message file will interrupt.
221	BSERR_EXTRECLEN	Record length specified in bsrtopen function is invalid.
222	BSERR_STRFILEREC	When you specify ordinary file for input, record length cannot be omitted.
223	BSERR_CLSEARG	Bad bsrtclse function argument.
224	BSERR_WRITETEMP	Error occurred when writing to temporary file.
225	BSERR_READTEMP	Error occurred when reading temporary file.
226	BSERR_ENVAR	Environment variable setting is wrong.
227	BSERR_SUPFILE	Invalid startup file is specified.
228	BSERR_SUPFILVAR	The mistake is found in startup file.
229	BSERR_PERMSFILE	No permission to read the startup file.
230	BSERR_EUC	Unreasonable code is found in the key field or selection field.
231	BSERR_COBOL85	File system error occurred. - No library exists for specified file system. - The mistake is found in environment variable LD_LIBRARY_PATH setting.
232	BSERR_SYMLINK	The number of symbolic link found while checking path name exceeds MAXSYMLINKS.
233	BSERR_NAMELONG	File path name is too long.
234	BSERR_NOTDIR	A path prefix component is not a directory.
235	BSERR_ISDIR	The specified path is a directory.
236	BSERR_OUTNOSPC	Running out of space left on the output device.
237	BSERR_FILEBIG	The file size exceeded the limit of process or the maximum file size.
238	BSERR_LIBC	System call or library function error.
250	BSERR_LOGICAL	Error occurred in PowerBSORT operation.

Information

The definition value of the error details code is defined by header file (bsrt.h).

Appendix A Examples

A.1 Examples of using bsort command

This section explains the examples of using the bsort command via the main options.

- Sort processing
- Merge processing
- Copy processing
- Using the record selection option
- Using the record reconstruction option
- Using the record summation option
- Using the suppression option
- Using the FIFO option
- Sorting the text files (specifying fixed fields)
- Sorting the text files (specifying floating fields)
- Using the output file switching option

Example 1 : Sort processing

It sorts records of the binary fixed length file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of 10 bytes ASCII code in 10 bytes from the head of the record in ascending order.

Input file (bsortin)

+0	+10 ... +100
key	
C	Record1
B	Record2
A	Record3

Output file (bsortout)

+0	+10 ... +100
key	
A	Record3
B	Record2
C	Record1

key : Key field

```
bsort -s -z100 -0.10asca -o bsortout bsortin

  -s          : Sort function
  -z100       : Record length
  -0.10asca   : Key field
  -o bsortout : Output file name
  bsortin     : Input file name
```

Example 2 : Merge processing

It merges records of the binary fixed length file bsortin1 and bsortin2 of which the record length is 100 bytes and outputs them to the file bsortout. File bsortin1 and bsortin2 are files that sorted the field of ASCII code in 10 bytes from the head of the record in ascending order respectively.

Input file (bsortin1)

+0	+10	+100
key		
A	Record1-1	
C	Record1-2	
E	Record1-3	

Input file (bsortin2)

+0	+10	+100
key		
B	Record2-1	
D	Record2-2	

Output file (bsortout)

+0	+10	+100
key		
A	Record1-1	
B	Record2-1	
C	Record1-2	
D	Record2-2	
E	Record1-3	

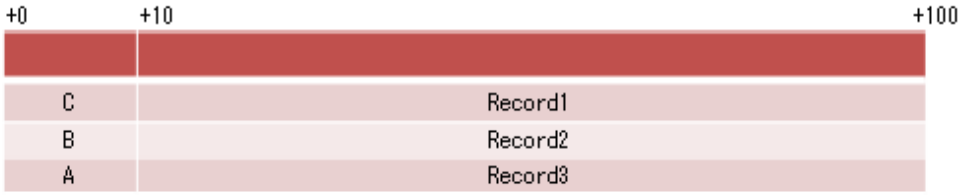
key : key field

```
bsort -m -z100 -0.10asca -o bsortout bsortin1 bsortin2

  -m          : Merge function
  -z100       : Record length
  -0.10asca   : Key field
  -o bsortout : Output file name
  bsortin1    : Input file name
  bsortin2    : Input file name
```

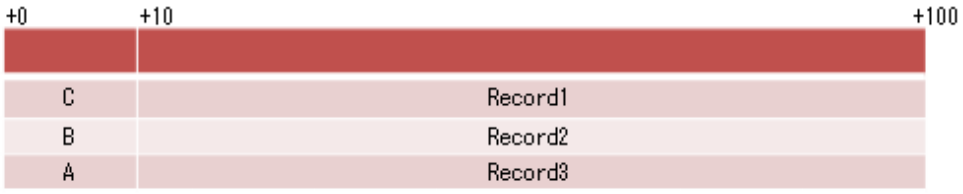
Example 3 : Copy processing

It copies records of the binary fixed length file bsortin of which the record length is 100 bytes to the sequential file bsortout of NetCOBOL.

Input file (bsortin)



Output file (bsortout)




```
bsort -c -z100 -Zf -Fcobs64,ufs -o bsortout bsortin

  -c           : Copy function
  -z100        : Record length
  -Zf          : Record format
  -Fcobs64,ufs : Input/Output file system
  -o bsortout  : Output file name
  bsortin      : Input file name
```

Example 4 : Using the record selection option

It merges records of the binary fixed length file bsortin1 of which the record length is 100 bytes and the records of the sequential file bsortin2 of NetCOBOL and outputs them to the sequential file bsortout of NetCOBOL. The input file is to be sorted in the field of the fixed-point binary number that is in 4 bytes from the 21st byte in ascending order. In addition, when merging the records, the records with the value of the fixed-point binary number that is in 4 bytes from 31st byte is more than 30 and less than 40 are to be merged.

Input file (bsortin1)

+0	+20	+24	+30	+34	+100
	key		sel		
	1		30	Record1-1	
	4		29	Record1-2	
	5		35	Record1-3	

Input file (bsortin2)

+0	+20	+24	+30	+34	+100
	key		sel		
	2		40	Record2-1	
	3		39	Record2-2	

Output file (bsortout)

+0	+20	+24	+30	+34	+100
	key		sel		
	1		30	Record1-1	
	3		39	Record2-2	
	5		35	Record1-3	

key : Key field sel : Compared field

```
bsort -m -z100 -Zf -20.4fbia -p30.4fbi.ge.d30,30.4fbi.lt.d40 -Fcobs64,ufs,cobs64 -o bsortout bsortin1
bsortin2

  -m                              : Merge function
  -z100                           : Record length
  -Zf                             : Record format
  -20.4fbia                       : Key field
  -p30.4fbi.ge.d30,30.4fbi.lt.d40 : Selection field
  -Fcobs64,ufs,cobs64             : Input/Output file system
  -o bsortout                     : Output file name
  bsortin1                        : Input file name
  bsortin2                        : Input file name
```

Example 5 : Using the record reconstruction option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 20 bytes from the 11th bytes of the record in descending order. In addition, the records are reconstructed at the same time of the sort processing. Reconstruction is to be specified to put 20 bytes from the 11th bytes of the input record in the left end of the input record and to put a field of 10 bytes in the form of external decimal number on the right of that and set 0 for that value.

Input file (bsortin)

+0 +10 +30 +100

	rcn	
	A	Record1
	B	Record2
	C	Record3

Output file (bsortout)

+0 +20 +30

rcn	lit
C	0
B	0
A	0

rcn : Key field & Reconstruction field lit : Literal value

```
bsort -s -z100 -0.20ascr -e10.20,d0.10zdl -o bsortout bsortin

  -s               : Sort function
  -z100            : Record length
  -0.20ascr        : Key field
  -e10.20,d0.10zdl : Reconstruction field
  -o bsortout      : Output file name
  bsortin          : Input file name
```

Point

In specifying the command to execute sort processing and reconstruction at the same time, the value after the reconstruction is specified for the position of key field.

Example 6 : Using the record summation option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 20 bytes from the 11th bytes of the record in descending order. In sorting, if the same record appears in the key field, the field of the packed decimal number in 8 bytes from the 51st byte is added to it and the records are to be summated.

Input file (bsortin)

+0	+10	+30	+50	+58	+100
	key		sum		
	A		15	Record1	
	A		25	Record2	
	B		10	Record3	
	A		30	Record4	

Output file (bsortout)

+0	+10	+30	+50	+58	+100
	key		sum		
	B		10	Record3	
	A		70	Record1	

key : Key field sum : Summation field

```
bsort -s -z100 -10.20ascr -g50.8pdl -o bsortout bsortin

  -s           : Sort function
  -z100        : Record length
  -10.20ascr   : Key field
  -g50.8pdl    : Summation field
  -o bsortout  : Output file name
  bsortin      : Input file name
```

Point

It is irregular which record output from among the record where the key field is equal.

Example 7 : Using the suppression option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 20 bytes from the 11th bytes of the record in descending order. In sorting, if the same record appears in the key field, arbitrary 1 record is kept and others are deleted.

Input file (bsortin)

+0	+10	+30	+50	+58	+100
	key				
	A		15	Record1	
	A		25	Record2	
	B		10	Record3	
	A		30	Record4	

Output file (bsortout)

+0	+10	+30	+50	+58	+100
	key				
	B		10	Record3	
	A		15	Record1	

key : Key field

```
bsort -s -u -z100 -10.20ascr -o bsortout bsortin

  -s           : Sort function
  -u           : Suppression option
  -z100        : Record length
  -10.20ascr   : Key field
  -o bsortout  : Output file name
  bsortin      : Input file name
```

Point

It is irregular which record output from among the record where the key field is equal.

Example 8 : Using the FIFO option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 20 bytes from the 11th bytes of the record in descending order. In sorting, if the same record appears in the key field, the records are output in the order of storing in the input file.

Input file (bsortin)

+0	+10	+30	+50	+58	+100
	key				
	A		15	Record1	
	A		25	Record2	
	B		10	Record3	
	A		30	Record4	

Output file (bsortout)

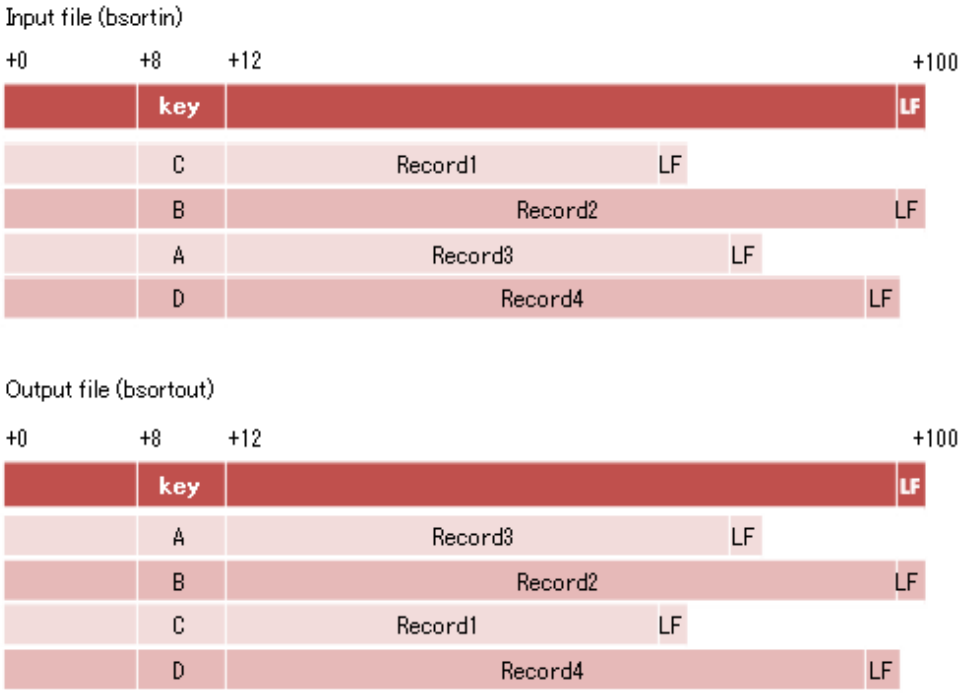
+0	+10	+30	+50	+58	+100
	key				
	B		10	Record3	
	A		15	Record1	
	A		25	Record2	
	A		30	Record4	

key : Key field

```
bsort -s -f -z100 -10.20ascr -o bsortout bsortin

  -s           : Sort function
  -f           : FIFO option
  -z100        : Record length
  -10.20ascr   : Key field
  -o bsortout  : Output file name
  bsortin      : Input file name
```

Example 9 : Sorting the text files (specifying fixed fields)

It sorts records of the text file bsortin of which the maximum record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 4 bytes from the 9th bytes of the record in ascending order.

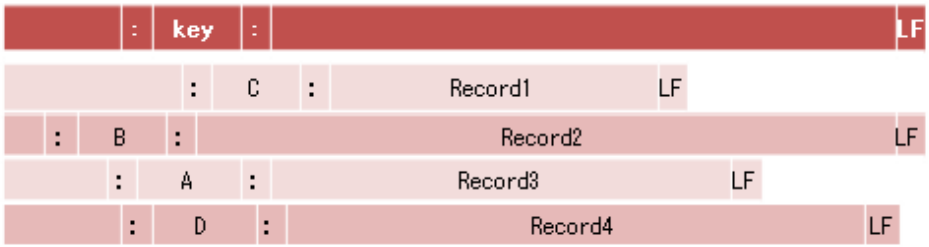



```
bsort -s -z100 -8.4asca -Tfix -o bsortout bsortin

  -s          : Sort function
  -z100       : Maximum record length
  -8.4asca    : Key field
  -Tfix       : Text file fixed field specification
  -o bsortout : Output file name
  bsortin     : Input file name
```

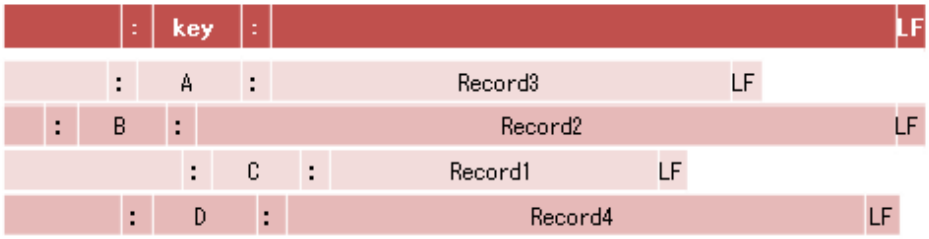
Example 10 : Sorting the text files (specifying floating fields)

It sorts records of the text file bsortin of which the maximum record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of the ASCII code in 4 bytes from the head of the 2nd field delimited by record separator colon (:) in ascending order.

Input file (bsortin)



Output file (bsortout)



key : Key field LF : Line feed

```
bsort -s -z100 -1.4asca -Tflt -t: -o bsortout bsortin

  -s          : Sort function
  -z100       : Maximum record length
  -1.4asca    : Key field
  -Tflt       : Text file floating field specification
  -t:         : Field separation character
  -o bsortout : Output file name
  bsortin     : Input file name
```

Example 11 : Using the output file switching option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. When the file bsortout1 abnormally terminates, the remaining records are output to file bsortout2. The key field is to be specified to sort the field of the ASCII code in 10 bytes from the head of the record in ascending order.

Input file (bsortin)

+0	+10	+100
key		
E	Record1	
B	Record2	
C	Record3	
D	Record4	
A	Record5	

Output file (bsortout1)

+0	+10	+100
key		
A	Record5	
B	Record2	
C	Record3	

* bsortout1 abnormally terminate

Output file (bsortout2)

+0	+10	+100
key		
D	Record4	
E	Record1	

key : Key field

```
bsort -s -z100 -0.10asca -o bsortout1 -o bsortout2 bsortin

  -s           : Sort function
  -z100        : Record length
  -0.10asca    : Key field
  -o bsortout1 : Output file name
  -o bsortout2 : Output file name (When the file bsortout1 abnormally terminates)
  bsortin      : Input file name
```

A.2 Examples of using bsortex command

This section explains the examples of using the bsortex command via the main options.

- Sort processing
- Merge processing
- Copy processing
- Using the record selection option
- Using the record reconstruction option
- Using the record summation option
- Using the suppression option
- Using the FIFO option
- Sorting the text files (specifying fixed fields)
- Sorting the text files (specifying floating fields)

- Using the output file switching option
- Using the conditional file output option (No.1)
- Using the conditional file output option (No.2)

Example 1 : Sort processing

It sorts records of the binary fixed length file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of 10 bytes ASCII code in 10 bytes from the head of the record in ascending order.

Input file (bsortin)

+0	+10	+100
key		
C	Record1	
B	Record2	
A	Record3	

Output file (bsortout)

+0	+10	+100
key		
A	Record3	
B	Record2	
C	Record1	

key : Key field

```
bsortex -sort key=0.10asca -input reclen=100 file=bsortin -output file=bsortout

  -sort           : Sort function
    key=0.10asca  : Key field
  -input
    reclen=100    : Record length
    file=bsortin  : Input file name
  -output
    file=bsortout : Output file name
```

Example 2 : Merge processing

It merges records of the binary fixed length file bsortin1 and bsortin2 of which the record length is 100 bytes and outputs them to the file bsortout. File bsortin1 and bsortin2 are files that sorted the field of ASCII code in 10 bytes from the head of the record in ascending order respectively.

Input file (bsortin1)

+0	+10	+100
key		
A	Record1-1	
C	Record1-2	
E	Record1-3	

Input file (bsortin2)

+0	+10	+100
key		
B	Record2-1	
D	Record2-2	

Output file (bsortout)

+0	+10	+100
key		
A	Record1-1	
B	Record2-1	
C	Record1-2	
D	Record2-2	
E	Record1-3	

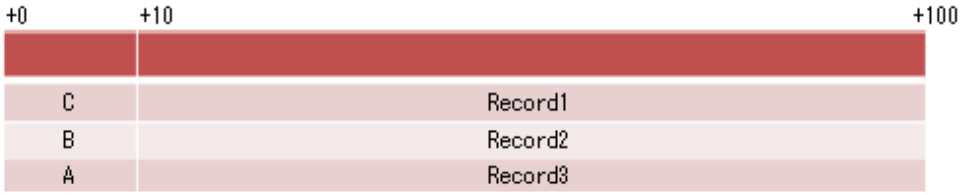
key : key field

```
bsortex -merge key=0.10asca -input reclen=100 file=bsortin1,bsortin2 -output file=bsortout

  -merge                    : Merge function
    key=0.10asca            : Key field
  -input
    reclen=100              : Record length
    file=bsortin1,bsortin2  : Input file name
  -output
    file=bsortout           : Output file name
```

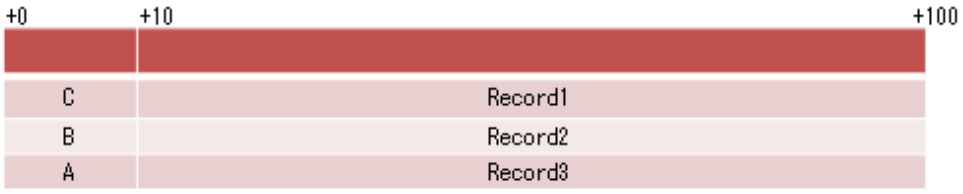
Example 3 : Copy processing

It copies records of the binary fixed length file bsortin of which the record length is 100 bytes to the sequential file bsortout of NetCOBOL.

Input file (bsortin)



Output file (bsortout)




```
bsortex -copy -record recform=fix -input reclen=100 file=bsortin filesys=ufs -output file=bsortout
filesys=cobs64

  -copy            : Copy function
  -record
    recform=fix    : Record format
  -input
    reclen=100     : Record length
    file=bsortin   : Input file name
    filesys=ufs    : Input file system
  -output
    file=bsortout  : Output file name
    filesys=cobs64 : Output file system
```

Example 4 : Using the record selection option

It merges records of the binary fixed length file bsortin1 and bsortin2 of which the record length is 100 bytes and outputs them to the sequential file bsortout of NetCOBOL. The input file is to be sorted in the field of the fixed-point binary number that is in 4 bytes from the 21st byte in ascending order. In addition, when merging the records, the records with the value of the fixed-point binary number that is in 4 bytes from 31st byte is more than 30 and less than 40 are to be merged.

Input file (bsortin1)

+0	+20	+24	+30	+34	+100
	key		sel		
	1		30	Record1-1	
	4		29	Record1-2	
	5		35	Record1-3	

Input file (bsortin2)

+0	+20	+24	+30	+34	+100
	key		sel		
	2		40	Record2-1	
	3		39	Record2-2	

Output file (bsortout)

+0	+20	+24	+30	+34	+100
	key		sel		
	1		30	Record1-1	
	3		39	Record2-2	
	5		35	Record1-3	

key : Key field sel : Compared field

```
bsortex -merge key=20.4fbia -record recform=fix -input reclen=100 file=bsortin1,bsortin2 filesys=ufs
include=30.4fbi.ge.d30.and.30.4fbi.lt.d40 -output file=bsortout filesys=cobs64

  -merge                                      : Merge function
    key=20.4fbia                              : Key field
  -record
    recform=fix                               : Record format
  -input
    reclen=100                                : Record length
    file=bsortin1,bsortin2                    : Input file name
    filesys=ufs                               : Input file system
    include=30.4fbi.ge.d30.and.30.4fbi.lt.d40 : Selection field
  -output
    file=bsortout                             : Output file name
    filesys=cobs64                            : Output file system
```

Example 5 : Using the record reconstruction option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 20 bytes from the 11th bytes of the record in descending order. In addition, the records are reconstructed at the same time of the sort processing. Reconstruction is to be specified to put 20 bytes from the 11th bytes of the input record in the left end of the input record and to put a field of 10 bytes in the form of external decimal number on the right of that and set 0 for that value.

Input file (bsortin)

+0 +10 +30 +100

	rcn	
	A	Record1
	B	Record2
	C	Record3

Output file (bsortout)

+0 +20 +30

rcn	lit
C	0
B	0
A	0

rcn : Key field & Reconstruction field lit : Literal value

```
bsortex -sort key=0.20ascr -input reclen=100 file=bsortin reconst=9.20,d0.10zdl -output file=bsortout

  -sort                     : Sort function
    key=0.20ascr            : Key field
  -input
    reclen=100              : Record length
    file=bsortin            : Input file name
    reconst=10.20,d0.10zdl : Reconstruction field
  -output
    file=bsortout           : Output file name
```

Point

In specifying the command to execute sort processing and reconstruction at the same time, the value after the reconstruction is specified for the position of key field.

Example 6 : Using the record summation option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 20 bytes from the 11th bytes of the record in descending order. In sorting, if the same record appears in the key field, the field of the packed decimal number in 8 bytes from the 51st byte is added to it and the records are to be summated. And the first input record out of the records in the same key record is output.

Input file (bsortin)

+0	+10	+30	+50	+58	+100
	key		sum		
	A		15	Record1	
	A		25	Record2	
	B		10	Record3	
	A		30	Record4	

Output file (bsortout)

+0	+10	+30	+50	+58	+100
	key		sum		
	B		10	Record3	
	A		70	Record1	

key : Key field sum : Summation field

```
bsortex -sort key=10.20ascr -input reclen=100 file=bsortin -summary field=50.8pdl first -output
file=bsortout

  -sort           : Sort function
    key=10.20ascr : Key field
  -input
    reclen=100    : Record length
    file=bsortin  : Input file name
  -summary
    field=50.8pdl : Summation field
    first         : Output the first input record
  -output
    file=bsortout : Output file name
```

Example 7 : Using the suppression option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 20 bytes from the 11th bytes of the record in descending order. In sorting, if the same record appears in the key field, the record input last is kept and others are deleted.

Input file (bsortin)

+0	+10	+30	+50	+58	+100
	key				
	A		15	Record1	
	A		25	Record2	
	B		10	Record3	
	A		30	Record4	

Output file (bsortout)

+0	+10	+30	+50	+58	+100
	key				
	B		10	Record3	
	A		30	Record4	

key : Key field

```
bsortex -sort key=10.20ascr -input reclen=100 file=bsortin -summary suppress last -output
file=bsortout

  -sort           : Sort function
    key=10.20ascr : Key field
  -input
    reclen=100    : Record length
    file=bsortin  : Input file name
  -summary
    suppress      : Suppression option
    last          : Output the last input record
  -output
    file=bsortout : Output file name
```

Example 8 : Using the FIFO option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 20 bytes from the 11th bytes of the record in descending order. In sorting, if the same record appears in the key field, the records are output in the order of storing in the input file.

Input file (bsortin)

+0	+10	+30	+50	+58	+100
	key				
	A		15	Record1	
	A		25	Record2	
	B		10	Record3	
	A		30	Record4	

Output file (bsortout)

+0	+10	+30	+50	+58	+100
	key				
	B		10	Record3	
	A		15	Record1	
	A		25	Record2	
	A		30	Record4	

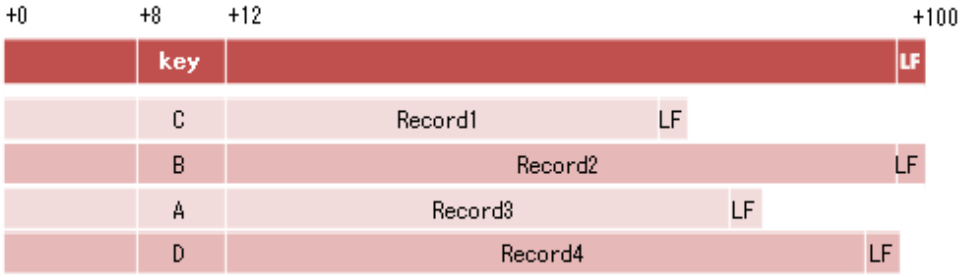
key : Key field

```
bsortex -sort key=10.20ascr -input reclen=100 file=bsortin -output file=bsortout -option fifo

  -sort           : Sort function
    key=10.20ascr : Key field
  -input
    reclen=100    : Record length
    file=bsortin  : Input file name
  -output
    file=bsortout : Output file name
  -option
    fifo          : FIFO option
```

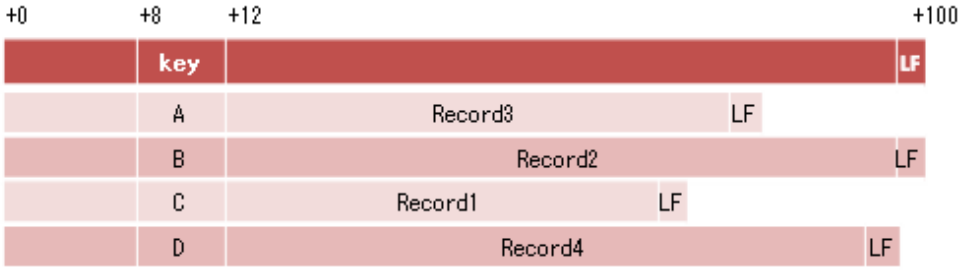
Example 9 : Sorting the text files (specifying fixed fields)

It sorts records of the text file bsortin of which the maximum record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of ASCII code in 4 bytes from the 9th bytes of the record in ascending order.

Input file (bsortin)



Output file (bsortout)



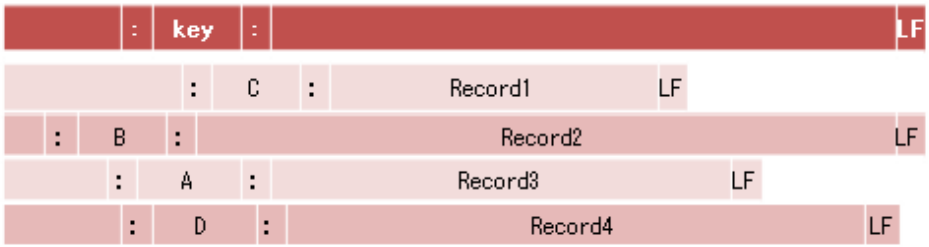
key : Key field LF : Line feed

```
bsortex -sort key=8.4asca -record recform=txtfix -input reclen=100 file=bsortin -output file=bsortout

  -sort            : Sort function
    key=8.4asca    : Key field
  -record
    recform=txtfix : Record format (text file fixed field specification)
  -input
    reclen=100     : Maximum record length
    file=bsortin   : Input file name
  -output
    file=bsortout  : Output file name
```

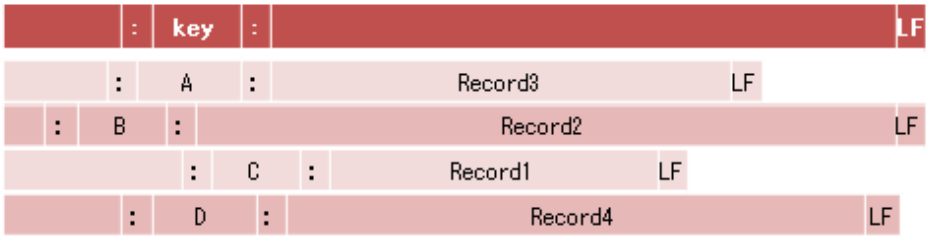
Example 10 : Sorting the text files (specifying floating fields)

It sorts records of the text file bsortin of which the maximum record length is 100 bytes and outputs them to the file bsortout. The key field is to be specified to sort the field of the ASCII code in 4 bytes from the head of the 2nd field delimited by record separator colon (:) in ascending order.

Input file (bsortin)



Output file (bsortout)



key : Key field LF : Line feed

```
bsortex -sort key=1.4asca -record recform=txtflt fldsep=: -input reclen=100 file=bsortin -output
file=bsortout

  -sort            : Sort function
    key=1.4asca    : Key field
  -record
    recform=txtflt : Record format (text file floating field specification)
    fldsep=:       : Field separation character
  -input
    reclen=100     : Maximum record length
    file=bsortin   : Input file name
  -output
    file=bsortout  : Output file name
```

Example 11 : Using the output file switching option

It sorts records of the binary file bsortin of which the record length is 100 bytes and outputs them to the file bsortout1 and file bsortout2. When outputting them, the file size is no more than 1 GB. The key field is to be specified to sort the field of ASCII code in 10 bytes from the head of the record in ascending order.

Input file (bsortin)

+0	+10 +100
key	
E	Record1
B	Record2
C	Record3
D	Record4
A	Record5
:	:

Output file (bsortout1)

+0	+10 +100
key	
A	Record5
B	Record2
C	Record3
:	:

* File size exceeds 1GB

Output file (bsortout2)

+0	+10 +100
key	
D	Record4
E	Record1
:	:

key : Key field

```
bsortex -sort key=0.10asca -input reclen=100 file=bsortin -output file=bsortout1,bsortout2
maxfilesize=1G

  -sort                      : Sort function
    key=0.10asca             : Key field
  -input
    reclen=100               : Record length
    file=bsortin             : Input file name
  -output
    file=bsortout1,bsortout2 : Output file name
    maxfilesize=1G           : Maximum output file size
```

Example 12 : Using the conditional file output option (No.1)

It sorts records of the file bsortin of which the record length is 100 bytes and outputs them to the file bsortout1 and file bsortout2. When outputting the records, the records in which the field of a packed decimal number in 4 bytes from 31st byte is more than 100 are output to bsortout1, and the records in which the field of an external decimal number in 8 bytes from 51st byte is less than 50 is output to bsortout2. The key field is to be specified to sort the field of the ASCII code in 10 bytes from the head of the record in ascending order.

Input file (bsortin)

+0	+10	+30	+34	+50	+58	+100
key		sel1		sel2		
E		100		50	Record1	
B		99		49	Record2	
C		101		30	Record3	
D		90		55	Record4	
A		110		60	Record5	

Output file (bsortout1)

+0	+10	+30	+34	+50	+58	+100
Key		sel1		sel2		
A		110		60	Record5	
C		101		30	Record3	
E		100		50	Record1	

Output file (bsortout2)

+0	+10	+30	+34	+50	+58	+100
key		sel1		sel2		
B		99		49	Record2	
C		101		30	Record3	

key : Key field sel1, sel2 : Selection field

```
bsortex -sort key=0.10asca -input reclen=100 file=bsortin -output file=bsortout1
include=30.4pdl.ge.d100 -output file=bsortout2 include=50.8zdl.lt.d50

  -sort                     : Sort function
    key=0.10asca            : Key field
  -input
    reclen=100              : Record length
    file=bsortin            : Input file name
  -output
    file=bsortout1          : Output file name
    include=30.4pdl.ge.d100 : Selection field
  -output
    file=bsortout2          : Output file name
    include=50.8zdl.lt.d50  : Selection field
```

Point

In the include operand, the record where the selection condition of each -output option is approved is output to each output file. Therefore, Record3 is output to both bsortout1 and bsortout2 in this example.

Example 13 : Using the conditional file output option (No.2)

It sorts records of the file bsortin of which the record length is 100 bytes and outputs them to the file bsortout1 and bsortout2 and bsortout3. When outputting the records, the records in which the field of a packed decimal number in 4 bytes from 31st byte is more than 100 are output to bsortout1, and among the records that were not output to bsortout1, the records in which the field of an external decimal number in 8 bytes from 51st byte is less than 50 are output to bsortout2, and the rest of the records that were not output to either bsortout1 nor

bsortout2 are to be output to bsortout3. The key field is to be specified to sort the field of the ASCII code in 10 bytes from the head of the record in ascending order.

Input file (bsortin)

+0	+10	+30	+34	+50	+58	+100
key		sel1		sel2		
E		100		50	Record1	
B		99		49	Record2	
C		101		30	Record3	
D		90		55	Record4	
A		110		60	Record5	

Output file (bsortout1)

+0	+10	+30	+34	+50	+58	+100
Key		sel1		sel2		
A		110		60	Record5	
C		101		30	Record3	
E		100		50	Record1	

Output file (bsortout2)

+0	+10	+30	+34	+50	+58	+100
key		sel1		sel2		
B		99		49	Record2	

Output file (bsortout3)

+0	+10	+30	+34	+50	+58	+100
key		sel1		sel2		
D		90		55	Record4	

key : Key field sel1, sel2 : Selection field

```
bsortex -sort key=0.10asca -input reclen=100 file=bsortin -output file=bsortout1 case=30.4pdl.ge.d100
-output file=bsortout2 case=50.8zdl.lt.d50 -output file=bsortout3 case=other

  -sort                  : Sort function
    key=0.10asca         : Key field
  -input
    reclen=100           : Record length
    file=bsortin         : Input file name
  -output
    file=bsortout1       : Output file name
    case=30.4pdl.ge.d100 : Selection field
  -output
    file=bsortout2       : Output file name
    case=50.8zdl.lt.d50  : Selection field
  -output
    file=bsortout3       : Output file name
    case=other           : Selection field (Other record)
```

Point

In the case operand, the record where the selection condition of the -output option is approved is output to the output file. The record that has already been output is not output in the following -output options. Therefore, Record3 is output only to bsortout1 in this example.

A.3 Examples of using C language program

The example of using the BSORT function is installed in the following places at each language (locale) used as a sample program.

Language	Directory of sample program
English	/opt/FJSVXbsrt/sample/C/
Japanese (Unicode)	/opt/FJSVXbsrt/sample/ja_JP.UTF-8/
Japanese (EUC)	/opt/FJSVXbsrt/sample/ja_JP.eucJP/

Makefile to compile sample1.c - sample8.c and the sample program is installed in the directory of the sample program. The outline of processing of each sample program is shown below.

Sample program	Outline of processing
sample1.c	Sort the record passed from the application program and return it to the application program.
sample2.c	Merge the record passed from the application program and return it to the application program.
sample3.c	Sort the record in the input file and output it to the output file.
sample4.c	Merge the record in the input files and output it to the output file.
sample5.c	Copy the record in the input file onto the output file.
sample6.c	Combine and process the record selection option, the record reconstruction option, and the record summation option to the sort function.
sample7.c	Sort the record in the text file and output it to the output file.
sample8.c	Combine and process a complex specified record selection option to the sort function.

The input file that the sample program uses is installed in the sample directory. Copy the input file in the current directory before executing the sample program. The list of the input file that each sample program uses is shown as follows.

Input file name	Sample program name
sortin	sample1.c, sample3.c, sample5.c, sample6.c
sortin.txt	sample7.c
selein	sample8.c
mergein1	sample2.c, sample4.c
mergein2	sample2.c, sample4.c

Appendix B Notes

B.1 Environmental Setting

It is necessary to add the directory of PowerBSORT to the following environment variables to use PowerBSORT.

Environment variable	Directory of PowerBSORT
PATH	/opt/FJSVXbsrt/bin
LD_LIBRARY_PATH	/opt/FJSVXbsrt/lib
MANPATH	/opt/FJSVXbsrt/man/%L:/opt/FJSVXbsrt/man
NLSPATH	/opt/FJSVXbsrt/lib/nls/%L/%N.cat:/opt/FJSVXbsrt/lib/nls/C/%N.cat

Also, the sample script to set the environment variable is installed and refer to an environment setting.

Shell	Sample script
Bourne shell	/opt/FJSVXbsrt/config/psort.sh
C shell	/opt/FJSVXbsrt/config/psort.csh

B.2 How to Resolve Insufficient Memory

If memory lacks in PowerBSORT execution, follow the procedure below.

When the specification of the memory size is too large

PowerBSORT needs the consecutive memory within the range of the specified memory size. It is thought that being not able to secure the consecutive memory when PowerBSORT is executed is one of the causes of the error. Therefore, omit or decrease the specification of the memory size, and try the operation again. When the specification of the memory size is omitted, PowerBSORT automatically sets an appropriate memory size that meets processing condition.

When the specification of the memory size is too small

A necessary memory size increases according to treated data size and operating condition (content of processing). It is thought that being not able to secure a memory necessary because the specification of the memory size is too small when PowerBSORT is executed is one of the causes of the error. Therefore, increase the specification of the memory size, and try the operation again. The maximum value for the memory size is 2097151KB (2GB). However, specify not to exceed the physical memory size that can be used.

See

Refer to the following for the specification of the memory size.

- memsize operand of Execution environment option (-option) in bsortex command
- Memory size option (-y) in bsort command
- memory_size of BSRTPRIM structure in bsrtopen function
- BSORT_MEMSIZE of startup file

B.3 How to Specify a Temporary File Directory

When a temporary file directory is specified by the BSORT_TMPDIR of startup file, tmpdir operand (-option option in bsortex command) or environment variable TMPDIR, note the following points.

- Specify the directory that can be used.

- When specifying two or more directories, delimit by the colon and specify the directory without putting the blank.

Example

1) Correct specification

```
/sortwk1:/sortwk2
```

2) Wrong specification

```
/sortwk1 : /sortwk2
```

- Specify directory with enough empty space.

B.4 How to Specify the Location of the Field

PowerBSORT processes each option in the following order.

1. Data input processing
2. Input record selection processing
3. Input record reconstruction processing
4. Sort processing, merge processing and Copy processing
5. Record summation processing
6. Output record selection processing
7. Output record reconstruction processing
8. Data output processing

Therefore, if the location of the field is changed by the input record reconstruction processing, the location of the field in the files to be specified after the processing also is different from the one in the input record. Specify the location of the record according to the following rules.

Input selection field

The field is always specified based on the input record.

Input reconstruction field

The field is always specified based on the input record.

Key field

Normally, it is specified based on the input record. However, in the case of using the input record reconstruction option, the field is specified based on the record after the reconstruction.

Summation field

Normally, it is specified based on the input record. However, in the case of using the input record reconstruction option, the field is specified based on the record after the reconstruction.

Output selection field

Normally, it is specified based on the input record. However, in the case of using the input record reconstruction option, the field is specified based on the record after the reconstruction.

Output reconstruction field

Normally, it is specified based on the input record. However, in the case of using the input record reconstruction option, the field is specified based on the record after the reconstruction.

B.5 Important Notes about the Order of the Data containing Character Strings and Numbers

When the data containing character strings and numbers are sorted by using N operation, the following notes are necessary.

Example of specification of key field 1

When the data containing character strings and numbers are sorted by using N operation, the number field is compared only by the value of numbers. Therefore, the output order of the data as the following cannot be expected.

```
0.9ascaN
```

Input data	Output result case 1	Output result case 2
data001	data001	data00001
data2	data00001	data1
data1	data1	data001
data00001	data2	data2

Point

Considering the number field of 1, 001 and 00001, the value is 1 in all data, data1, data001 and data00001 are judged to have the same key value. If the FIFO (First-in First-out) option is specified, the data is output in the same order as the input order. However, if the FIFO option is not specified, the output order becomes irregular.

Example of specification of key field 2

If the digit numbers are to be considered to sort the data, specify as follows.

```
0.9ascaN,0.9ascr
```

Input data	Output result
data001	data1
data2	data001
data1	data00001
data00001	data2

Point

The order of the output is decided as follows.

1. The order between data001/data1/data00001 and data2 is decided by comparing the first key (0.9ascaN). Since the order becomes definite by the first key comparison, data2 comparison using the second key will no longer be executed.
2. Then, as the values of the data001/data1/data00001 key are the same, they are compared by the second key (0.9ascr).
3. The second key is sorted from the biggest to the smallest by comparing the character string. In this example, the character next to the data is used to decide the order. Therefore, data1 is judged as bigger than data001/data00001 and is output first. In the same way, when comparing data001 and data00001, the third figure of data001 stands at 1 while the corresponding figure of data00001

stands at 0. So, data001 is judged as bigger than data00001 and is output secondly. The remaining data, data00001, and data2 which order has already been decided, are output in order, resulting in the above output result.

B.6 How to Specify the Arranging ASCII code in Order of EBCDIC or the Arranging EBCDIC in Order ASCII code

How to specify the arranging ASCII code in order of EBCDIC

Specify the ASCII code system for the input code system, and specify EBCDIC for data format of the key field. Moreover, specify the conversion methods between ASCII code and EBCDIC.

Example

1. Example of specifying bsort command

```
bsort -s -z 100 -20.8ebca -q au -Q 1 bsortin -o bsortout
```

2. Example of specifying bsortex command

```
bsortex -sort key=20.8ebca -input reclen=100 file=bsortin -output file=bsortout -option icode=au
iconv=1
```

How to specify the arranging EBCDIC in order of ASCII code

Specify the EBCDIC code system for the input code system, and specify ASCII code for data format of the key field. Moreover, specify the conversion methods between ASCII code and EBCDIC.

Example

1. Example of specifying bsort command

```
bsort -s -z 100 -20.8asca -q eb -Q 2 bsortin -o bsortout
```

2. Example of specifying bsortex command

```
bsortex -sort key=20.8ebca -input reclen=100 file=bsortin -output file=bsortout -option icode=au
iconv=1
```

Note

The following processing cannot be performed.

- Arrange the field of the EUC in order of the EBCDIC/JEF code.
- Arrange the field of the EBCDIC/JEF code in order of the EUC.
- Arrange the field of the Unicode (UCS-2/UTF-32/UTF-8) in order of the EBCDIC/JEF code.
- Arrange the field of the EBCDIC/JEF code in order of the Unicode (UCS-2/UTF-32/UTF-8).

B.7 Important Notes about the File Types

- If an inappropriate type of file is specified, it may cause an error.

- Even if the type of file is different between the input file and the output file, the record format is the same in both the input and the output files.
- When the input file is a text file with the EOF (End of File) code, the EOF code is not recognized as end of the file. However, the EOF code can be recognized as end of the file in the bsortex command. Refer to the eof operand of input file information option (-input) for details.
- When the input file is a text file with the EOF code and 'effect' is specified by the eof operand of the bsortex command, add the EOF code at the end of output file. However, the EOF code of the output file can be deleted in the bsortex command. Refer to the removeeof operand of output file information option (-output) for details.
- When the input file is a text file with the EOF code and 'effect' is specified by the eof operand of the bsortex command,
- Even if there are two or more input files with the EOF code, only one EOF code is added per output file.

B.8 Important Notes about the NetCOBOL File system

Points of concern about sharing and exclusion of the NetCOBOL file

When the NetCOBOL file is specified for an input file or an output file, PowerBSORT inputs and outputs the NetCOBOL file by using the NetCOBOL file system. Therefore, sharing and the exclusion of the file are controlled by the NetCOBOL file system. Open in the sharing mode when the input file is a NetCOBOL file. Open in the exclusion mode when using an output file. Refer to the manual of the NetCOBOL for details.

Points of concern about file access library for NetCOBOL sequential files and NetCOBOL physical sequential files

Two kinds of libraries are prepared in the file access library for a NetCOBOL sequential file and a NetCOBOL physical sequential file. The outline of processing of each file access library is as follows. Refer to the manual of NetCOBOL for details of BSAM in NetCOBOL.

File organization	File access library	Outline of processing
sequential file	libbscblrt64.so	Use the NetCOBOL file system to open, to close, and to input and output the file. It is low-speed because it inputs and outputs one record at a time.
	libbscblfast64.so	Use the NetCOBOL file system to open and to close the file. Use the read system call and the write system call to input and output the file. It is high-speed because it inputs and outputs bringing two or more records together. It is similar to BSAM in NetCOBOL.
physical sequential file	libbscblps64.so	Use the NetCOBOL file system to open, to close, and to input and output the file. It is low-speed because it inputs and outputs one record at a time.
	libbscblpsfast64.so	Use the NetCOBOL file system to open and to close the file. Use the read system call and the write system call to input and output the file. It is high-speed because it inputs and outputs bringing two or more records together. It is similar to BSAM in NetCOBOL.

These file access libraries can be used properly by defining BSORT_FILESYS_fs of a startup file.
However, the following file access libraries can be used without defining BSORT_FILESYS_fs of a startup file when the following identifiers (bsort command or bsortex command) or the definition values (BSORT function) are specified.

File organization	Identifier	Definition value	File access library used
sequential file	cobs64	BSFS_COBS64	libbscblfast64.so
physical sequential file	cobp64	BSFS_COBP64	libbscblpsfast64.so

About NetCOBOL Line Sequential files

The NetCOBOL line sequential file can be processed as a text file in PowerBSORT.
Specify it as follows when you process the NetCOBOL line sequential file. However, note that file sharing and exclusive control of the NetCOBOL file system is not executed.

- File system specifies "Native file system of the system".
  - For the bsortex command
    filesys operand of Input file information option (-input) or filesys operand of Output file information option (-output)
  - For the bsort command
    I/O file system option (-F)
  - For the BSORT function
    inpfsys_tbl and outfsys of BSRTFILE structure
- The input record format specifies "Text file fixed field specification".
  - For the bsortex command
    recform operand of Input record information option (-record)
  - For the bsort command
    Text file option (-T)
  - For the BSORT function
    fieldmode of BSRTPRIM structure

B.9 Important notes about functions that can cause irregular results

Sort option

In sort option, the order of outputting the record with the key field of equivalence is irregular.

The record where the value of the key field is the same can be output to input sequence by using the FIFO option.

See

- Environment variable BSORT_FIFO
- BSORT_FIFO of startup file
- FIFO option (-f)
- fifo operand of execution environment option (-option)
- optionfunc of BSRTPRIM structure in bsrtopen function

Record summation option

In the record summation option, the summation record is irregularly output from the record of the same key field.

In the bsortex command, the summation record can be specified by the first operand or the last operand of the record summation option.

See

- first operand of record summation option (-summary)
- last operand of record summation option (-summary)

Suppression option

In the suppression option, the output record is irregularly output from the record of the same key field.

In the bsortex command, the output record can be specified by the first operand or the last operand of the record summation option.

 See

- first operand of record summation option (-summary)
- last operand of record summation option (-summary)

Glossary

In core sort

When all input records are entered in a memory field, they are sorted within the memory field without using a temporary file.

Key field

A field in a record used to decide size comparison. It consists of information on location, length, data format and operation method of the field.

Fixed field

A field divided by a byte position. The position of the field is common in all records.

Copy option

This option copies input files into output files. It can be used by simply connecting input files or selecting records to copy.

Reconstruction field

A field in an input record specified in the record reconstruction option. It consists of information on location and length of the field in the record.

Fist-in first-out (FIFO) option

When sorting files or records having several key fields with the same value, the record that is first entered is first output. This option is used in combination with the sort option.

Suppression option

When sorting or merging files or records that have several key fields with the same value, this option keeps 1 record and deletes all the others. This option is used in combination with the sort option or the merge option.

Literal value

Constants of the character-string, the decimal number or the hexadecimal number that is used for the record selection option and the record reconstruction option.

Summation field

The field for numeric that is used in the summation option. A summation field consists of information on location, length, data format, and output form of the field.

String

A set of records that have been sorted out.

Selection field

A field used as a condition of the records to be processed.

Sort option

This option sorts records specified by more than 1 key field, either in ascending order (0 to 9 and A to Z) or descending order (Z to A and 9 to 0). Normally, the key field is specified when records are sorted. PowerBSORT rearranges records based on the specified key fields.

Field

Fields include the key field used in the sort option and merge option, selection field used in the record selection option, reconstruction field used in the record reconstruction option and the summation field used in the record summation option.

Field separation character

It is a character string that divides the field in the floating field.
Blanks and tabs are used as "field separation characters" as default value. In addition, arbitrary character strings that can be entered at the keyboard and arbitrary codes specified in hexadecimal number can also be used as the field separation characters. Make sure not to use the same characters as record separators.

Floating field

A field that can be divided by field separation character. A field position is specified in field number that is divided by field separation character. Because the field position is decided by the position of the field separation character, the position of the field (column position) is different in each record.

Merge option

This option merges two or more sorted files into 1 file. It guarantees the same order of the key fields as the sort processing.

Record selection option

This option selects only needed fields by specifying the selection method for the target records. It is used in combination of the main option. The record separation option includes the following 2 methods: method to compare the comparing field with the compared field and method to compare the compared filed and the literal value. PowerBSORT compares the specified 2 fields by the comparing method and selects the fields.

Record reconstruction option

This option changes locations of fields or embed literal values in a record. When reconstructing the records, specify the field margin in order, from left to right. To pack the field of input records, specify the location or the length of the field. To pack the literal value, specify the literal value.

Record summation option

When sorting or merging files or records that include several key fields with the same values, the values in the summation field are added to make 1 record.

Record separator

A record separator can be selected from one of the followings: CRLF (carriage return followed by a line feed), CR (carriage return) and LF (line feed). General text files can be separated by LF.

BOM

BOM (Byte Order Mark) is a specific code filled in on the head of the file to specify the endian of Unicode.

Big endian

When describing integer with several bytes, the high-order byte of the number is stored in memory at the lowest address.

bsrt.h

The header file in which the information used by BSORT function is specified. The header file is located in the include directory that is created in the PowerBSORT installation.

Little endian

When describing integer with several bytes, the low-order byte of the number is stored in memory at the lowest address.

Unicode

An international character-code used universally. In PowerBSORT, the following formats are supported.

- UCS-2 form
  A form to describe 1 character in 2 bytes.

- UTF-32 form
  A form to describe 1 character in 4 bytes.
- UTF-8 form
  A form to convert 1 character to 1 to 6 bytes of variable-length code. In PowerBSORT, 1 to 3 bytes are supported.

Index

[W]