DataGeneral

ECLIPSE® -LINE SORT AND MERGE UTILITY PROGRAMS

User's Manual

093–000126–02

ECLIPSE®-LINE

SORT AND MERGE

UTILITY

PROGRAMS

User's Manual

093–000126–02

Ordering No. 093-000126

Printed in the United States of America
Rev. 02, March 1976

CHAPTER 8 Continued

CHAPTER 9 EXECUTING THE MERGE UTILITY PROGRAM

APPENDIX A OCTAL AND HEXADECIMAL CONVERSIONS

APPENDIX B ASCII CHARACTER CODES

APPENDIX C SORT AND MERGE SYSTEM ERROR CODES

APPENDIX D NUMBER REPRESENTATIONS

CHAPTER 1

INTRODUCTION TO THE SORT UTILITY PROGRAM

GENERAL

The Data General Sort Utility is a general purpose sorting program that orders files of logical data records according to a predetermined collating sequence. The program accepts any file that can be handled by the operating system - with a minimum of time and intervention. The only practical limit to the number of files that can be input to one sort application is the amount of available disk space. Input and output files can reside on any physical device known to the operating system.

Multiple input files need not have the same characteristics. The files can have fixed-length, variable-length, undefined, line, or data sensitive record formats - blocked or unblocked. Sort will bypass any number of records at the beginning of each input file, and the maximum number of records to be sorted for each input file can also be specified. INFOS system files are fully supported.

Output records can be either the input record in its original form, or selected portions of the record. Constants can be inserted into the output when desired. The output records can be fixed-length, variable-length, undefined, line, or data sensitive; either blocked or unblocked.

The contents of the sort keys determines the sequence of the records to be sorted. There is no limit to the number of keys that can be handled. Key types supported include character, unsigned bit strings, unsigned zoned decimal, and unsigned packed decimal. Ascending or descending sequence may be specified for each individual key. If identical keys are encountered, their order of occurrence is not preserved.

Binary keys are not restricted to starting and ending byte boundaries; each such key can be specified to the bit level. In addition, the start or end of a field can be specified as a character to be scanned for, instead of an absolute location. This option is extremely helpful for special Sort applications (i.e., sorting source listings, etc.).

Output of the Sort application can be a complete record sort, TAGOUT, TAG, record keys, or an inverted INFOS (ISAM) index. Although complete record sorts are often faster than TAG sorts, with limited disk space the TAG feature sorts records which could not otherwise be handled.

Two work files can be specified. Each file must be able to hold all the sort data and keys and tags as specified. The second work file is only used if an intermediate merge is necessary prior to the final merge. This is of particular advantage if a contiguous area cannot be found for one work file large enough to hold twice the amount of data. Also, when two disk drives are available, head movement during the sort phase can be minimized when there is an intermediate merge phase. Work file space can be randomly or contiguously allocated. When two work files are specified, they do not have to have the same format.

The Sort Utility program consists of a number of distinct, sequentially executed phases, and the same phases are used for every Sort application. During execution the program chooses the correct sorting technique based on control statement information supplied by the user.

Control statements are designed with completely defaultable specifications; but, these should be used with care. Operand values need be entered only if something other than the default values are desired. Only the IN, OUT, KEY, and END control statements are required for any successful Sort execution.

The control statements can be entered dynamically via the system console, or through another system supported device. If the system console is used to input the control statements, all control statement syntax errors are immediately displayed on the system console. In most cases, the user can correct, add, delete, or modify the control statements through the system console before executing the Sort. It is recommended that the user always maintain a listfile (listfile name/L) to facilitate corrections.

Numeric operand values in control statements can be input in any one of the following radices: decimal, octal, hexadecimal, or binary.

Provision is made to include externally coded routines into the sort to handle non-standard situations. Thus, the user can take explicit responsibility for input, output, and key comparisons.

END OF CHAPTER

CHAPTER 2

EXPLANATION OF SORT CONTROL STATEMENTS

GENERAL

Sort control statements supply the Sort Utility program with information required to identify and initiate the sort application. These control statements define control field parameters, specify information about the characteristics of the input and output files, key fields, record and data selection criteria, and the various options that were selected for the sort operation.

COMMAND WORDS

Each control statement begins with a Command Word. The eight unique Command Words and an explanation of each of their functions is as follows:

Command Word	Definition
GLOBAL (Optional)	Describes the overall environment and specifies the general options to be used for the particular sort application.
KEY (Required)	Specifies one or more keys to be used to order the files during the sort. Multiple keys are entered in order of major to minor precedence.
SEL (Optional)	Permits the user to delete and/or rearrange data from the input record to be used as output. This control statement can also be used to specify constant data to be inserted in the output record in addition to the data from the input record. When used, this phase occurs after sort is complete.
IN (Required)	Defines the input files to be sorted. More than one file can be described using a single IN control statement as long as the files being described have the same characteristics (logical record length, blocking factor, record format, access method, etc.).
OUT (Required)	Defines the output file specifications. The output file does not necessarily have to bear any relationship to the characteristics of the input file(s).
WORK1 (Optional)	Describes the characteristics of the primary work file to be used for the sort application.
WORK2 (Optional)	Depending on available core, available disk space, number of records to be sorted, etc., it may be desirable to assign a supplemental work file. This control statement describes the characteristics of the second work file.
END (Required)	Indicates the end of the control statements. The END statement signals the sort utility to start execution of the sort.

CONTROL STATEMENT FORMAT

A control statement begins with a Command Word, followed by a series of operand fields. The operand fields are based on the keyword concept, whereby keywords are entered followed by an alphanumeric or numeric value. Each keyword and its correspondng value are linked together by an equal sign.

The Command Word and related operand fields are separated from each other by at least one space, comma, or tab. The separators can be used freely as well as intermixed. Separators can also surround the equal signs and right and left parentheses, if desired.

In the following example, the control statements are all syntactically correct:

```
GLOBAL TYPE=REC USERKEY=YES)
GLOBAL, TYPE=REC USERKEY=YES)
GLOBAL TYPE = REC, USERKEY=YES)
```

Note that most control statements are designed for completely defaultable specifications but these should be used with care. Therefore, Command Words and their related operands need only be entered if options other than default values are desired. Only the IN, OUT, KEY, and END Commands are required for any particular sort application.

End And Continuation Lines

A control statement line ends with a TERMINATOR (carriage return (), form feed, or null). Control statements can be continued from one line to the next by means of a ↑(shift N) IMMEDIATELY preceding the TERMINATOR.

For example, the following are valid control statements:

```
GLOBAL,↑
TYPE = REC, USERKEY=YES)
        (or)
GLOBAL,↑)
TYPE=REC,↑)
USERKEY=YES)
```

As can be seen from the above examples, there is no practical limit to the physical size of a control statement, but 1300 bytes must not be exceeded or an error results.

Control Statement Corrections

When more than one control statement is encountered, the last control statement of the same type entered is used. This permits the user to make immediate corrections to the control statements from the system console device.

Example:

(first entry)

```
GLOBAL, TYPE=REC, USERKEY=YES)
```

(second entry)

```
GLOBAL, TYPE=TAG, USERKEY=YES)
```

In the example above, the second control statement overrides the first control statement entry.

When more than one GLOBAL, WORK1, or WORK2 control statement is entered and the succeeding entry consists merely of a Command Word, all previous entries for that control statement are ignored. All values previously entered for the control statement are reset to the default values.

Example:

(first entry)

```
GLOBAL,TYPE=TAG,WRTKEY=NO)
```

(second entry)

```
GLOBAL)
```

In the example above the second GLOBAL control statement overrides the Keyword values in the first entry. The Sort Utility resets the GLOBAL operands to the default values (TYPE=REC,WRTKEY=YES, etc.).

Order of Operands

Although the Command Word must be the first entry in a control statement, operands present in GLOBAL, IN, OUT, WORK1, and WORK2 control statements need not appear in any predefined order.

Example:

```
GLOBAL,TYPE=TAG,USERKEY=YES )
```

can also be entered as:

```
GLOBAL,USERKEY=YES, TYPE=TAG)
```

Numeric Values Used In Operands

All numbers in the value portion of an operand can be entered in either decimal, octal, hexadecimal, or binary radix. This is accomplished by preceding the number with the proper modifier. The modifiers are:

D	-	Decimal
O	-	Octal
X	-	Hexadecimal
B	-	Binary

If a modifier is not used, the default is decimal.

Examples:

```
MAX=50
MAX=D50
MAX=O62
MAX=X32
MAX=B110010
```

All equivalent

Boolean Values Used In Operands

Additionally, boolean values can be entered as either TRUE, T, YES, Y or 1 for true, and FALSE, F, NO, N or 0 for false.

Examples:

```
USERKEY=YES
USERKEY=Y
USERKEY=TRUE      } All equivalent
USERKEY=T
USERKEY=1
```

CONTROL STATEMENT NOTATION

The following is a definition of the notation used to describe the coding of control statements.

1. Upper case characters and numeric values must be entered exactly as shown.
2. Left and right parentheses and the equal sign must be entered exactly as shown.
3. Brackets [ ], braces { }, elipses ..., lower case characters, and underlines are used as example notations and must not appear in the actual control statement.

[ ]	Square brackets indicate that the enclosed items are optional, and not required items.
{ }	Braces indicate that at least one of the operand values must be chosen.
_	Underlined values indicate the operand default values.
number (lower-case)	All integers in the value portion of an operand field can be entered preceded by a modifier. The modifier values are D (decimal), O (octal), X (hexadecimal), and B (binary). If a numeric value is not preceded by a modifier, the default is decimal.
YES or NO	Boolean values YES and NO can be entered as TRUE, T, YES, Y or 1 for the value of true; FALSE, F, NO, N, or 0 for the value false.
Commas	Commas are used in the notation examples as SEPARATORS. Commas in the actual control statements can be replaced by spaces or tabs.

Notation Example

```
[GLOBAL,
                  { REC    }
                  { TAG    }
                  { TAGOUT }
[TYPE =           { KEY    } ],
                  { TKY    }
                  { INV    }

[PAD = { 0      } ]
       { number }

[WRTKEY = { YES } ],
          { NO  }

[USERKEY = { NO  } ],
           { YES }

[FLOAT = number],

[SPACE = { 0      } ]]
         { number }
```

In this example the entire control statement is optional (as indicated by the beginning and ending square brackets). The default value for each Keyword is underlined. In the instance where the user wished to use all of the default options, the entire GLOBAL control statement could be omitted. If, however, the user wished to substitute NO in the WRTKEY keyword instead of the default value YES, the control statement could be entered as:

```
GLOBAL,WRTKEY=NO)
```

and the control statement would be executed as if the following had been entered:

```
GLOBAL,TYPE=REC,PAD=0,↑)
WRTKEY=NO,USERKEY=NO,SPACE=0
```

END OF CHAPTER

CHAPTER 3

SORT CONTROL STATEMENTS

GLOBAL CONTROL STATEMENTS

```
[GLOBAL,

[TYPE=  { REC  }      ],
        { TAG  }
        { TAGOUT }
        { KEY  }
        { TKY  }
        { INV  }

[PAD=   {0     }      ],
        {number}

[WRTKEY=  {YES}       ],
          {NO }

[USERKEY= {NO }       ],
          {YES}

[FLOAT = number],

[SPACE=  {0     }     ]]
         {number}
```

NOTE: If the SPACE= option is used, the GLOBAL control statement must be the first control statement input. The default for the FLOAT option is no float.

General

GLOBAL is an optional control statement that specifies general options for the Sort application. These operands are contained in the GLOBAL control statement because they cannot logically be placed in any of the remaining control statements.

GLOBAL

Command Word

TYPE=

```
[TYPE=  { REC  }    ]
        { TAG  }
        { TAGOUT }
        { KEY  }
        { TKY  }
        { INV  }
```

Specifies the output of the Sort application. The default value is REC (record sort).

REC (Record Sort)

A Record sort is one in which the entire input record, or selected fields of the input record, and the key are formed into a sort record. The record is then carried through all of the intermediate phases of the sort and eventually becomes the actual output record (possibly excluding the key field). A record sort is valid with all file types and record formats.

TAG (Tag Sort)

A tag sort differs from a record sort in that a pointer to each input record is constructed in conjunction with the key field which is extracted from the input record. This compact form of the input record is then used in all of the intermediate phases of the sort, primarily to conserve the amount of space required for the sort work files. During the final phase of the sort the "tags" are used to retrieve the original input records from which the output record is selected. Tag sorts are valid only with INFOS SAM Disk Files.

TAGOUT

The TAGOUT Sort type is used to provide the user with a mechanism for outputting the internal record tag as the final output record. The format of the tag is as follows:

3 words of INFOS feedback
2 words of logical record number
1 word of record length information

TAGOUT is legal with INFOS SAM input files only.

KEY

The KEY Sort type is used to indicate that the output records from the final phase of the sort are to be the key fields which were selected in the KEY control statement. The output records contain the keys used internally by the Sort and at times may not be identical to the specified keys. This may occur when signed numbers, descending collating sequence, and non-aligned keys are specified. However, the internal key is suitable for INFOS use as INFOS keys are unsigned. This is the only type of output available from a KEY Sort.

TKY

The TKY Sort type enables the user to get a combination TAGOUT KEY Sort output information. The format of the output is the KEY output with the standard TAGOUT format appended.

INV

The INV Sort type is provided to facilitate the optimal addition of records to a secondary single level index for a DBAM type of file. The input file for this type of sort must be an Index file and Data Base file for an ISAM or DBAM file. The records of that index file will be retrieved sequentially and used as the input to the standard type of sort processing. In the final phase of the sort it opens the new index file for the existing Data Base file and then writes each of the new keys into the Index file and utilizes the feedback as the link to the original data Base record. This facility is provided to create indexes because the case for optimized index creation occurs when the keys are introduced sequentially. This optimizes both the space requirements for the Index file and the speed with which it can be created. Note, however, that the keys used for Sort must be acceptable to INFOS. (See the description of REC= in the OUT CONTROL STATEMENT later in this Chapter.)

PAD=

[PAD= {0 / number}]

This operand specifies the character to be used for variable-length record padding. The default value is 0 (decimal value of ASCII null); but you should use an octal 40 (blank) if possible with RDOS line type records and INFOS data sensitive records, or other record types when they are to be printed later.

WRTKEY=

[WRTKEY= {YES / NO}]

This operand can be specified when disc space is at a premium as it saves Workfile space. The sort extracts an external sort key when records enter the system. In most cases, the external key is carried along with the record for the remainder of the sort (except for single alphanumeric keys). The NO option can be used to eliminate writing of the sort key to the work file. This option should not be used if TYPE= does not equal REC.

USERKEY=

[USERKEY= {NO / YES}]

This operand indicates whether key comparisons are to be handled by the Sort Utility or by the user. The KEY control statement may or may not be included with this option. The default value specifies that key comparisons are handled by the Sort Utility program.

USERKEY = YES implies that the user has included the KEYCHK entry point (see the chapter titled "USER CODE").

FLOAT=

The number specified for this option represents the octal equivalent of the ASCII character to be used as the last character of the record. This can be useful to obtain fixed length output from variable length input. For example:

```
GLOBAL    PAD   = 040↑)
          FLOAT =  015↑)
```

causes a) (carriage return) to be placed at the end of each record, and makes the records printable.

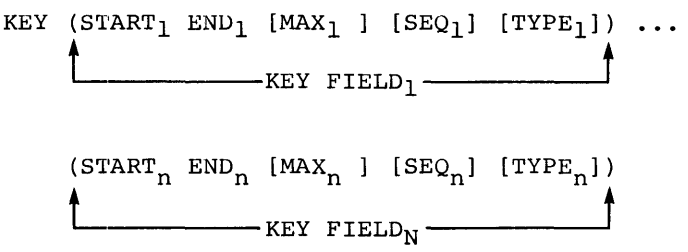
SPACE=

When Sort is invoked it takes all the space available in the partition if the partition is less than 32K words. If greater than 32K words, Sort takes a maximum of 32K. Some additional space is required when processing INFOS files. If the SPACE= option is not specified or is specified as SPACE=Ø then the area required for INFOS files is taken from memory not allocated to Sort. When SPACE= number is specified, Sort still takes space as described above but then subtracts number from that space. The number specified cannot exceed 16K words.

The procedures for determining INFOS file space requirements are described in the Planning Manual. Note that only the input or the output file is open at any given time, both files are never open simultaneously.

NOTE: If this operand is used, the GLOBAL control statement must be the first control statement entered.

KEY CONTROL STATEMENT



General

KEY is a required control statement that specifies the control field(s) (keys) by which the records are to be ordered. There is no maximum limit to the number of keys that can be handled by the Sort Utility. Keys are entered in the Key Control Statement in major-to-minor precedence. Each key field must be enclosed by parentheses. Note that the Key Control Statement does not use the keyword concept.

KEY

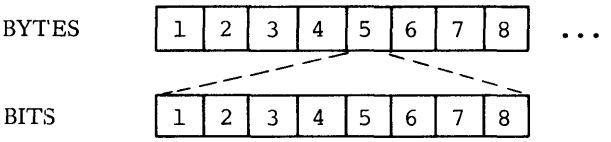
Command Word

Byte And Bit Specifications

In the following discussion, a BYTE is one-half of a 16-bit computer word; each byte consists of eight bits.

A byte can store one ASCII character, 2 packet decimal digits, or an unsigned binary number.

Bytes and bits are numbered from left to right, with byte/bit number 1 being the left-most position. Bytes are packed from left-to-right, and all bytes and bits are 1-relative.



Byte Boundaries

An unsigned binary key does not necessarily have to start and end on a byte boundary. If a key does not start or end on a byte boundary it's called a non-aligned key. Non-aligned key locations are indicated by the inclusion of a colon (:) between a byte and bit specification. Separators are not permitted on either side of the colon. Non-aligned keys are used for unsigned bit string keys.

```
5:3          (byte 5, bit 3)
```

A key address, therefore, can be specified as:

The default for the bit position depends on whether the START or END of the key is being specified.

1. Default for the bit position of the START specification is 1:

```
(START       implies 5:1 (byte 5, bit 1)
  ↓
  5
```

2. Default for the bit position of the END specification is 8:

```
END)         implies 7:8 (byte 7, bit 8)
 ↓
 7
```

When the specified bitposition is greater than 8, the byte and bit fields are adjusted to modulo 8:

Example:

```
3:15
```

is converted to 4:7 (byte 4, bit 7).

START (Required)

The START specification is mandatory, and can be entered in one of three ways:

1. ABSOLUTE LOCATION - as previously explained in BYTE BOUNDARIES.

Examples:

```
5            (implies byte 5, bit 1)
5:3          (byte 5, bit 3)
3:15         (byte 4, bit 7)
```

2. CHARACTER - The user can enter a character to be scanned for and when located is used as the starting location of the key starting field. The character to be scanned for must be entered preceded by a quotation mark. A beginning angle bracket (<) must not be used as a scan character.

```
(START
  ↓
 ("B
```

The character to be scanned for has an implied bit position of 1.

3. CHARACTER REPRESENTATION - Characters to be scanned for can be entered by using any allowed numeric representation of the ASCII code. The code is preceded by a quotation mark, and the value is entered by enclosing the value in angle brackets (default is decimal if not preceded by a modifier entry).

   Example:

   Scan for a carriage return as the starting location:

```
(START
   ↓
 ("<13>
```

   Where the value 13 is the decimal representation of the ASCII Carriage Return code.

END (Required)

The END specification can be entered as:

1. ABSOLUTE LOCATION - Used basically in the same manner as the START specification.

   Examples:

```
11:3   (byte 11, bit 3)
8:15   (byte 9, bit 7)
15     (in the END specification, the implied
        bit position is 8)
```

2. BYTES AND BITS FROM THE START LOCATION - This indicates that the number of bytes and bits specified in the END specification are added to the START location to form the key. This entry must be preceded by @.

   Examples:

```
(START       END)
   ↓          ↓
(  5         @4 )
```

   implies the key starts at byte 5 (bit 1) and ends at byte 8 (bit 8).

3. CHARACTER - The user can also enter a character to be scanned for and used as the END location of the key. The character must be preceded by quotation marks.

   Example:

```
(START       END)
   ↓          ↓
( "A         "G )
```

   specifies that the character A will be scanned for and used as the beginning of the key. "G" indicates the character G will be scanned for and used as the end of the key. A beginning angle bracket (<) must not be used as a scan character.

4. CHARACTER REPRESENTATION - Characters to be scanned for can be entered by using any allowed numeric representation of the ASCII code. The code is preceded by a quotation mark, and the value is entered by enclosing the value in angle brackets (default is decimal if not preceded by a modifier entry).

   Example:

   Scan for an angle bracket as the starting location:

```
END
 ↓
"<60>)
```

   Where the value 60 is the decimal representation of the ASCII beginning angle bracket.

5. EOR - EOR can be entered to indicate end-of-record, and is particularly useful when variable-length records are used. In the following example:

```
(START       END)
   ↓          ↓
( "A         "G )
```

   Each variable-length record in the file is sorted using as its key byte 1, bit 1 to end-of-record.

[MAX] (Optional)

MAX is an optional entry used to indicate that the size of a variable-length key is limited. When used, this option reduces the amount of main memory required and greatly speeds up the sorting process. The specified value must be preceded by #.

For example, if the maximum size of a variable-length record is 1,000 characters, the entry:

```
(START          END
   ↓             ↓
( "A            "G
```

Implies that the key starting location is the first occurrence of A; G is scanned for as the end of the key up to the end of the record.

If, however, the following is entered:

```
(START          END          [MAX])
   ↓             ↓            ↓
(" A            "G           #15   )
```

The #15 specifies that the key will end at the first occurrence of G, or 15 characters from the first occurrence of A, (includes the character A), whichever is lesser.

If MAX is not used a key area equal in size to RECORD SIZE is reserved. If MAX is used a key area equal to MAX is reserved which saves core and makes sort run faster.

[SEQ] (Optional)

Either ascending or descending sequence may be specified for each key.

```
[SEQ]           [SEQ]
  ↓               ↓
  A               D
```

The default value is ascending sequence.

For restrictions on the use of descending sequence see the following page.

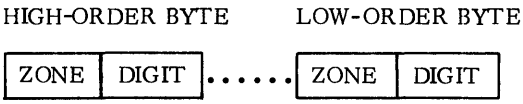
[TYPE] (Optional)

The key field can be a character string, unsigned bit string, unsigned zoned decimal, or unsigned packed decimal number. The default value is character string.

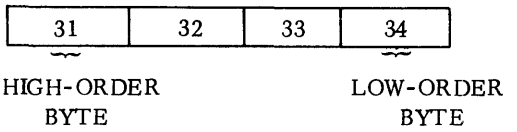
The allowable values for TYPE are as follows:

Entry	Description
C	Character string (default)
B	Unsigned bit string
UZ	Unsigned zoned decimal
UP	Unsigned packed decimal
F	Leading overpunch
E	Trailing overpunch
L	Leading
T	Trailing
P	Signed packed (trailing)
S	Signed binary

The format of an unsigned zoned decimal field is as follows:

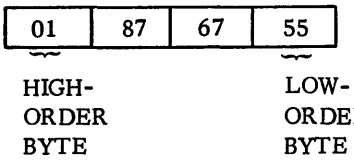


where 1234 is represented as



The format of an unsigned packed decimal field is as follows:

where 1876755 is packed as:



NOTE: Descending key sequence in SEQ can only be specified for character strings (C) and unsigned zoned decimal (UZ). If descending sequence is specified for unsigned bit strings or unsigned packed decimal key, unpredictable results may occur.

Examples of KEY Control Statements

1. KEY (5 5)

KEY is one byte long (byte 5, bit 1 to byte 5, bit 8); default is ascending sequence; character string. This can also be specified as (5:1 5:8).

2. KEY ("A 10)

First occurrence of A to byte 10, bit 8; ascending sequence; character string.

3. KEY (8:3 EOR D) ...

Byte 8, bit 3 to end of record; descending sequence; character string.

4. KEY (2 @6) (12 @4) (32:4 32:6 D UZ)

a. First Key - Byte 2, bit 1 to byte 7, bit 8; ascending sequence; character string.

b. Second Key - Byte 12, bit 1 to byte 15, bit 8; ascending sequence, character string.

c. Third Key - Byte 32, bit 4 to byte 32, bit 6; descending sequence; unsigned zoned decimal.

SEL CONTROL STATEMENT

SEL (START$_1$ END$_1$) ... (START$_n$ END$_n$)

General

This control statement is optional. It permits the user to delete and/or rearrange data from the input record to be used as output. This control statement can also be used to specify constant data to be inserted in the output record in addition to the data from the input record. SEL can only be used for record and TAG sorts (TYPE=REC or TYPE=TAG in the GLOBAL control statement). If this control statement is omitted, the default is that the entire input record is output.

SEL

Command Word

Data From The Input Record

Data to be selected for output from the input record is indicated in the same manner as the format of the START, END, and MAX operands in the KEY control statement. The SEQ and TYPE specifications are not used.

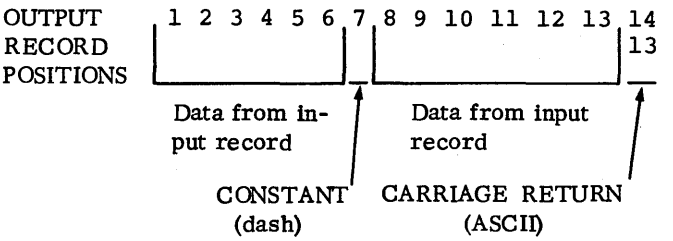
Literal Option

Constant data to be included in the output record must be enclosed by apostrophes. The constant data is inserted into the output record exactly as it is coded in the SEL control statement. Non-printable characters can be inserted by enclosing them in angle brackets (preceded and ended by an apostrophe). Note that this option adds characters to those selected. Consequently, the user must be sure the total number of characters to be output is equal to the output record size or, for an RDOS line record, equal to or less than 132 characters.

Example:

```
SEL (5 @6) '-' (50 55) '<13>'
```

causes the format of the output record to be output as:



NOTE: Angle brackets, when used as constants, must be represented by their ASCII code values.

IN CONTROL STATEMENT

```
IN,
FILE = filename1 [+filename2 + ...
                 + filenamen]
[SEQ = number ],
[TYPE = { RDOS | SAM | ISAM | USER } ]
[FMT = { L | F | V | U | D } ]
[REC = { 132 | number } ]
[BLK = { @1 | @number | number } ]
[SKIP = { 0 | number } ]
[MAX = { all records | number } ]
```

General

IN is a required control statement that names and describes the input file(s) to be sorted. Only the FILE = filename operand is mandatory; all other operands are optional. If more than one file is to be input to Sort, and the files have different characteristics, additional IN control statements must be entered, and must be entered using the SEQ operand.

IN

IN (Command word)

FILE=

```
FILE=filename1 [+filename  + ...
               + filenamen]
```

More than one filename can be entered in a single IN control statement as long as all files have the same characteristics. Thus, if FILE1, FILE2, FILE3, and FILE4 have the same characteristics, then the control statement can be entered as:

```
IN FILE=FILE1+FILE2+FILE3+FILE4,
                REC=80, .........
```

NOTE: For each inversion an INFOS system ISAM or DBAM filename must be entered as Indexname plus Databasename.

SEQ=

```
[SEQ=  number ]
```

This entry is used when multiple IN control statements are used in a sort application. It specifies the order in which the input files will be processed by the sort program.

Example:

```
IN FILE = CUST, REC=80,...SEQ=2
IN FILE = ACCOUNT, REC=80,...SEQ=1
IN FILE = DEDUCT, REC=60,...SEQ=3
```

TYPE=

```
[TYPE= { RDOS | SAM | ISAM | USER } ]
```

Identifies the type of access method to be used to read the Sort input file(s). Default value is RDOS.

FMT=

```
[FMT= { L | F | V | U | D } ]
```

Specifies the record format of the records in the input file(s).

L	(RDOS ASCII line)*
F	(fixed-length)
V	(variable-length)
U	(undefined)
D	(data sensitive)

The default value is L (RDOS ASCII line).

*Record must not contain non-ASCII characters.

When fixed-length records are processed, all data records in the file must have the same length. When variable-length records are processed, the length of the data records in the file can vary.

The undefined-length record format is primarily provided for processing records created in some other operating system environment. When a file of this type is processed, the length of each data record in the file is not predetermined.

When data sensitive records are processed, the records in the file can also vary in length. Records are delimited by user-selected characters that indicate the end of each record. The normal record delimiters for this type of record are carriage return, form feed, and null.

For further information concerning the use of variable-length, undefined, and data sensitive records, see the Data General publication entitled "INFOS System Planning Manual", 093-000115.

Only fixed-length and line record formats can be specified for RDOS files; fixed-length, variable-length, undefined, and data sensitive for SAM files; and variable-length for ISAM files. All record format types can be used with USER files.

	FILE TYPES			
Record Format	RDOS	USER	ISAM	SAM
Fixed-length	YES	YES	NO	YES
Variable-length	NO	YES	YES	YES
Undefined	NO	YES	NO	YES
Line	YES	YES	NO	NO
Data Sensitive	NO	YES	NO	YES

REC=

[REC= {132 / number}]

Specifies the logical record length of records in the file. This entry is ignored for RDOS Line-type records (FMT=L). When variable-length, undefined, or data sensitive records are being described, this entry must be coded and also must indicate the length of the longest record in the file. When TYPE = INV in the Global control statement, record length equals minimum node size which, usually, is page size minus six bytes. The default value can only be used for fixed-length records.

BLK=

[BLK= {@1 / @number / number}]

Specifies the blocking factor for blocked records. This value can be specified in terms of:

1. The number of BYTES in a block:

   number

2. The number of RECORDS per block:

   @number

The default value is one RECORD per block:

@1

The following rules apply when coding BLOCK:

1. When FMT=U, V, or D the block length must be entered and the value specified in bytes. Thus, @1 and @number must not be used.

2. BLK is ignored for RDOS Line-Type records (FMT=L), and should not be coded.

3. When TYPE=INV in the Global control statement, block size should be equal to page size or be some multiple of 512 bytes.

SKIP=

[SKIP= {0 / number}]

This operand is used to specify the number of records to be skipped at the front of the file (i.e., labels, control records, etc.). The default is zero records.

MAX=

[MAX= {all records / number}]

This entry specifies the maximum number of records to be extracted from the file.

The default value indicates that all the records in the file will be used as input (except for the records specified in the SKIP operand).

OUT CONTROL STATEMENT

```
OUT,
FILE = filename,
[CREATE  {YES}
         {NO }  ],
[TYPE=   {RDOS}
         {SAM }
         {ISAM}
         {USER}  ],
[FMT=    {L}
         {F}
         {U}
         {V}
         {D}  ],
[REC=    {132   }
         {number}  ],
[BLK=    {@1     }
         {@number}
         {number }  ],
[ORG=    {S}
         {R}
         {C}
         {D}  ],
[OWRITE= {YES}
         {NO }  ]
```

General

OUT is a required control statement that names and describes the Sort output file.

OUT

OUT (Command word)

FILE=

```
FILE=filename
```

Specifies the name of the file to which the output records are written. The output file may or may not exist before the Sort application is executed (see the explanation of the CREATE operand below).

CREATE=

```
[CREATE=  {YES}
          {NO }  ]
```

This operand specifies whether or not the user wishes to create the output file.

If CREATE = YES is used and the file exists it is overwritten. CREATE = YES is the default value.

TYPE=

```
[TYPE=   {RDOS}
         {SAM }
         {ISAM}
         {USER}  ]
```

Identifies the type of output file. Default value is RDOS.

ISAM is valid only for INVERT sorts.

FMT=

```
[FMT=    {L}
         {F}
         {V}
         {U}
         {D}  ]
```

Specifies the record format of the records in the output file:

L	(RDOS ASCII line)
F	(fixed-length)
V	(variable-length)
U	(undefined)
D	(data sensitive)

Remember the following when coding FMT:

1. FMT=F must be coded if the output of the sort is from a TAGOUT, KEY, TKY, or INVERT sort type (see GLOBAL).

2. Output delimiters from the sort for data sensitive record format (FMT=D) are carriage return, form feed, or null only. Make sure that data sensitive and line type records have delimiters.

The default value is L (line).

Only fixed-length and line record formats can be specified for RDOS files; fixed-length, variable-length, undefined, and data sensitive for SAM files; and variable-length for ISAM files. All record format types can be used with USER files.

REC=

```
[REC=  {132   }  ]
       {number}
```

Specifies the logical record length of records in the file. This entry is ignored for RDOS Line-type records (FMT=L). When variable-length, undefined, or data sensitive records are being described, this entry must be coded and also must indicate the length of the longest record in the file. The default value can only be used for fixed-length records (FMT=F).

When inverting a file, REC is defined as minimum mode size and BLK as page size. For a disk with 512-byte sectors, REC=506 bytes and BLK=512 or 1024 bytes.

BLK=

```
[BLK=  {@1     }  ]
       {@number}
       {number }
```

Specifies the blocking factor for blocked records. This value can be specified in terms of:

1. The number of BYTES in a block:

   number

2. The number of RECORDS per block:

   @number

The default value is one RECORD per block:

@1

The following rules apply when coding BLOCK:

1. When FMT=U, V, or D the block length must be entered and the value specified in bytes. Thus, @1 and @number must not be used.

2. BLK is ignored for RDOS Line-type records (FMT=L), and should not be coded.

ORG=

```
[ORG=  {S}  ]
       {R}
       {C}
       {D}
```

Specifies the physical organization of the output file. The default value is sequential. This operand is used only if the output file needs to be created (CREATE = YES).

S	Sequential
R	Random
C	Contiguous
D	Duplicate Keys (ISAM files only)

OWRITE=

```
[OWRITE=  {YES}  ]
          {NO }
```

This operand is used when the CREATE = NO operand is used. When OWRITE = YES is specified, records are to overwrite data in the existing file.

When OWRITE = NO, the output records are appended to the end of the existing file.

WORK1 CONTROL STATEMENT

```
[WORK1,
[FILE=  {SORTWORK.W1}  ],
        {filename   }
[TYPE=  {RDOS }  ],
        {INFOS}
[ORG=   {R}  ],
        {C}
[CREATE=  {YES}  ],
          {NO }
[SAVE=  {NO }  ]]
        {YES}
```

WORK2 CONTROL STATEMENT

```
[WORK2,

[FILE=filename],

[TYPE= {RDOS  } ],
       {INFOS }

[ORG= {R} ],
      {C}

[CREATE= {YES} ],
         {NO }

[SAVE= {NO  } ]]
       {YES }
```

General

These Control statements specify the intermediate storage area(s) to be used during the sorting phase. One or two intermediate work files can be specified for the Sort application (WORK1 and WORK2).

There are reasons why a user may wish to use two work files. First, the user may wish to use contiguous work files but may not have enough space for one large contiguous file. Hence, WORK1 and WORK2 offer the capability of breaking up work space into two contiguous areas.

Secondly, if two disc drives are available, then the user can allocate one work file to the first drive, and the second work file to the other available drive. This significantly reduces arm movement.

WORK1 Control Statement

When this control statement is omitted, the Sort Utility program automatically allocates this work file. The default filename is SORTWORK.W1.

WORK1

WORK1 (Command word)

FILE=

```
[FILE= {SORTWORK.W1} ]
       {filename    }
```

Specifies the name of the work file. Default is SORTWORK.W1. Filename, however, can be specified by the user.

TYPE=

```
[TYPE= {RDOS  } ]
       {INFOS }
```

Indicates the type of work file. RDOS is the default value. RDOS work files must not exceed 2^{16}-1 blocks or 512_{10} bytes. INFOS system work files must not exceed 2^{32}-1 blocks. For INFOS multi-volume files, the files must have been already created.

ORG=

```
[ORG= {R} ]
      {C}
```

Specifies either random or contiguous work file organization. The default value is random.

Sequential file organization is not permitted.

NOTE: If ORG=C and CREATE=NO then sort will be much faster.

CREATE=

```
[CREATE= {YES} ]
         {NO }
```

Indicates whether the user has created the work file. Work file creation is assumed.

When a user filename is used and the file already exists, the sort data overwrites the existing data regardless of CREATE.

NOTE: If CREATE = YES and ORG = C, the user must code the MAX parameter in each IN control statement (the sort utility must know how many records to be sorted in order to allocate sufficient contiguous work space).

SAVE=

```
[SAVE= {NO  } ]
       {YES }
```

Specifies whether or not the work file is to be saved or deleted after the Sort application. Work file deletion is assumed.

NOTE: Saving contiguous work files for reuse saves time at the expense of space.

WORK2 Control Statement

The WORK2 Control statement operands are similar to the WORK1 Control statement, except that a user filename must be entered in FILE=filename. Work file 2 may never be used if the merge phase is executed only once; that is, if few sort strings were generated either because the data was already well sorted or because the amount of input data is small. Approximately 50,000 bytes of random data generates one string in a record sort.

Abnormal Program Termination

It should be noted that if a sort application terminates abnormally the Sort Utility program does not automatically delete work files. Thus, when the utility program builds SORTWORK.W1 by default and the program abnormally terminates and CREATE=YES it is necessary to delete SORTWORK.W1 (or other user-named workfile) via the CLI before any subsequent sort programs using SORTWORK.W1 can be executed.

END CONTROL STATEMENT

END (Command word)

General

The appearance of this control statement indicates the end of the control statement entries for the particular Sort application.

END OF CHAPTER

CHAPTER 4

EXECUTING THE SORT UTILITY PROGRAM

GENERAL

The Sort Utility Program can be called for execution through the following:

Command Line Interpreter (CLI)

BATCH Monitor

User Application program

EXECUTION THROUGH COMMAND LINE INTERPRETER

```
SORT [ {/N} ] [file specifier]
       {/S}

     [listfile name/L]
```

In the CLI mode of operation the control statements can be input in one of two ways:

1. Dynamically through the system console device (normally $TTI):

```
                SORT
sort        ------>  GLOBAL
prompt          >>      .
                >>      .
                >>      .
                >>  END
```

2. From a system supported device other than the system console (i.e., disk, tape, etc.). The file containing the control statements must have been previously created and stored (or mounted) on the device to be used for input. In this instance, the CLI execution command must contain the file name or input device name. As an example:

   ```
   SORT MT0:0
   ```

   signifies that the control statement file is contained on tape and read from tape device MT0:0.

Error Checking

Regardless of the device used to input the Sort control statements, syntax checking is performed on each control statement and any errors encountered are automatically displayed on the system console.

When the system console is used to input the Sort Control statements, any detected errors can be immediately corrected. After the END Control statement has been processed, the Sort is immediately executed. If errors have been detected and corrected an additional END statement must be input from the console.

Global Switches /S And /N

Global switches /S and /N are provided for sort statement syntax checking when the control statements are not entered via the system console device.

Global Switch /S

Global switch /S signifies that control is to be passed back to the system console after the control statements have been input.

This option can be used effectively to:

1. Correct errors in the control statements.
2. Add, delete, or modify control statements.
3. Delay the immediate execution of the Sort.

After the control statement file has been read, the Sort Utility issues the prompt:

```
>>
```

At this time the user can enter corrections, additions, deletions, or modifications to the control statements. At the end of this process, the user must issue:

```
>> END
```

This command causes the sort to be immediately executed.

Global Switch /N

Global switch /N specifies that the control statements are to be input, decoded, and that the Sort is to be immediately executed.

When this switch is used the presence or absence of errors in the control statement file has the following effect:

1. If errors are not detected, the Sort is immediately executed.

2. When errors are detected, abnormal termination of the Sort occurs. The user is returned to CLI.

List File

The [listfile name/L] specifies that a permanent record of the content of the control statements and errors is to be output and normally should be specified. (The listfile is an append file). Listfiles should be kept for each Sort execution because if a failure occurs debugging is made possible. The listfile, since it is an append file, should be cleared periodically. Without a listfile, detecting where and why the Sort failed is very difficult because there is no permanent record.

EXECUTION THROUGH BATCH MONITOR

See the BATCH USER'S MANUAL (093-000087).

EXECUTION BY USER PROGRAM

When the Sort is executed from a user application program, then the system file COM.CM must be written so as to look as if the sort is to be executed from the system console device.

END OF CHAPTER

CHAPTER 5
USER CODE

GENERAL

Provision has been made for the insertion of user procedures in order to process non-standard situations that are not automatically handled by the Sort Utility program. In these instances, the user can take the responsibility for

INPUT - Opening, reading and closing of an input file(s). The IN control statement must include the TYPE = USER operand.

OUTPUT- Opening, writing, and closing of the output file. The OUT control statement must also include the TYPE = USER operand.

KEYS - The user can decide which of two records is to be selected for output. The KEY control statement in this case is optional, but the GLOBAL control statement must include the USERKEY = YES operand.

INCLUDING USER ROUTINES

In order to include a user routine into the Sort, the user must bind together the procedures and Sort Utility program by application of the Relocatable Loader Utility.

Assume, as an example, that the desired procedures are in a source file named USER.SR, and that the routine is in Assembly language.

First, the procedure is assembled to produce a relocatable file called USER.RB. Next, the following commands are issued:

```
RLDR/A/N/L/S $LPT/L USERSORT/S SORTMAIN↑)
 SYS.LB USER↑)
 [SORTINPUT, SORTWRKIN] SORTWRAP↑)
 [SORTWRKOT,SORTOTPUT]↑)
 [SORTKEY,SORTOPEN]↑)
 [SORTPARSE,SORTINIT,SORTDISTR,SORTMERGE]↑)
 [SORTCONFG]↑)
```

The preceding causes the Relocatable Loader to link the Sort Utility program and user procedures under the filename USERSORT.

There are, however, a few items that should be noted:

1. The resulting executable file should not be named SORT, since this conflicts with the name of the standard Sort Utility program.

2. The user procedures must be in the resident portion of the resulting program, and cannot be in overlays.

3. The PAGE-0 is virtually non-existent as far as the user is concerned. Therefore, the user procedures should be written so as to be as self-contained as possible.

RESERVED NAMES

There are a few reserved names that must be included in the user's procedures and declared as entry points (.ENT) to establish proper linkage. These are:

ENTRY LABEL	EXIT LABEL	FUNCTION
IOPEN	IOPNR	Open an input file
IREAD	IREDR	Read the next input record
ICLOS	ICLSR	Close the input file
OOPEN	OOPNR	Open the output file
OWRIT	OWRTR	Write an output record
OCLOS	OCLSR	Close the output file
ERRER	ERRTN	Process an error situation
KEYCK	KYRTN	Choose a record for output

For example, when it is time to open a user file, the Sort issues a

```
JMP   @.+1
      IOPEN
```

command and the user (upon completion of the OPEN), returns via a

```
JMP   @.+1
      IOPNR
```

sequence. Note that IOPEN, IREAD and ICLOS are all required for user-handled input files and that OOPEN, OWRIT and OCLOS are all needed for user output.

User-included routines are costly because they take up valuable room which could otherwise be used for workspace. Hence, they should be included only when necessary.

DATA COMMUNICATIONS

Communications with the user routines are as follows:

IOPEN — AC0 contains a pointer to a 14-word table describing the input file environment. This table contains a coded version of the submitted statement. The format of the table is summarized below. Nothing need be returned from this call.

IREAD — AC0 contains a table address pointer as described for IOPEN. The user must return AC0 as a byte pointer to the start of the record and AC1 must contain the record size (in bytes).

ICLOS — AC0 contains the table pointer. No information need be returned.

OOPEN — AC0 contains the pointer to the output file table. No data returned.

OWRIT — AC0 contains output file table address. AC1 contains a byte pointer to the output record. AC2 has the output record size.

OCLOS — AC0 contains table address. No return.

KEYCK — AC0 and AC1 contain byte pointers to the two records to be compared. AC2 contains the address of the key table, which is a 4-word table consisting of the key environment information. The user must return the byte pointer of the winning record in AC0.

17-WORD TABLE

The 17-word tables for the Input and Output files are outlined below:

0-8	FILENAME
9	OPTIONS (LEFT-TO-RIGHT) RESERVED - 2 BITS FORMAT - 3 BITS 0 = UNDEFINED 1 = VARIABLE 2 = FIXED 3 = DATA SENSITIVE 4 = LINE RESIDUE - 1 BIT: 0 = ON LEFT 1 = ON RIGHT ORG - 2 BITS 0 = SEQUENTIAL 1 = RANDOM 2 = CONTIGUOUS CREATE - 1 BIT 0 = YES 1 = NO SAVE - 1 BIT 0 = NO 1 = YES OVERWRITE - 1 BIT 0 = YES 1 = NO RESERVED - 5 BITS
10	RECORD SIZE IN BYTES
11	BLOCK SIZE IN BYTES
12	SKIP COUNT
13	MAXIMUM RECORDS TO INPUT
14	LINK TO NEXT
15	LEFT-HAND SKIP WORD
16	LEFT-HAND MAX WORD

6-WORD TABLE

The 6-word key table consists of the following information, which is extracted from the 'KEY' command. All values are zero relative:

0	KEY START VALUE
1	KEY END VALUE
2	MAXIMUM KEY SIZE IN BYTES

3	OPTIONS (LEFT-TO-RIGHT)
	SEQ - 2 BITS:
	0 = ASCENDING
	1 = DESCENDING
	BIT VALUES - 6 BITS
	STARTING BIT # - 3 BITS
	ENDING BIT # - 3 BITS
	START FORMAT - 1 BIT:
	0 = START VALUE IS BYTE #
	1 = START VALUE IS CHARACTER LITERAL
	END FORMAT - 2 BITS:
	0 = END VALUE IS BYTE #
	1 = END VALUE IS # BYTES (@)
	2 = END VALUE IS CHARACTER LITERAL
	RESERVED - 1 BIT
	TYPE - 4 BITS:
	0 = SIGNED BINARY
	1 = ASCII (CHAR)
	2 = PACKED DECIMAL
	3 = ZONED DECIMAL
	4 = BIT STRING
	5 = FLOATING POINT
	6 = UNSIGNED PACKED
	7 = UNSIGNED ZONED

4	DUMMY
5	DUMMY

END OF CHAPTER

CHAPTER 6

INTRODUCTION TO THE MERGE UTILITY PROGRAM

GENERAL

The Data General Merge Utility is a general purpose program that enables the user to merge files of logical data records according to a predetermined collating sequence. The program is capable of accepting any file that can be handled by the operating system - with a minimum of time and intervention. There is no practical limit to the number of files that can be input to one merge application. Input and output files may reside on any physical device known to the operating system.

Multiple input files do not have to have the same characteristics. The files can consist of fixed-length, variable-length, undefined, line, or data sensitive record formats - blocked or unblocked. Any number of records may be bypassed at the beginning of each input file ($2^{32}-1$). The maximum number of records to be merged for each input file can also be specified. INFOS system files are fully supported.

Output records can consist of either the input record in its original form, or selected portions of the record. Constants can be inserted into the output when desired. The output records can be fixed-length, variable-length, undefined, line or data sensitive; either blocked or unblocked.

The sequence of the records to be merged is determined by the contents of the sort keys. There is no limit to the number of keys that can be handled. Key types supported include character, bit strings, unsigned binary, zoned decimal, and unsigned packed decimal. Ascending or descending sequence can be specified for each individual key.

Keys are not restricted to starting and ending byte boundaries. Therefore, binary keys can be specified down to the bit level. In addition, the start or end of a field can be specified as a character to be scanned for, instead of an absolute location.

Output of the merge application is a single file. It can also be multi-volume if it has been externally created before using the Merge utility.

The Merge Utility program consists of a number of distinct, sequentially executed phases. During execution, the program chooses the correct merging technique based on control statement information supplied by the user.

Control statements are designed with completely defaultable specifications. Operand values need be entered only if something other than the default values are desired. Only the IN, OUT, KEY, and END control statements are required for any successful Merge execution.

The control statements can be entered dynamically via the system console, or through another system supported device. Regardless of the device used to input the control statements, all control statement syntax errors are immediately output to the listing file or the device if the system console is not used. The user can correct, add, delete, or modify the control statements through the system console before executing the merge if desired.

Numeric operand values in control statements can be input in any one of the following radices: decimal, octal, hexadecimal, or binary.

Merge messages can be directed to any valid system file or device. Interactive or non-interactive modes are available even though a parameter file is not used.

END OF CHAPTER

CHAPTER 7

EXPLANATION OF MERGE CONTROL STATEMENTS

GENERAL

Merge control statements supply the Merge Utility program with information required to identify and initiate the merge application. These control statements specify control field parameters, general information about the input and output files to be used, and the various options that can be selected for each merge operation.

COMMAND WORDS

Each control statement begins with a Command word. The six unique Command words that the program acts upon are:

Command Word	Definition
GLOBAL (Optional)	Describes the overall environment of the merge, and gives some general options which do not pertain to any one Command word in general.
KEY (Required)	Describes each key that is used to order the files during the merge. Multiple keys are entered in order of major to minor precedence.
SEL (Optional)	Describes the data to be extracted from the record for output. This control statement can also specify constant data that is to be placed in the output record in addition to, or in place of input data.
IN (Required)	This control statement defines the input files to be merged. More than one file can be described using a single IN statement, as long as the files being described have the same characteristics. If multiple input files do not have the same characteristics, additional IN control statements are required.
OUT (Required)	This control statement defines the output file specifications. The output file does not necessarily have to bear any relationship to the structure of the input data file(s).
END (Required)	Indicates the end of the control statements to be used for the particular merge application.

Note that most parameters are designed for completely defaultable specifications. Therefore, Command words and their related operands need only be entered if options other than default values are desired. Only the IN, OUT, KEY, and END Commands are required for any particular merge application.

CONTROL STATEMENT FORMAT

A control statement begins with a Command word, followed by a series of operand fields. The operand fields are based on the keyword concept, whereby keywords are given followed by an alphanumeric or numeric value. Each keyword and its corresponding value are linked together by an equal sign.

The Command word and related operand fields are separated from each other by at least one space, comma, or tab. The separators can be used freely as well as intermixed. Separators can also surround the equal signs and right and left parentheses, if desired.

In the following example both control statements are syntactically correct:

```
IN FILE=FILEA REC=80 FMT=F )
IN FILE=FILEA,REC=80,FMT=F )
```

End and Continuation Lines

A control statement line ends with a TERMINATOR (carriage return (), form feed, or null). Control statements can be continued from one line to the next by means of a ↑(Shift N) immediately preceding the TERMINATOR.

For example, the following are valid executable control statements:

```
IN,)
FILE=FILEA,REC=80,FMT=F)

          (or)

IN,↑)
FILE=FILES,↑)
REC=80,↑)
FMT=F)
```

As can be seen from the above examples, there is no practical limit to the physical size of a control statement.

Control Statement Corrections

When more than one control statement of the same type is encountered, the last control statement of the same type entered is used. This permits the user to make immediate corrections on the control statements from the terminal.

If a control statement consists merely of a Command word, this means that any previous entries for that control statement are ignored, and all operands previously entered for that command control statement will be reset to the default values where applicable (i.e., GLOBAL, SEL).

Example:

(first entry)

```
↑N FILE=FILES,REC=80,FMT=F)
```

(second entry)

```
IN FILE=FILEA, REC=60,FMT=F)
```

In the example above, the second control statement overrides the first control statement entry.

Order of Operands

Although the Command word must be the first entry in a control statement, the operands, if present in GLOBAL, IN, and OUT control statements, need not appear in any predefined order within the control statement.

Example:

```
IN FILE=FILEA,REC=80,FMT=F)
```

May also be entered as:

```
IN REC=80,FILE=FILEA,FMT=F)
```

Numeric and Boolean Values Used in Operands

All numbers in the value portion of an operand can be entered in either decimal, octal, hexadecimal, or binary radix. This is accomplished by preceding the number with the proper modifier. The modifiers are:

D - Decimal
O - Octal
X - Hexadecimal
B - Binary

If a modifier is not used, the default is decimal.

As an example, the numbers 50, D50, O62, X32, and B110010 are all equivalent.

Additionally, boolean values can be entered as either TRUE, T, YES, Y or 1 for true, and FALSE, F, NO, N or 0 for false.

CONTROL STATEMENT NOTATION

The following is a definition of the notation used to describe the coding of control statements:

1. Upper case characters and numeric values must be entered exactly as shown.

2. Left and right parentheses and the equal sign must be entered exactly as shown.

3. Brackets [ ], braces { }, elipses ..., lower case characters, and underlines are used as example notations and must not appear in the actual control statement.

[ ]	Square brackets indicate that the enclosed items are optional, and not required items.
{ }	Braces indicate that at least one of the operand values must be chosen.
<u> </u>	Underlined values indicate the operand default values.

number (lowercase)	All integers in the value portion of an operand field can be entered preceded by a modifier. The modifier values are D (decimal), O (octal), X (hexadecimal), and B (binary). The modifier D (decimal) is the default value.
YES or NO	Boolean values YES and NO can be entered as TRUE, T, YES, Y, or 1 for the value true; FALSE, F, NO, N, or 0 for the value false.
Commas	Commas are used in the notation examples as separators. Commas in the actual control statements can be replaced by spaces or tabs.

END OF CHAPTER

CHAPTER 8

MERGE CONTROL STATEMENTS

GLOBAL CONTROL STATEMENT

```
[GLOBAL,
[PAD= {0/number} ]]
```

General

GLOBAL is an optional control statement that can specify the PAD option for the Merge application. This operand is contained in the GLOBAL statement because it cannot logically be placed in any of the other control statements.

GLOBAL

Command Word

PAD=

```
[PAD= {0/number} ]
```

This operand specifies the character to be used for variable-length record and variable-length key padding. The default value is 0 (decimal value of ASCII null).

KEY CONTROL STATEMENT

The KEY Control statement in the Merge Utility is used in the same manner as the KEY Control statement in the Sort Utility. See pages 3-5 through 3-12 of this manual for instructions.

SEL CONTROL STATEMENT

The SEL Control statement in the Merge Utility is used in the same manner as the SEL Control statement in the Sort Utility. See pages 3-13 and 3-14 of this manual for instructions.

IN CONTROL STATEMENT

```
IN,
FILE = filename1 [+filename2 + ...
       + filenamen],
[SEQ = number ],
[TYPE = {RDOS/SAM/ISAM/USER} ],
[FMT = {L/F/V/U/D} ],
[REC = {132/number} ],
[BLK = {@1/@number/number} ],
[SKIP = {0/number} ],
[MAX = {all records/number} ]
```

General

IN is a required control statement that names and describes the input file(s) to be merged. Only the FILE = filename operand is mandatory; all other operands are optional. If more than one file is to be input to Merge, and the files have different characteristics, additional IN control statements must be entered.

IN

IN (Command word)

FILE=

```
FILE=filename1 [+filename2 + ...
      +filenamen]
```

More than one filename can be entered in a single IN control statement as long as all files have the same characteristics. Thus, if FILE1, FILE2, FILE3, and FILE4 have the same characteristics, then the control statement can be entered as:

```
IN FILE=FILE1+FILE2+FILE3+FILE4,
     REC=80, .........
```

The input files are processed in the order in which they are entered.

NOTE: An INFOS system filename must be entered as Indexname plus Databasename.

SEQ=

```
[SEQ= number  ]
```

This entry must be used when multiple IN control statements are used in a merge application.

Example:

```
IN FILE = CUST, REC=80,...SEQ=2
IN FILE = ACCOUNT, REC=80,...SEQ=1
IN FILE = DEDUCT, REC=60,...SEQ=3
```

TYPE=

```
[TYPE= {RDOS}
       {SAM }
       {ISAM} ]
```

Identifies the type of access method to be used to read the input file(s). Default value is RDOS.

FMT=

```
[FMT= {L}
      {F}
      {V}
      {U}
      {D} ]
```

Specifies the record format of the records in the input file(s).

L	(RDOS ASCII line)
F	(fixed-length)
V	(variable-length)
U	(undefined)
D	(data sensitive)

The default value is L (RDOS line).

When fixed-length records are processed, all data records in the file must have the same length. When variable-length records are processed, the length of the data records in the file can vary.

The undefined-length record format is primarily provided for processing records created in some other operating system environment. When a file of this type is processed, the length of each data record in the file is not predetermined.

When data sensitive records are processed, the records in the file can also vary in length. Records are delimited by user-selected characters that indicate the end of each record. The normal record delimiters for this type of record are carriage return, form feed, and null.

For further information concerning the use of variable-length, undefined, and data sensitive records, see the Data General publication entitled "INFOS System Planning Manual", 093-000115.

Only fixed-length and line record formats can be specified for RDOS files; fixed-length, variable-length, undefined, and data sensitive for SAM files; and variable-length for ISAM files. All record format types can be used with USER files.

	FILE TYPES			
Record Format	RDOS	USER	ISAM	SAM
Fixed Length	YES	YES	NO	YES
Variable length	NO	YES	YES	YES
Undefined	NO	YES	NO	YES
Line	YES	YES	NO	NO
Data Sensitive	NO	YES	NO	YES

REC=

```
[REC= {132   }  ]
      {number}
```

Specifies the logical record length of records in the file. This entry is ignored for RDOS Line-type records (FMT=L). When variable-length, undefined, or data sensitive records are being described, this entry must be coded and also must indicate the length of the longest record in the file. The default value can only be used for fixed-length records (FMT-F).

BLK=

```
[BLK= {@1     }  ]
      {@number}
      {number }
```

Specifies the blocking factor for blocked records. This value can be specified in terms of:

1. The number of BYTES in a block:

   number

2. The number of RECORDS per block:

   @number

The default value is one RECORD per block:

@1

The following rules apply when coding BLOCK:

1. When FMT=U, U, or D the block length must be entered and the value specified in bytes. Thus, @1 and @number must not be used.

2. BLK is ignored for RDOS Line-type records (FMT=L), and should not be coded.

SKIP=

```
[SKIP={0     }  ]
      {number}
```

This operand is used to specify the number of records to be skipped at the front of the file (i.e., labels, control records, etc). The default is zero records.

MAX=

```
[MAX= {all records}  ]
      {number     }
```

This entry specifies the maximum number of records to be extracted from the file.

The default value indicates that all the records in the file will be used as input (except for the records specified in the SKIP operand).

OUT CONTROL STATEMENT

```
OUT,
FILE = filename,
[CREATE= {YES}  ],
         {NO }
[TYPE=   {RDOS}   ]
         {SAM }
         {ISAM}
[FMT= {L}  ]
      {F}
      {U}
      {V}
      {D}
[REC= {132   }  ],
      {number}
[BLK= {@1     }  ],
      {@number}
      {number }
[ORG= {S}  ],
      {R}
      {C}
      {D}
[OWRITE= {YES}  ]
         {NO }
```

General

OUT is a required control statement that names and describes the Merge output file.

OUT

OUT (Command word)

FILE=

```
FILE= filename
```

Specifies the name of the file to which the output records are written. The output file may or may not exist before the Merge application is executed (see the explanation of the CREATE operand below).

CREATE=

[CREATE= {YES / NO}]

This operand specifies whether or not the user wishes to create the output file.

If CREATE = YES is used, then the file must not already exist. CREATE = YES is the default value.

TYPE=

[TYPE= {RDOS / SAM / ISAM}]

Identifies the type of output file. Default value is RDOS.

If TYPE = ISAM, the file must already exist (CREATE = NO).

FMT=

[FMT= {L / F / V / U / D}]

Specifies the record format of the records in the output file:

L	(RDOS ASCII line)
F	(fixed-length)
V	(variable-length)
U	(undefined)
D	(data sensitive)

The default value is L (line).

Only fixed=length and line record formats can be specified for RDOS files; fixed-length, variable-length, undefined, and data sensitive for SAM files; and variable-length for ISAM files.

REC=

[REC= {132 / number}]

Specifies the logical record length of records in the file. This entry is ignored for RDOS Line-type records (FMT=L). When variable length, undefined, or data sensitive records are being described, this entry must be coded and also must indicate the length of the longest record in the file. The default value can only be used for fixed-length records (FMT=F).

BLK=

[BLK= {@1 / @number / number}]

Specifies the blocking factor for blocked records. This value can be specified in terms of:

1. The number of BYTES in a block:

   number

2. The number of RECORDS per block:

   @number

The default value is one RECORD per block:

@1

The following rules apply when coding BLOCK:

1. When FMT = U, V, or D the block length must be entered and the value specified in bytes. Thus, @1 and @number must not be used.
2. BLK is ignored for RDOS Line-Type records (FMT=L), and should not be coded.

ORG=

[ORG= {S / R / C / D}]

Specifies the physical organization of the output file. The default value is sequential. This operand is used only if the output file needs to be created (CREATE = YES).

S	Sequential
R	Random
C	Contiguous
D	Duplicate Keys (ISAM files only)

OWRITE=

[OWRITE= {YES / NO}]

This operand is used when the CREATE = NO operand is used. When OWRITE = YES is specified, records are to overwrite data in the existing file.

When OWRITE = NO, the output records are appended to the end of the existing file.

END CONTROL STATEMENT

END (Command word)

General

The appearance of this control statement indicates the end of the control statement entries for the particular Merge application.

END OF CHAPTER

CHAPTER 9

EXECUTING THE MERGE UTILITY PROGRAM

GENERAL

The Merge Utility Program can be called for execution through the following:

Command Line Interpreter (CLI)

BATCH Monitor

User application program

EXECUTION THROUGH COMMAND LINE INTERPRETER

```
MERGE [ {/N} ]  [file specifier]
      [ {/S} ]

      [listfile name/L]
```

In the CLI mode of operation the control statements can be input in one of two ways:

1. Dynamically through the system console device (normally $TTI):

```
                MERGE
Merge  ------>  >>  GLOBAL
prompt          >>    .
                >>    .
                >>    .
                >>  END
```

2. From a system supported device other than the system console (i.e., disk, tape, etc.). The file containing the control statements must have been previously created and stored (or mounted) on the device to be used for input. In this instance, the CLI execution command must contain the file name or input device name. As an example:

```
MERGE MT0:0
```

signifies that the control statement file is contained on tape and read from tape device MT0:0.

Error Checking

Regardless of the device used to input the Merge control statements, syntax checking is performed on each control statement, and any errors encountered are automatically displayed on the system console.

When the system console is used to input the Merge Control statements, any detected errors can be immediately corrected. After the END Control statement has been processed, the Merge is immediately executed.

Global Switches /S And /N

Global switches /S and /N are provided for statement syntax-checking when the control statements are not entered via the system console device.

Global Switch /S

Global switch /S signifies that control will be passed back to the system console after the control statements have been input.

This option can be used effectively to:

1. Correct errors in the control statements.
2. Add, delete, or modify control statements.
3. Delay the immediate execution of the Merge.

After the control statement file has been read, the Merge Utility issues the prompt:

```
>>END
```

This command causes the merge to be immediately executed.

Global Switch /N

The Global switch /N specifies that the control statements are to be input, decoded, and that the merge is to be immediately executed.

When this switch is used the presence or absence of errors in the control statement file has the following effect:

1. If errors are not detected, the Merge is immediately executed.

2. When errors are detected, abnormal termination of the Merge occurs.

List File

The [listfile name /L] specifies that a permanent record of the content of the control statements and errors is to be be output.

EXECUTION THROUGH BATCH MONITOR

See the BATCH USER'S MANUAL (093-000087).

EXECUTION BY USER PROGRAM

When the merge is executed from a user application program, then the system file COM.CM must be written so as to look as if the merge is to be executed from the system console device.

END OF CHAPTER

APPENDIX A

OCTAL AND HEXADECIMAL CONVERSIONS

To convert a number from octal or hexadecimal to decimal, locate in each column of the appropriate table the decimal equivalent for the octal or hex digit in that position. Add the decimal equivalents to obtain the decimal number.

To convert a decimal number to octal or hexadecimal:

1. Locate the largest decimal value in the appropriate table that will fit into the decimal number to be converted;

2. note its octal or hex equivalent and column position;

3. find the decimal remainder.

Repeat the process on each remainder. When the remainder is 0, all digits will have been generated.

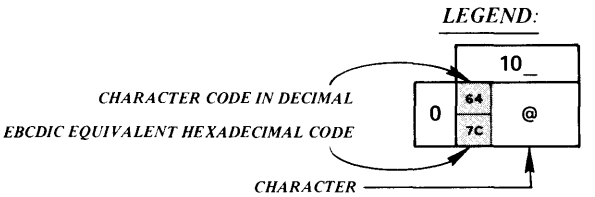
	8^5	8^4	8^3	8^2	8^1	8^0
0	0	0	0	0	0	0
1	32,768	4,096	512	64	8	1
2	65,536	8,192	1,024	128	16	2
3	98,304	12,228	1,536	192	24	3
4	131,072	16,384	2,048	256	32	4
5	163,840	20,480	2,560	320	40	5
6	196,608	24,576	3,072	384	48	6
7	229,376	28,672	3,584	448	56	7

	16^5	16^4	16^3	16^2	16^1	16^0
0	0	0	0	0	0	0
1	1,048,576	65,536	4,096	256	16	1
2	2,097,152	131,072	8,192	512	32	2
3	3,145,728	196,608	12,288	768	48	3
4	4,194,304	262,144	16,384	1,024	64	4
5	5,242,880	327,680	20,480	1,280	80	5
6	6,291,456	393,216	24,576	1,536	96	6
7	7,340,032	458,752	28,672	1,792	112	7
8	8,388,608	524,288	32,768	2,048	128	8
9	9,437,184	589,824	36,864	2,304	144	9
A	10,485,760	655,360	40,960	2,560	160	10
B	11,534,336	720,896	45,056	2,816	176	11
C	12,582,912	768,432	49,152	3,072	192	12
D	13,631,488	851,968	53,248	3,328	208	13
E	14,680,064	917,504	57,344	3,584	224	14
F	15,728,640	983,040	61,440	3,840	240	15

END OF APPENDIX

APPENDIX B

ASCII CHARACTER CODES



↑ *means* CONTROL

OCTAL	00_	01_	02_	03_	04_	05_	06_	07_
0	0 00 NUL	8 16 BS *(BACK-SPACE)*	16 10 DLE ↑P	24 18 CAN ↑X	32 40 SPACE	40 4D (	48 F0 Ø	56 F8 8
1	1 01 SOH ↑A	9 05 HT *(TAB)*	17 11 DC1 ↑Q	25 19 EM ↑Y	33 5A !	41 5D)	49 F1 1	57 F9 9
2	2 02 STX ↑B	10 15 NL *(NEW LINE)*	18 12 DC2 ↑R	26 3F SUB ↑Z	34 7F " *(QUOTE)*	42 5C *	50 F2 2	58 7A :
3	3 03 ETX ↑C	11 0B VT *(VERT. TAB)*	19 13 DC3 ↑S	27 27 ESC *(ESCAPE)*	35 7B #	43 4E +	51 F3 3	59 5E ;
4	4 37 EOT ↑D	12 06 FF *(FORM FEED)*	20 3C DC4 ↑T	28 1C FS ↑\	36 5B $	44 6B , *(COMMA)*	52 F4 4	60 4C <
5	5 2D ENQ ↑E	13 0D RT *(RETURN)*	21 3D NAK ↑U	29 1D CS ↑]	37 6C %	45 60 –	53 F5 5	61 7E =
6	6 2E ACK ↑F	14 0E SO ↑N	22 32 SYN ↑V	30 1E RS ↑↑	38 50 &	46 4B . *(PERIOD)*	54 F6 6	62 6E >
7	7 2F BEL ↑G	15 0F SI ↑O	23 26 ETB ↑W	31 1F US ↑←	39 7D ' *(APOS)*	47 61 /	55 F7 7	63 6F ?

OCTAL	10_	11_	12_	13_	14_	15_	16_	17_
0	64 7C @	72 C8 H	80 D7 P	88 E7 X	96 79 ` *(GRAVE)*	104 88 h	112 97 p	120 A7 x
1	65 C1 A	73 C9 I	81 D8 Q	89 E8 Y	97 81 a	105 89 i	113 98 q	121 A8 y
2	66 C2 B	74 D1 J	82 D9 R	90 E9 Z	98 82 b	106 91 j	114 99 r	122 A9 z
3	67 C3 C	75 D2 K	83 E2 S	91 8D [	99 83 c	107 92 k	115 A2 s	123 C0 {
4	68 C4 D	76 D3 L	84 E3 T	92 E0 \	100 84 d	108 93 l	116 A3 t	124 4F |
5	69 65 E	77 D4 M	85 E4 U	93 9D]	101 85 e	109 94 m	117 A4 u	125 D0 }
6	70 C6 F	78 D5 N	86 E5 V	94 5F ↑ or ^	102 86 f	110 95 n	118 A5 v	126 A1 ~ *(TILDE)*
7	71 C7 G	79 D6 O	87 E6 W	95 6D ← or _	103 87 g	111 96 o	119 A6 w	127 07 DEL *(RUBOUT)*

CHARACTER CODE IN OCTAL AT TOP AND LEFT OF CHARTS.

SD-00217

END OF APPENDIX

APPENDIX C

SORT AND MERGE SYSTEM ERROR CODES

500 - MEMORY EXHAUSTED AT DISTRIBUTION TIME

512 - NOT INFOS SAM FOR TAG SORT

513 - USING LINE TYPE WITH INFOS FILE ON INPUT

520 - RDOS WORKFILE ADDRESS OVERFLOW

530 - TOO MANY RECORDS FOR SORTWORK

531 - TOO MANY RECORDS FOR CONTIGUOUS OUTPUT FILE CREATION

532 - TOO MANY RECORDS FOR CONTIGUOUS WORK FILE CREATION

561 - UNKNOWN ERROR IN INTERPRETER PHASE

575 - MEMORY EXHAUSTED IN INTERPRETER PHASE

END OF APPENDIX

APPENDIX D

NUMBER REPRESENTATIONS

DECIMAL NUMBERS

Decimal numbers may be represented internally in two ways, character decimal and packed decimal. In character decimal, the number is made up of a string of ASCII characters and the sign, if present, may appear in one of four places. The sign of the number may be indicated by a leading or trailing byte which contains the ASCII code for plus ($2B_{16}$) or minus ($2D_{16}$). Alternatively, either the high-order digit or the low-order digit of the number may indicate the sign in addition to carrying a digit of the number. The table below gives the correspondence between certain ASCII characters and the sign and digit values that they carry.

SIGN VALUE	DIGIT VALUE	ASCII CHARACTER	HEX CODE
+	0		7B
+	1	A	41
+	2	B	42
+	3	C	43
+	4	D	44
+	5	E	45
+	6	F	46
+	7	G	47
+	8	H	48
+	9	I	49
-	0		7D
-	1	J	4A
-	2	K	4B
-	3	L	4C
-	4	M	4D
-	5	N	4E
-	6	O	4F
-	7	P	50
-	8	Q	51
-	9	R	52

The digits that are not carrying the sign must be valid ASCII characters for the digits 0-9 (30_{16} - 39_{16}).

Examples:

In the following examples, the hex value of a byte is shown inside the box: The corresponding ASCII character is shown beneath the box.

+2,048 (leading sign)	2B	32	30	34	38
	+	2	0	4	8
-1,756 (trailing sign)	31	37	35	36	2D
	1	7	5	6	-
+1,850 (high-order sign)	41	38	35	60	
	A	8	5	0	
-3,970 (low-order sign)	33	39	37	7D	
	3	9	7		

For packed decimal, each digit of the decimal number occupies one hex digit. The sign is specified by a trailing hex digit. The number must start and end on a byte boundary. In other words, the number cannot start or end halfway through a byte. This means that a packed decimal number will always consist of an odd number of digits followed by the sign. The sign must be either C_{16} for plus or D_{16} for minus. The only valid codes for digits are $0\text{-}9_{16}$.

Examples:

In the following examples, the hex value of a digit is shown within the box: the corresponding decimal digit is shown beneath the box.

+2,048	0	2	0	4	8	C
	0	2	0	4	8	+
+32,456	3	2	4	5	6	C
	3	2	4	5	6	+
-1,756	0	1	7	5	6	D
	0	1	7	5	6	-
-25,989	2	5	9	8	9	D
	2	5	9	8	9	-

END OF APPENDIX

DataGeneral

SOFTWARE DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title	Date

Company Name

Address (No. & Street)	City	State	Zip Code

FOLD DOWN FIRST FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

BUSINESS REPLY MAIL

No Postage Necessary If Mailed In The United States

Postage will be paid by:

Data General Corporation

Southboro, Massachusetts 01772

ATTENTION: Software Documentation

FOLD UP SECOND FOLD UP

STAPLE