IBM Systems Reference Library

OS Sort/Merge Program

Program Number 360S-SM-023

OS-Release 21

This publication describes the use of the OS Sort/Merge Program. It discusses:

- Program capabilities.
- Sorting and merging techniques.
- Sort/merge program control statements.
- Intermediate storage requirements.
- Job control language requirements.
- Program initiation.
- Program modification.
- Efficient program use.
- Standard operating system collating sequence.
- Sort/merge program messages.

The program has generalized sorting and merging capabilities that can be tailored to the needs of a particular installation and application.

Preface

This publication is a guide for users of the System/360 Operating System Sort/Merge program. It contains a general description of the program and specific information about control statement formats, program operation, the inclusion of user-written routines, efficient use of the program, and program generated messages. Merging techniques used by the program are briefly described. General information about basic sorting and merging methods is contained in the IBM publication Sorting Techniques, Order No. GC20-1639.

ORGANIZATION AND USE OF THIS PUBLICATION

If you want to set up a simple sort or merge quickly, fold out the chart in Appendix A at the back of the book and refer to Section 2 for details as you follow the chart. Eventually, however, you should plan to read the entire publication, which is organized as follows:

Section 1: Sort/Merge Program -- This section describes sorting and merging specifications, control fields, sorting and merging techniques used by the program, and error correction facilities.

Section 2: How to Use Sort/Merge -- This section is divided into four main topics: "Defining the Sort or Merge" which describes the format and use of sort/merge control statements and contains a number of complete sorting and merging examples; "Determining Intermediate Storage Requirements" which describes how to calculate the amount of intermediate storage for a given application; "Required Job Control Language Statements" which describes the JOB, EXEC, and DD statements necessary for sort/merge execution and contains a number of complete JCL and sort/merge statement examples; and "Invoking the Sort/Merge Program" which describes initiating sort/merge via the system input stream and via a macro instruction in another program.

Section 3: Program Modification -- This section describes sort/merge program exits and the requirements for user-written routines that use them. Users who do not

include their own routines to modify records or handle errors during sort/merge program execution can skip this section.

Section 4: Efficient Program Use -- This section describes the factors that contribute to efficient use of the sort/merge program.

Appendixes -- These sections contain a flow chart summary of how to use the sort/merge program, a summary of considerations for MVT users, the standard operating system collating system, and sort/merge messages.

PREREQUISITE PUBLICATIONS

OS Introduction, Order No. GC28-6534

SUGGESTED READING

OS Sort/Merge Timing Estimates, Order No. GC28-6662 for information about sorting speeds with a variety of work devices, input data set sizes, main storage sizes, blocking factors, etc.

OS Supervisor Services and Macro Instructions, Order No. GC28-6646 and OS Data Management Macro Instructions, Order No. GC26-3794 for descriptions of linkage conventions, program and task management, data organization, and access features, and the use of macro instructions.

OS Job Control Language Reference, Order No. GC 28-6704 for a reference to job control language statements.

OS Advanced Checkpoint/Restart Planning Guide, Order No. GC28-6708.

The following additional publications are referred to in the text:

OS Loader & Linkage Editor, Order No. GC28-6538
OS Storage Estimates, Order No. GC28-6551
OS Data Management for Systems Programmers, Order No. GC28-6550
OS Sysgen, Order No. GC28-6554.

3

Contents

This publication explains how to use the System/360 Operating System Sort/Merge Program to fulfill the sorting and merging requirements of System/360 installations that use magnetic tape and direct access input and output devices.

The sort/merge program can arrange data sets into a predesignated order. The program places the records of a data set in sequence according to the contents of a control word which is contained in each record. The program is generalized to perform a variety of sorts and merges. Because of this ability, the sort/merge program can simplify many data processing applications that require the sequential updating of previously created data sets.

Input to and output from the program can be any data set that consists of fixed-length or variable-length, blocked or unblocked records (except U format) and can be accessed by the queued sequential access method (QSAM). Any I/O device that operates with QSAM can be used for input and output.

The program uses sorting and merging techniques that take advantage of machine configurations and data set sizes. These techniques are designed to provide efficient operation for a great variety of sorts and merges. The technique used by sort/merge depends upon information supplied to the program through control statements which define the application to be performed. These statements can be supplied to the program in the operating system input stream or as parameters passed by another program.

User-written routines can operate in conjunction with the sort/merge program to perform many functions during sort/merge execution. The program gives control to user-written routines at various exits in the program. When they receive control, the routines can insert, summarize, delete, and alter the records being sorted or merged.

Relationship to the Operating System

The sort/merge program is part of the System/360 Operating System and operates under the supervisory control of the operating system control program. Sort/merge execution must be initiated according to operating system conventions, and any data sets used by the program must be defined according to operating system standards. At the user's option, the checkpoint and label checking (standard and nonstandard) facilities of the operating system can be used during a sort/merge program execution. Information about operating system label checking facilities can be found in the publication OS Data Management for System Programmers.

The sort/merge program also makes extensive use of the operating system data management facilities. All data sets necessary for program operation must be defined in data definition statements; these statements must be placed in the operating system input stream with the job step that initiates sort/merge execution. DD statements are described in the publication OS JCL Reference.

The sort/merge program can be tailored to the needs of a particular installation when the operating system for that installation is generated.

Minimum Machine Requirements

The sort/merge program requires:

- For main storage, a System/360 model that is large enough to use the operating system and provide at least 15,500 bytes of main storage for sort/merge execution. (Sort/merge uses 12,000 bytes; system functions use 3,500 bytes.)

- At least one selector channel or one multiplexor channel.

- For intermediate storage, at least one IBM 2311 Disk Storage Drive, or one IBM 2301 Drum Storage Drive, or one drive of an IBM 2314 Direct Access Storage Facility or three magnetic tape units.

MAIN STORAGE REQUIREMENTS

Sort/merge performance usually improves as the amount of main storage available to the program increases. Approximately 44K bytes of main storage are required for efficient operation. Refer to "Section 4: Efficient Program Use" for more information about main storage requirements.

Determining Region Size

Use the following formula to estimate the region size required when the sort/merge program is run under MVT:

Region size = 1.2(sort size) + 8K

Sort size is the amount of main storage assigned to the sort/merge program at system generation time. If the user overrides the SYSGEN value at execution time, then the overriding value is used for sort size. The constant 1.2 provides for space lost through fragmentation, and the additional 8K is used by the system.

If the formula yields a region size less than the minimum allowed, use the minimum. If calculated region size is not a multiple of 2K, round up to the nearest 2K multiple .

INTERMEDIATE STORAGE REQUIREMENTS

The amount of intermediate storage needed to perform sorting applications depends upon the size of the input data set. This storage may be allocated on either magnetic tape or direct access devices. The program needs at least threee magnetic tape units or one direct access device for intermediate storage.

The amount of main storage available to the sort/merge program affects the size of records that the program can handle. Figure 1 shows the maximum record size that the program will accept for a given amount of main storage when fixed- or variable-length unspanned records are used. Figure 2 gives sizes for variable-length spanned records. (Spanned records, also referred to as VRE records, are records that can have fractional blocking factors such as one third, two and one half, etc. Thus a record may "span" blocks and/or direct access tracks.)

Figures 1 and 2 assume that the minimum number of intermediate storage data sets are assigned, and no control fields are to be extracted (placed in a work area and modified by user-written routines). Minimum record size is 18 bytes. Conditions such as control field extraction, or large numbers of intermediate storage data sets require additional main storage. Since a work area is used for VRE records, the available storage space for buffers and sorting is decreased and therefore, the maximum record lengths for VRE records are somewhat smaller than for unspanned records.

Main Storage Available to the Sort	Maximum Record Size for Input and Output Records				
	Intermediate Storage Device				
	Tape (Bytes)	IBM 2311 Disk (Bytes)	IBM 2301 Drum (Bytes)	IBM 2314 Facility (Bytes)	
				3 workareas	6 workareas
18K¹ (18,432)	1,100	1,300	1,300	1,300	---
44K (45,056)	5,500	3,600	6,600	6,600	3,500
100K (102,400)	14,900	3,600	18,000	7,272	7,272
200K (and up) (204,800)	32,000	3,600	20,458	7,272	7,272
¹The value of K is 1,024 bytes.					

Figure 1. Estimated Maximum Record Sizes for Input and Output with Fixed-Length
or Variable-Length Records

Main Storage Available to the Sort	Maximum Record Size for Input and Output Records				
	Intermediate Storage Device				
	Tape (Bytes)	IBM 2311 Disk (Bytes)	IBM 2301 Drum (Bytes)	IBM 2314 Facility (Bytes)	
				3 Workareas	6 Workareas
18K¹ (18,423)	800	900	1,100	900	---
44K (45,056)	4,600	3,600	5,100	5,100	3,100
100K (102,400)	12,800	3,600	12,900	7,272	7,272
200K (and up) (204,800)	27,400	3,600	20,458	7,272	7,272
¹The value of K is 1,024 bytes.					

Figure 2. Estimated Maximum Record Sizes for Input and Output with
Variable-Length Spanned Records (VRE)

OPERATING SYSTEM OPTIONS

The sort/merge program is part of the System/360 Operation System and operates
under the control of the Operating System control program. The following
Operating System services may be used with the sort/merge program if they are
included in the system when it is generated:

1. Checkpoint/Restart
2. Input/Output Recovery Management Support (see OS Introduction)
3. Multiple Console Support and Message Routine (see OS Data Management for
    System Programmers)
4. System support for up to 12 channel addresses.

When the system is generated, certain sort/merge facilities may be omitted if they
are not needed by a particular installation. This saves library space. You
should be aware of what particular Operating System features and options are
available at your installation.

This section discusses control fields, sort and merge requirements, the sorting technique used by sort/merge, the sort/merge sequence distribution techniques, and error correction facilities.

Control Fields

Each record in a data set is sorted or merged on the basis of control information contained in the record's control word. A control word, which can be up to 256 bytes long, has from 1 to 64 control fields. Control fields can overlap; the end of one control field can share data with the beginning of another control field. Figure 3 shows a control word with five control fields.
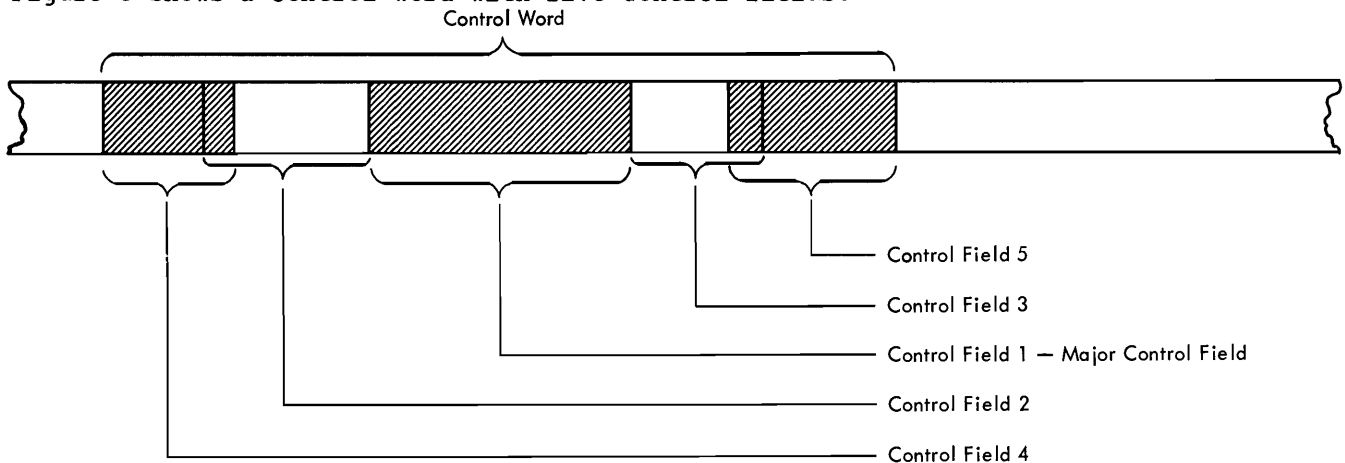
Figure 3. Control Word With Five Fields

Each control word, along with the record in which it appears, is sorted into either ascending or descending order, using standard IBM System/360 collating sequences.[1]

Nonstandard collating can be achieved without physically changing the control fields. A user-written routine can modify one or more of the control fields each time the sort/merge program collates a record. The modified control fields are used for collating purposes only; they do not replace the fields in the records. User-written routines can be entered at sort/merge program exits. (These exits and the requirements for user-written routines that use them are discussed in "Section 3: Program Modification".)

The maximum control field lengths for the various control field data formats accepted by the sort/merge program are:

• Character, fixed-point, or normalized floating point data -- 1 through 256 bytes.
• Packed or zoned decimal data -- 1 through 32 bytes.
• Binary data -- 1 bit through 256 bytes.

Control fields must be contained within the first 4,092 bytes of a record.

----

[1]The collating sequence for character data and binary data is absolute; that is, character and binary fields are not interpreted as having signs. (Refer to Appendix C: Collating Sequence.) For packed decimal, zoned decimal, fixed point, and normalized floating-point data, collating is algebraic; that is, each quantity is interpreted as having an algebraic sign.

Sort Requirements

Control fields for a sorting application are defined in a SORT control statement such as

SORT FIELDS= (10,30,A) ,FORMAT=CH

(described in "Defining the Sort or Merge" in Section 2). Input, output, and intermediate storage data sets are defined on standard job control language DD statements such as

```
//SORTOUT   DD   DSNAME=OUTPUT,UNIT=2400,DISP= (NEW,CATLG) ,      X
//               DCB= (RECFM=FB,LRECL=90,BLKSIZE=900)
```

(described in "Job Control Language for Sort/Merge" in Section 2).

INPUT: Sort input can be a blocked or unblocked sequential data set containing fixed- or variable-length records on any I/O device that can be used with QSAM.

OUTPUT: Output from the sort can be a blocked or unblocked sequential data set containing fixed- or variable-length records. The output device can be any device that can be used with QSAM. It need not be related in any way to the input device.

INTERMEDIATE STORAGE: All intermediate storage for a particular sort/merge application must be on the same type of device. Up to 32 tape units, 17 modules of a 2314 storage facility, six 2311 disk storage drives, or six 2301 drum storage devices can be used for intermediate storage. Data set organization must be sequential. The amount of intermediate storage required is based primarily on the size of the input data set. The amount of main storage available to sort/merge is also a factor in determining intermediate storage requirements. Intermediate storage is discussed in greater detail and formulas for the amount of storage needed are given in "Determining Intermediate Storage Requirements" in Section 2.

USER MODIFICATIONS: User-written routines can summarize, insert, delete, shorten, lengthen, or otherwise alter records while they are being sorted. A detailed discussion of exits in the sort/merge program that permit control to be transferred to user-written routines is given in "Section 3: Program Modification."

INVOKING THE SORT: Execution of the sort is initiated by control statements in the operating system input stream, or by another program through the use of an ATTACH, LINK, or XCTL macro instruction.

Merge Requirements

Control fields for a merging application are defined in a MERGE control statement such as

MERGE FIELDS= (10,30,A) ,FORMAT=CH

(described in "Defining the Sort or Merge" in Section 2). Input and output data sets are defined on standard job control language statements such as

```
//SORTIN01 DD   DSNAME=MERGE1,VOLUME=SER=000111,DISP=OLD,       X
//               LABEL= (,NL) ,UNIT=2400,DCB= (RECFM=FB,        X
//               LRECL=80,BLKSIZE=240)
```

(described in "Job Control Language for Sort/Merge" in Section 2).

INPUT: Input to the merge can be up to 16 blocked or unblocked sequential data sets containing fixed- or variable-length records. For a given application, all records must be of the same format (only blocking factors may differ). The records in the input data sets must be in proper sequence. The input devices must be acceptable for use with QSAM.

OUTPUT: Output from the merge can be a blocked or unblocked sequential data set
containing fixed- or variable-length records. The output device must be
acceptable for use with QSAM. It need not be related in any way to the input
device type.

INTERMEDIATE STORAGE: Not needed for a merge-only operation.

USER MODIFICATION: The merge provides exits for user-written routines to
summarize, insert, delete, lengthen, shorten, or otherwise alter output records.
A detailed discussion of these exits and the requirements for routines that use
them is given in "Section 3: Program Modification."

INVOKING THE MERGE: Execution of the merge can only be initiated by control
statements in the operating system input stream.

Sorting Technique

The sort/merge program uses the replacement selection technique to sort records.
Figure 4 shows in general how this technique works.

    The input data set is almost always too large to be brought into main storage
and sorted all at once. Instead, it is broken up into sections. Each section is
placed in sequence and stored on an intermediate storage device. The sorted
sections of the input data set are called sequences.

Input Data Set SORTIN

| 74 | 91 | 3 | 8 | 21 | 7 | 18 | 17 | 89 | 1 | 19 | 72 | 12 |

Input Buffer (usually one of two)

| 74 | 91 | 3 | 8 | 21 | 7 | 18 | 17 | 89 |

Record from buffer   RSA (already primed)

74   compare
6
70
25
11
5 ◄── First record selected for output

to see if record 74 fits
in current sequence;
i.e., collates higher
than record selected
for output

Output Buffer

| | | | 5 |

Next record from buffer

91 ◄── compare ── ► 6 ◄── Second record selected for output
70
25
11
74

Record 91 fits in
this sequence

Output Buffer

| | | 6 | 5 |

Next record from buffer

3 ◄──
91
70
25
compare
11 ◄── Third record selected for output
74

This record does not fit into
the current sequence. It
replaces record 11 and is
flagged. It will not be
examined again until the
next output sequence is
begun.

Output Buffer

| | 11 | 6 | 5 |

Next record from buffer
91
70
8 ◄── compare ──► 25 ◄── Fourth record selected for output
3*
Record 8 does not fit   74
in the current sequence

Output Buffer

| | 25 | 11 | 6 | 5 |

Next record from buffer
91
70 ◄── Fifth record selected for output
21 ◄── compare ──►
8*
3*
Record 21 does not fit   74

Output Buffer

| 70 | 25 | 11 | 6 | 5 |

Next record from buffer
91
21*
7 ◄──
8*
compare
3*
74 ◄── Sixth record selected for output
Record 7 does not fit

Output Buffer

| 74 | 70 | 25 | 11 | 6 | 5 |

Next record from buffer
18 ◄── compare ──► 91 ◄── Seventh record selected for output
21*
8*
3*
7*
Record 18 does not fit

Output Buffer

| 91 | 74 | 70 | 25 | 11 | 6 | 5 |

End current sequence –   18
begin new sequence   21
8
3
7

Figure 4. Replacement Selection Sorting Technique

Sequence Distribution Techniques

The sort/merge program selects one of five sequence distribution techniques based on information it has about a specific sorting application. The object of all five techniques is to enable the intermediate merge phase of the program to combine the many small sequences of records produced by the sort phase into a few longer sequences. The number of sequences must be reduced to the point where the final merge phase of the sort/merge program can combine them into a single sequence in one pass.

TAPE TECHNIQUES

If the intermediate storage medium is tape, the program chooses the balanced tape technique, the polyphase tape technique, or the oscillating tape technique.

DIRECT ACCESS TECHNIQUES

If the intermediate storage medium is direct access, the program chooses either the balanced direct access technique or the crisscross direct access technique.

Figure 5 lists the basic requirements for the five sequence distribution techniques and their major advantages and disadvantages.

FORCING A TECHNIQUE

If you find that for a particular sort/merge application, the sort/merge program does not choose the most efficient technique, you can request sort to use another technique. The program will comply if you provide enough main storage and work areas to meet the technique's requirements (see Figure 5). If the requirements cannot be met, sort will use another technique rather than terminate the program.

Caution: Be extremely cautious about forcing a technique. The sort/merge program attempts to choose the most efficient technique for a given application. If it is forced to use another technique, performance is usually not as efficient.

Refer to the discussion of the EXEC statement PARM field in "Job Control Language Statements for Sort/Merge" in Section 2 for information on how to force a sequence distribution technique.

Technique	Minimum Main Storage For Sort/Merge	Maximum Input	Minimum Intermediate Storage Areas Required	Maximum Intermediate Storage Areas Permitted	Comments
Balanced Tape BALN	12,000 bytes	15 reels	2(x+1), where x is the number of input volumes	32 tape units	Always used if more than three intermediate storage tapes are available and input data set size is not specified or estimated.
Polyphase Tape POLY	12,000 bytes	1 reel	3 reels	17 tape units	Always used if only three intermediate storage tapes are available.
Oscillating Tape OSCL	21,000 bytes	15 reels	x+2 or 4, whichever is greater, where x is the number of input volumes	17 tape units	Input data set size must be given or closely estimated. The tape drive containing SORTIN, the input data set, cannot be assigned as an intermediate storage unit.
Balanced Direct Access BALN	13,000 bytes	No fixed maximum, depends on available main storage and available intermediate storage	3 areas	6 areas	The only technique available for the 2301 and 2311. Always used on 2314 when less than six work areas are available. Used on 2314 when six areas are available unless CRCX is forced.
Crisscross Direct Access CRCX	24,000 bytes		6 areas	17 areas	Always used on 2314 when more than six work areas are available. Used on 2314 when six areas are available but must be forced. Not used on 2301 or 2311.

Figure 5. Sequence Distribution Technique Requirements

Error Correction Facilities

The sort/merge program provides exits where control can be transferred to user-written error routines. (Refer to "Section 3: Program Modification".) These routines may be able to correct:

- I/O errors that cannot be corrected by the operating system.
- Errors that arise because the input data set is larger than the intermediate storage capacity estimated by the program for a given application.

I/O ERRORS

The sort/merge program passes control to a user-written I/O error routine only when the operating system cannot correct the error condition. In the case of a permanent read error the user-written routine can accept the block as is, attempt to correct the error, skip the block, or request termination. For an

uncorrectable write error, the user-written routine can perform any necessary abnormal end-of-task operations before the program is terminated.

If no user-written routines are supplied, the sort/merge program issues the message IER061A-I/O ERR xxx, where xxx represents the number of the unit on which the error occurred. Then the program terminates.

EXCEEDING INTERMEDIATE STORAGE CAPACITY

The sort/merge program estimates a maximum intermediate storage capacity (Nmax) from the information supplied to it at the beginning of the sorting operation.

Note: Nmax for magnetic tape is calculated for 2400-foot tapes. For shorter tapes the figure should be reduced proportionately.

You can supply an actual or an estimated input data set size to the program. (This is done via the SIZE parameter on a SORT control statement, described in "Defining the Sort or Merge" in Section 2.) If you supply an actual data set size, and the size is larger than Nmax, the program terminates before starting to sort. If you supply an estimated data set size, or if you do not give a data set size, and the number of records processed while sorting reaches Nmax, the program gives control to a user-written Nmax routine, if one is supplied. The Nmax routine can take one of the following actions:

- Indicate to sort/merge that it should continue sorting the entire input data set with available intermediate storage. (If the estimated input data set size was high, there may be enough intermediate storage left to complete the application.)
- Direct sort/merge to continue sorting with only part of the input data set. (The remainder of the data set could be sorted later and the two results merged to complete the application.)
- Terminate the program without any further processing.

If an Nmax routine is not supplied, sort/merge continues to process records beyond Nmax. If the intermediate storage capacity is sufficient to contain all the records in the input data set, the sort completes normally; when intermediate storage is not sufficient, the program terminates.

The sort generates a separate message for each of the three possible error conditions. These messages are:

IER041A-N GT NMAX: Generated before sorting begins when the exact data set size supplied on a SORT control statement is greater than Nmax.

IER046A-SORT CAPACITY EXCEEDED: Generated when the sort has used all available intermediate storage while processing.

IER048I-NMAX EXCEEDED: Generated when the sort has exceeded Nmax and has transferred control to a user-written Nmax routine for further action.

(A full description of all program messages is contained in Appendix D.)

Section 2: How to Use the Sort/Merge Program

There are three basic things you must do to use the sort/merge program:

1. Define your sorting or merging job with sort/merge control statements. (See "Defining the Sort or Merge" in this section.)

2. If your job is a sort, determine the amount of intermediate storage your data will require while it is being sorted and merged. (See "Determining Intermediate Storage Requirements" in this section.)

3. Prepare job control language statements for the job and combine them in proper order with the sort/merge control statements. (See "Job Control Language for Sort/Merge" in this section.)

Defining the Sort or Merge

The sort/merge program must know what to do with your input data. The program needs a general description of the input data, information about the control fields in the input records, and a description of your modification routines, if any, that will be used during sort/merge execution. Sort/merge control statements supply this information to the program.

Control statement formats for all System/360 sort/merge programs are constant even though operating environments and data descriptions are different. Compatibility of control statements among System/360 sort/merge programs is discussed later in this section. The five control statements that are acted upon by the operating system sort/merge program are:

SORT Statement   Provides information about control fields and data set size. Use this statement if your job is a sort. Do not use this statement for a merge-only job.

MERGE Statement   Provides the same information as a SORT statement. Use this statement if your job is a merge. Do not use this statement for a sort operation.

RECORD Statement   Provides record length and type information. This statement is required only when your modification routines change record lengths during sort/merge execution.

MODS Statement   Associates your modification routines with particular sort/merge program exits. This statement is required only when you supply modification routines to be executed at sort/merge exits. ("Section 3: Program Modification", describes these exits and the requirements for routines that use them.)

END Statement   Signifies the end of a related group of sort/merge control statements. This statement is required when sort/merge statements are not followed immediately in the input stream by a /* statement.

Each statement is checked for validity before it is acted upon by the sort/merge program. If the program finds an error, it issues a diagnostic message. (See Appendix D for descriptions of messages.) However, the program may not be able to detect all errors or inconsistent combinations of entries so you should be very careful in preparing control statements.

CONTROL STATEMENT FORMAT

All sort/merge control statements have the same general format:

```
┌Column 1 must
│ be blank                                                    72 73-80
↓
┌ Operation  Operand   Comments                              ↑ Sequence or
│                                                               Identification
│                                                        _____
│                                                        Continuation Column
│
```
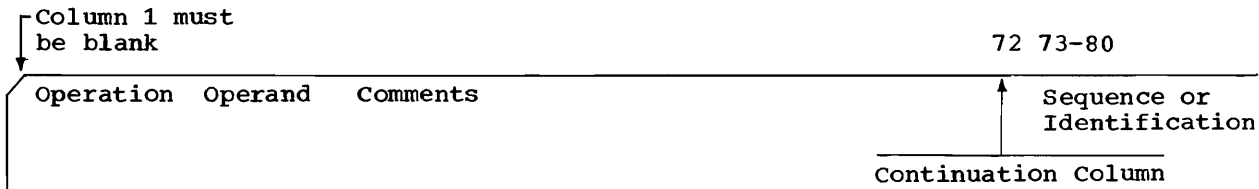
The control statements are free-form; that is, the operation definer, operand(s),
and comments may appear anywhere in a statement, as long as they appear in the
proper order, and are separated by one or more blank characters. Column 1 of each
control statement must be blank.

Operation Field: This field must appear first on the card. It must not extend
beyond column 71 of the first card. It contains a word (SORT, MERGE, RECORD,
MODS, or END) that identifies the statement type to the sort/merge program. In
Figure 6, the operation definer SORT is in the operation field of the sample
control statement.

Operand Field: The operand field is made up of one or more operands separated by
commas. This field must be the second field on the card and be separated from the
operation field by at least one blank. If the statement occupies more than one
card, this field must begin on the first card. Operands supply parameters to the
sort/merge program. Each operand is made up of an operand definer, or keyword (a
group of characters that identifies the operand type to the sort/merge program).
A value or values may be associated with a keyword. The three possible operand
formats are:

- keyword= (value$_1$,value$_2$,...,value$_n$)
- keyword=value
- keyword

Figure 6 contains an example of each of these formats.

```
┌──────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                               │
│be blank                                                    72     73-80     │
││                                                                            │
│↓                                                                           │
│/SORT    FIELDS=(10,30,A),FORMAT=CH,CKPT                   X      000001     │
│ ↑         ↑              ↑        ↑                        ↑        ↑        │
│ _____   _____          ____     ___                     _____   _____│
│Operation  Operand 1  Operand 2  Operand 3                Continuation Sequence│
│Field                                                      Punch        Field  │
│                                                                            │
└──────────────────────────────────────────────────────────────────────────┘
```
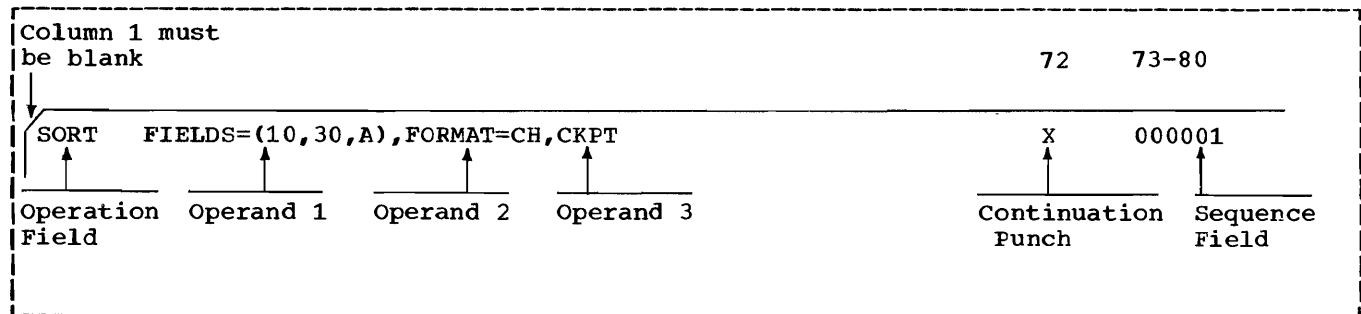
Figure 6. Control Statement Example

Comments Field: This field may contain any information you desire. It is not
required, but if it is present, it must be separated from the operand field by at
least one blank. Message IER009I appears for each statement containing comments.

Continuation Column (72): Any character other than a blank in this column
indicates that the present statement is continued on the next card. In Figure 6,
X is used to specify that the next card contains more information pertaining to
this SORT control statement.

Columns 73-80: This field may be used for any purpose you desire. It may be used
for identification, or as shown in Figure 6, for sequencing.

22

Continuation Cards

The format of the sort/merge continuation card is:

```
Column
| 16                                                     72 73-80
    ↳Continued Operand or Comments               ↑   Sequence
                                                 Continuation Column
```

   The continuation column and columns 73-80 of a continuation card fulfill the same purpose as they do on the first card of a control statement. Columns 1 through 15 of a continuation card must be blank. The maximum number of continuation cards allowed for each type of control statement is shown in the following table:

Control Statement Type	Maximum Number of Continuation Cards
SORT	19
MERGE	19
RECORD	5
MODS	19
END	none allowed

   A continuation card is treated as a logical extension of the preceding card. Either an operand or a comments field may begin on one card and continue on the next. The following rules apply to continuing operands or comments fields:

* If an operand is continued through column 71, the next character of the operand must appear in column 16 of the continuation card. Columns 1-15 must be left blank. For example:

```
                                              71 ──┐ ┌── 72
                                                   ↓ ↓
                                                     FORX
         MAT=CH
         ▲
         16
```
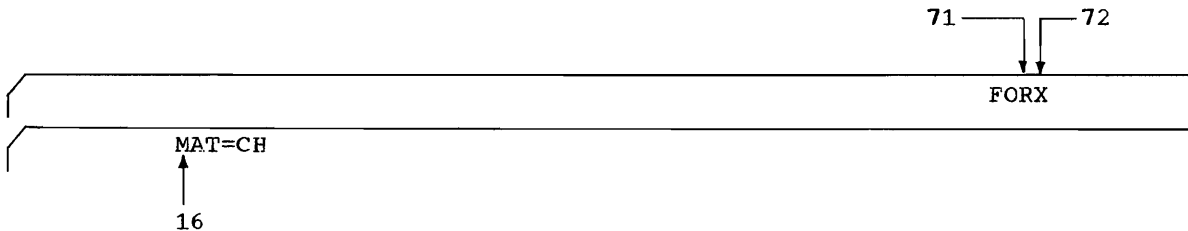
* If an operand field is broken between two cards without filling the first card through column 71, it must be done in either of two ways:

   1. At the end of a complete operand followed by a comma and a blank (or blanks). For example:

```
SORT  FIELDS=(10,30,A),                                           X
         FORMAT=CH,CKPT
```
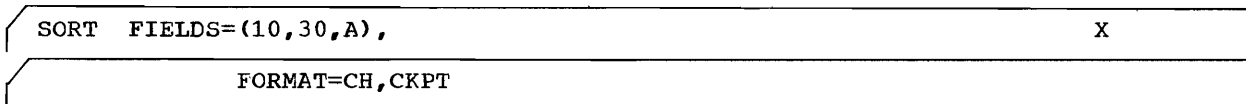
   2. At the end of any of the values in an operand of the type keyword=$(value_1, value_2,...,value_n)$, followed by a comma and a blank. For example:

```
SORT  FIELDS=(10,                                                 X
         30,A),FORMAT=CH,CKPT
```
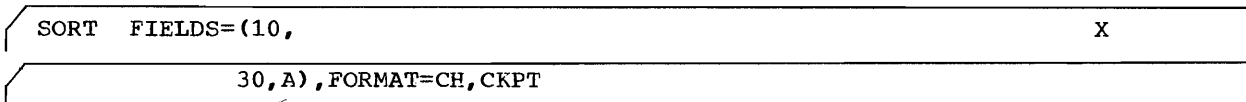
The following rules apply to control statement preparation:

- Column 1 of each control statement must be blank.

- The operation field must be the first field on the first card of a control statement and may not be carried over onto a continuation card.

- The operand field, if present, must begin on the first card of a control statement. The last operand in a statement must be followed by at least one blank.

- Embedded blanks are not allowed in operands. Anything following a blank is considered part of the comments field.

- Values may contain no more than eight alphameric characters.

- Commas and blanks can be used only as field delimiters. They must not be used in values.

- Each type of sort/merge control statement may appear only once for each execution of the sort/merge program.

- No more than 33 control statement cards, including continuation cards, are allowed for a sort/merge program execution.

SORT CONTROL STATEMENT

The SORT control statement must be used when a sorting application is to be performed. It describes the control fields on which the program will sort.

The format of the SORT statement is shown in Figure 7. The first field in the statement must be the operation definer SORT, followed by at least one blank.

```
Column 1 must
be blank

            FIELDS= ⎧(p₁,m₁,f₁,s₁,....,p₆₄,m₆₄,f₆₄,s₆₄)        ⎫
   SORT     ⎨                                                  ⎬ [,SIZE=y] [,SKIPREC=z] [,CKPT]
            FIELDS= ⎩(p₁,m₁,s₁,....,p₆₄,m₆₄,s₆₄),FORMAT=xx     ⎭
```
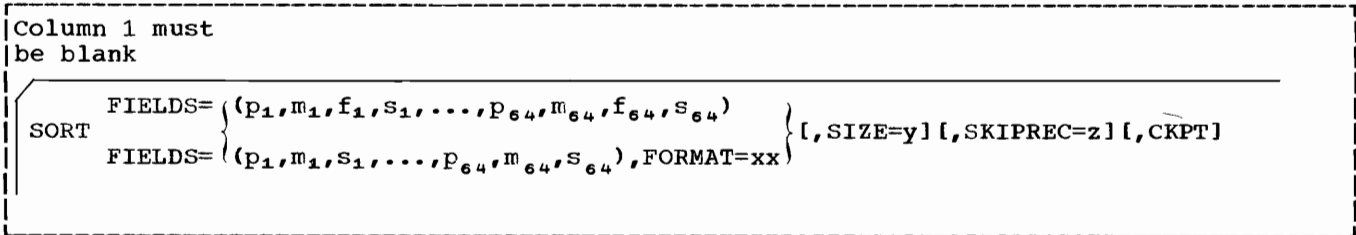
Figure 7. SORT Control Statement Format

Parameters

The FIELDS operand describes control fields. As shown in Figure 7, it can be written in two ways. Use the FIELDS format shown at the top of Figure 7 to describe control fields that contain different data formats. Use the format at the bottom of the figure to describe control fields that contain data of the same format. The format at the bottom of the figure is optional; you can always use the top format if you prefer.

The sort/merge program requires four facts about each control field in the input records: the position of the field within the record, the length of the field, the format of the data in the field, and the sequence into which the field is to be sorted. These facts are communicated to the program by the values of the FIELDS operand which are represented by p, m, f, and s in Figure 7.

The major control field, the one sort examines first, is specified first. Successive minor control fields are specified following the major control field. Up to 64 control fields can be used. In Figure 7, p_1,m_1,f_1,s_1 describe the major control field. $p_2,m_2,f_2,s_2,....,p_{64},m_{64},f_{64},s_{64}$ describe the successive minor control fields.

24

<u>p</u>
specifies the beginning (high-order location) of a control field relative to the beginning of the record which contains the control field. (For variable-length records, the logical record includes the four-byte record length indicator.) The first (high-order) byte in a record is byte 1, the second is byte 2, etc. All control fields, except binary, must begin on a byte boundary. Fields containing binary values are described in bytes and bits as follows:

First give the byte location relative to the beginning of the record and follow it with a period. Then give the bit location relative to the beginning of that byte. The resulting notation is then -- bytes.bits. The first (high-order) bit of a byte is bit 0; the remaining bits are numbered 1 through 7.

Thus, 1.0 represents the beginning of a record. A binary field beginning on the third bit of the third byte of a record is represented as 3.2. When the beginning of a field falls on a byte boundary, (say, for example, the fourth byte) you can write it in one of three ways:

    4.0
    4.
    4

Other examples of this notation are:
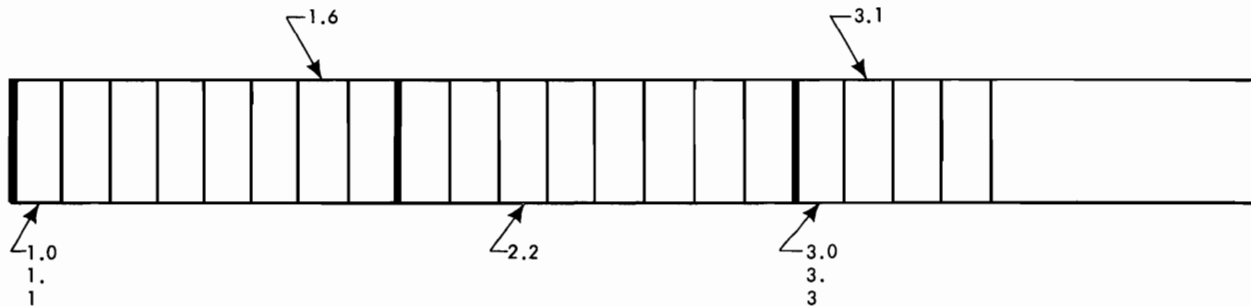
<u>m</u>
specifies the length of the control field. All control fields except binary must be a whole number of bytes long. The length of a control field that is a whole number (<u>d</u>) bytes long can be expressed in one of three ways:

    d.0
    d.
    d

Binary fields are expressed in the notation -- <u>bytes.bits</u>. The number of bits specified must not exceed 7. A control field two bits long would be represented as 0.2.

<u>f</u>
specifies the format of the data in the control field. <u>f</u> can be any one of the following two-character abbreviations:

    CH -- Character
    ZD -- Zoned decimal
    PD -- Packed decimal
    FI -- Fixed-point
    BI -- Binary
    FL -- Floating-point

If all the control fields contain the same type of data, you can omit the f parameters and use the optional FORMAT=xx operand.

The table below contains the data formats, indicates whether or not they are signed, and shows the maximum control field length for each format.

FORMAT	SIGNED	NUMBER OF BYTES
CH	NO	1-256
ZD	YES	1-32
PD	YES	1-32
FI	YES	1-256
BI	NO	1 bit - 256 bytes
FL	YES	1-256

s

specifies how the control field is to be ordered. One of the following one-character codes must be used for s:

    A -- Ascending sequence
    D -- Descending sequence
    E -- User modification

If you are including your own routine to modify control fields before the sort/merge program sequences them, use E. After your program has modified the control fields, the sort/merge program orders the fields in absolute ascending sequence. (See "Exit E61", described in "Section 3: Program Modification", for further information about modifying control fields.)

Options

You can use the following optional operands with the SORT control statement.

FORMAT=xx: If all the control fields contain the same type of data, you can use this operand instead of the f parameter of the FIELDS operand to specify the data format. If all the control fields are not of the same type, you must use the f parameter of the FIELDS operand. The possible values for xx are the same as those for the f parameter.

SIZE=y: This operand specifies the number of records in the input data set. The value y can be either the actual data set size or an estimate of the size.

If you give an actual data set size, do not include any records inserted in the input data set by one of your routines. If the number of records in the input data set, as counted by the sort/merge program, does not agree with the value of the SIZE parameter, the sort terminates. The value specified in the SIZE parameter is placed in the IN field of message IER047A or IER054I. If you give an estimated data set size, precede the value by E (for example, E5000).

If you omit the SIZE operand, the sort/merge program assumes that:

• If intermediate storage is tape, the input data set can be contained on one volume at the blocking factor used by the sort.

• If intermediate storage is direct access, the input data set will fit into the space you have allocated.

SKIPREC=z: If you want the sort to skip a certain number of records before starting to process the input data set, use this operand. Substitute the number of records you want skipped for z. On a preceding sort/merge program execution you may have exceeded storage capacity and only part of your input data was sorted. (The program prints a message specifying the number of records sorted in a partial run.) Using this operand, you could request that sort skip over the records it processed in the preceding run and sort the remaining records. You could then merge the output from the two sort runs to complete the sort/merge operation.

If you were using a routine to insert or delete records in a run during which sort capacity was exceeded, you will have to provide a routine that will

reposition the modified data set before the second part of the data set can be sorted.

CKPT: This operand tells the sort/merge program to activate the checkpoint facility of the operating system. The program takes checkpoints at the start of the sort phase and at the start of the final merge phase. In addition when the balanced or polyphase tape techniques are used, the program takes a checkpoint at the start of each intermediate merge phase pass. If the oscillating tape technique is used, the program takes checkpoints at intervals during the intermediate merge phase.

In addition to those taken at the beginning of each pass, the balanced direct access technique takes checkpoints at selected intervals during the intermedate merge phase.

You can have the program restart from the last checkpoint taken or from the checkpoint written at the start of the sort phase.

When you use the checkpoint/restart facility, you must define a data set for the checkpoint records. The data set is described further in this section under "Job Control Language for Sort/Merge".

The following rules apply to the control fields described on a SORT control statement:

- All control fields must be located within the first 4,092 bytes of a record.

- The first byte of a floating-point field is interpreted as a signed exponent. The rest of the field is interpreted as the fraction.

- All floating-point data must be normalized before the sort/merge program can collate it properly. You can use your own routine to do this at execution time. (See "Exit E61" in "Section 3: Program Modification".) Specify the E option for the value of s in the FIELDS operand for each control field you are going to modify.

- The total number of bytes occupied by all control fields must not exceed 256. A binary field is considered to occupy an entire byte if it occupies any part of it. For example, a binary field that begins on byte 2.6 and is 3 bits long occupies two bytes.

This three bit binary control field

Occupies two bytes
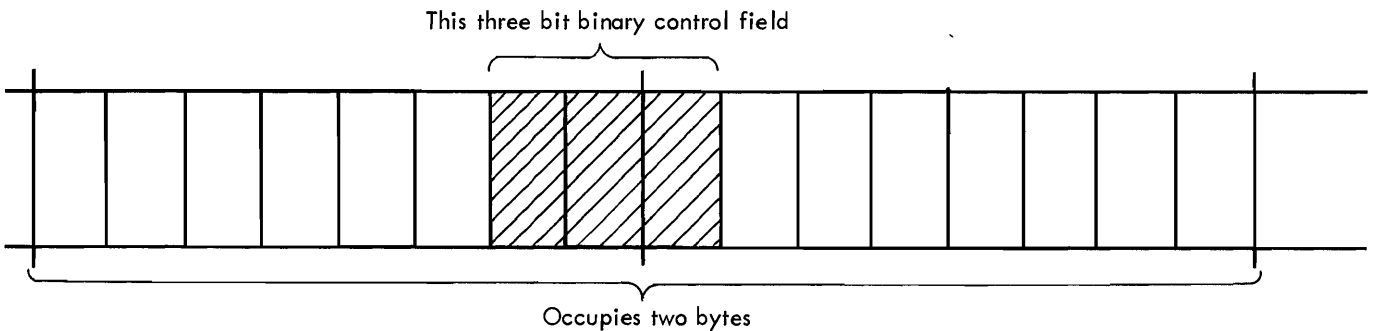
<u>SORT Statement Examples</u>

```
 _____
|Column 1                                                                    |
|must be blank                                                               |
|                                                                            |
|   _____                          |
|  /                                                                         |
|  |  SORT   FIELDS=(2.0,5.0,CH,A),SIZE=29483                                |
| ||                                                                         |
|_||_____|
```

SORT Statement Example 1. One Control Field and Size Option.

FIELDS operand
     2.0 means that the control field begins on the second byte of each record in
         the input data set.
     5.0 means the control field is five bytes long.
     CH means the control field contains character data.
     A   instructs the program to sort the fields into ascending order.

SIZE operand
     The input data set contains exactly 29,483 records.


```
 _____
|Column 1 must                               Column 72 _____           |
|be blank                                                        |           |
|                                                                |           |
|   _____v___        |
|  /                                                                         |
| |/  SORT FIELDS=(7.0,3.0,CH,D,1.0,5.0,FI,A,398.4,7.6,BI,D,99.0,230.2,BI,A,X |
| ||                                                                         |
| |                                                                          |
| |         Column 16                                                        |
| |            |                                                             |
| |            v                                                             |
| |   _____  |
| |  /                                                                        |
| |/          452.0,8.0,FL,A),SIZE=10693,CKPT                                 |
| ||                                                                         |
|_||_____|
```

SORT Statement Example 2. Five Control Fields, Size and Checkpoint Options

FIELDS operand   The first four values describe the major control field. It
                        begins on byte 7 of each record, is 3 bytes long, contains
                        character data, and is to be sorted into descending sequence.
                        The next four values describe the second control field. It
                        begins on byte 1, is 5 bytes long, contains fixed-point data,
                        and is to be sorted into ascending sequence.
                        The third control field begins on bit 5 (bits are numbered 0
                        through 7) of byte 398. The field is 7 bytes and 6 bits long
                        (occupies 9 bytes), and contains binary data to be placed in
                        descending order.
                        The fourth control field begins on byte 99, is 230 bytes and 2
                        bits long, contains binary data, and should be sorted into
                        ascending order.
                        The fifth control field begins on byte 452, is 8 bytes long,
                        contains normalized floating-point data which is to be sorted
                        into ascending order. If the data in this field was not
                        normalized, you would specify E instead of A and include your
                        own routine to normalize the field, before sort/merge examines
                        them.

SIZE operand   The input data set contains exactly 10693 records.

CKPT operand   Instructs the sort/merge program to take checkpoints during
                        this run.

```
┌─────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                  │
│be blank                                                                       │
│                                                                               │
│   ┌───────────────────────────────────────────────────────────               │
│  /                                                                            │
│ / SORT FIELDS=(3.0,8.0,ZD,E,40.0,6.0,CH,D),SIZE=E30000                        │
│ │                                                                             │
│ │                                                                             │
│                                                                               │
└─────────────────────────────────────────────────────────────────────────────┘
```

SORT Example 3.   Two Control Fields, User Modification, Size Option

FIELDS operand   The first four values describe the major control field. It
                      begins on byte 3 of each record, is 8 bytes long, contains
                      zoned decimal data that will be modified by your routine
                      before sort examines the field.
                      The second field begins on byte 40, is 6 bytes long, contains
                      character data and will be sorted into descending sequence.

SIZE operand   The input data set contains approximately 30,000 records.

```
┌─────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                  │
│be blank                                                                       │
│                                                                               │
│   ┌───────────────────────────────────────────────────────                   │
│  /                                                                            │
│ / SORT FIELDS=(25,4,A,48,8,A),FORMAT=ZD                                       │
│ │                                                                             │
│ │                                                                             │
│                                                                               │
└─────────────────────────────────────────────────────────────────────────────┘
```

SORT Statement Example 4.   Two Control Fields, Format Option

FIELDS operand   The major control field begins on byte 25 of each record, is 4
                      bytes long, contains zoned decimal data (FORMAT=ZD) , and is to
                      be sorted into ascending sequence.
                      The second control field begins on byte 48, is 8 bytes long,
                      has the same data format as the first field, and is also to be
                      sorted into ascending order.
                      The FORMAT=xx option can be used because both control fields
                      have the same data format. It would also be correct to write
                      this SORT statement as follows:

```
┌─────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                  │
│be blank                                                                       │
│                                                                               │
│   ┌───────────────────────────────────────────────────────                   │
│  /                                                                            │
│ /SORT FIELDS=(25,4,ZD,A,48,8,ZD,A)                                            │
│ │                                                                             │
│ │                                                                             │
│                                                                               │
└─────────────────────────────────────────────────────────────────────────────┘
```

MERGE CONTROL STATEMENT

    The MERGE control statement must be used when a merge-only operation is to be
performed. It provides essentially the same information to the sort/merge program
for a merge as the SORT statement does for a sort. As you can see from Figure 8,
the format of the MERGE statement is very much like that of the SORT statement.
There are the following differences:

   • The operation definer is MERGE.

   • The SKIPREC and CKPT options are not used.

   • The value of the SIZE operand is the total number of records in all the input
     data sets.

```
┌─────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                  │
│be blank                                                                       │
│                                                                               │
│  ╱─────────────────────────────────────────────────────────────              │
│ │       ⎧FIELDS=(p₁,m₁,f₁,s₁,p₂,m₂,f₂,s₂,...,p₆₄,m₆₄,f₆₄,s₆₄)⎫              │
│ │ MERGE⎨                                                       ⎬ [,SIZE=y]   │
│ │       ⎩FIELDS=(p₁,m₁,s₁,p₂,m₂,s₂,...,p₆₄,m₆₄,s₆₄),FORMAT=xx⎭              │
│ │                                                                             │
│                                                                               │
└─────────────────────────────────────────────────────────────────────────────┘
```

Figure 8. MERGE Control Statement Format

Parameters

The FIELDS operand is written exactly the same way for a merge as it is for a
sort. The meanings of p, m, f, and s were described previously in the discussion
of the SORT statement.

   The SIZE operand is optional. Its value can be either exact or estimated. The
value refers to the total number of records in all the input data sets to be
merged.

MERGE Statement Examples

```
┌─────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                  │
│be blank                                                                       │
│                                                                               │
│  ╱──────────────────────────────────────────────────────────                 │
│ │  MERGE FIELDS=(2.0,5.0,CH,A),SIZE=29483                                     │
│ │                                                                             │
│                                                                               │
└─────────────────────────────────────────────────────────────────────────────┘
```

MERGE Statement Example 1. One Control Field, Size Option

FIELDS operand   The control field begins on byte 2 of each record in the input
                      data sets. The field is 5 bytes long, and contains character
                      data that has been presorted into ascending order.

SIZE operand   The input data sets contain exactly 29,483 records.


```
┌─────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                  │
│be blank                                                                       │
│                                                                               │
│  ╱──────────────────────────────────────────────────────────                 │
│ │  MERGE FIELDS=(3.0,8.0,ZD,E,40.0,6.0,CH,D),SIZE=E30000                      │
│ │                                                                             │
│                                                                               │
└─────────────────────────────────────────────────────────────────────────────┘
```

MERGE Statement Example 2. Two Control Fields, User Modification, Size Estimate

FIELDS operand   The major control field begins on byte 3 of each record, is 8
                      bytes long, and contains zoned decimal data that will be
                      modified by your routine before the merge examines it.

                      The second control field begins on byte 40; is 6 bytes long,
                      and contains character data that is in descending order.

SIZE operand   The input data sets contain approximately 30,000 records.

30

```
r----------------------------------------------------------------------------
|Column 1 must                                                               |
|be blank                                                                    |
|                                                                            |
|  /-----------------------------------------------------------------------  |
| /  MERGE FIELDS=(25,4,A,48,8,A),FORMAT=ZD                                  |
||                                                                     .      |
||                                                                            |
|                                                                            |
L----------------------------------------------------------------------------J
```

MERGE Statement Example 3.   Two Control Fields, Format Option

FIELDS operand   The major control field begins on byte 25 of each record, is 4
                        bytes long, and contains zoned decimal data that has been
                        placed in ascending sequence.
                        The second control field begins on byte 48, is 8 bytes long,
                        is also in zoned decimal format, and is also in ascending
                        sequence. The FORMAT=xx option can be used because both
                        control fields have the same data format.

RECORD CONTROL STATEMENT

The RECORD statement is required only when your routines change record lengths
during a sort/merge program run. The statement describes the format and lengths
of the records being sorted or merged. The format of the RECORD statement is
shown in Figure 9.

Parameters

The RECORD statement has two operands, TYPE and LENGTH. Both are required when
the RECORD statement is used.

TYPE: The TYPE operand specifies whether the input records to sort/merge are
fixed- or variable-length format.

TYPE=F indicates fixed-length records.
TYPE=V indicates variable-length records.

LENGTH: The LENGTH operand specifies the length in bytes of the input records,
the length in bytes of the records that enter the sort phase of the sort/merge
program, (you can include your own routine to change record lengths before the
records are sorted), and the length in bytes of the records in the output data
set. (You can change record lengths during the final merge phase of the program.)

   The value l_1 is required whenever the RECORD statement is used. The values l_2
and l_3 are required only when your routines change record lengths before the sort
or during the final merge. The values 1 and 1 are used only for variable-length
records.


```
r----------------------------------------------------------------------------
|Column 1 must                                                               |
|be blank                                                                    |
|                                                                            |
|  /-----------------------------------------------------------------------  |
| /          (TYPE=F,LENGTH= (l1,l2,l3)            )                         |
||  RECORD  {                                       }                        |
||          (TYPE=V,LENGTH= (l1,l2,l3,l4,l5) )                               |
||                                                                            |
|                                                                            |
L----------------------------------------------------------------------------J
```

Figure 9. RECORD Control Statement Format

Defining Fixed-Length Records

If your input records are fixed-length, use l_1, l_2, and l_3 as follows:

l_1
> is the length of each record in the input data set. If you use the RECORD control statement, you must include this value. The value should be the same as the value you specified in the LRECL subparameter of the DCB parameter on the SORTIN DD statement (discussed later in this section.) If the values are not the same, sort/merge uses the value specified on the DD statement.

l_2
> is the length of each record handled by the sort phase. If you do not specify a value for l_2, the program assumes that it is equal to l_1 as it is specified on the record card. If you are going to change record lengths in the sort phase, you must include a value for l_2. You do not need l_2 for a merging application.

l_3
> is the length of each record in the output data set. If you do not specify a value for l_3, the program assumes that $l_3=l_2$ for a sorting application and that $l_3=l_1$ for a merging application. If your routines change record lengths during the final merge phase of the program, you must specify a value for l_3. This value should be the same as the value you specified for the LRECL subparameter of the DCB parameter on the SORTOUT DD statement (discussed later in this section). If the values are different, the sort/merge program uses the value given on the DD statement.

Defining Variable-Length Records

If your input records are variable-length, use l_1, l_2, l_3, l_4, and l_5 as follows:

l_1
> is the maximum length of the records in the input data set. If you use the RECORD statement, you must specify a value for l_1. The value should be the same as the value you specified in the LRECL subparameter of the DCB parameter on the SORTIN DD statement (discussed later in this section). If the values are not the same, the program uses the LRECL value.

l_2
> is the maximum length of the records handled by the sort phase. If you do not specify a value for l_2, the program assumes it is equal to l_1 as it is specified on the record card. If you change record lengths in the sort phase, you must provide a value for l_2. You do not need l_2 for a merging application.

l_3
> is the maximum length of each record in the output data set. If you do not specify a value for l_3, the program assumes $l_3=l_2$ for a sort and $l_3=l_1$ for a merge. If you include a routine that changes record lengths in the final merge phase, you must specify a value for l_3. The value should be the same as the value you provided for the LRECL subparameter of the DCB parameter on the SORTOUT DD statement. If it is not, the program uses the LRECL value.

l_4
> is the minimum length of records in the input data set. If you do not specify a value for l_4, the program assumes it is equal to the minimum record size necessary to contain the control fields defined on the SORT or MERGE control statement, or the minimum record length allowed by the operating system, whichever is greater. You need not specify this value for a merge.

l_5
> is the record length that occurs most frequently in the input data set (modal length). You should use this value to help define a data set biased toward a particular length. If you do not specify a value for l_5, the program assumes

it is equal to the average of the maximum and minimum record lengths in the input data set. If, for example, your data set contains mostly small records and just a few long records, the program would assume a high modal length and would allocate a larger record storage area than necessary. Conversely, if your data set contains just a few short records and many long records, the program would assume a low modal length and might not allocate a large enough record storage area to sort your data.

When you use the RECORD statement, consider the following:

* The lengths you specify for variable length records must include the 4-byte count field that the operating system places at the beginning of each record.

* When you use a direct access device for intermediate storage, record length cannot exceed the capacity of one track.

* The sort/merge control fields defined in the SORT control statement must be contained in the minimum length l_4, if stated.

* The minimum logical record length that can be handled by the program is 18 bytes.

* The record format you specify in the TYPE operand must be the same as the format you used in the RECFM subparameter of the DCB parameter on the SORTIN and SORTOUT DD statements (described later in this section). If the formats are not the same, the program uses the one you specified in the DD statement.

* When you use an operand like the LENGTH operand of the type, keyword=$(value_1,value_2,...,value_n)$, you can omit values that are equal to those assumed by the program. The following rules apply to omitting values from the LENGTH operand:

  1. You can drop values from right to left. If all the values after l_2 are equal to the values assumed by the program, you could write -- LENGTH=(l_1,l_2).

  2. If you drop values from the middle or from left to right, you must use commas to indicate their omission. If l_2 is equal to the value assumed by the program, you could write -- LENGTH=$(l_1,,l_3)$.

RECORD Statement Examples

```
Column 1 must
be blank


 RECORD TYPE=F,LENGTH=(60,40,50)


```

RECORD Statement Example 1. Fixed-length, Three Length Values

TYPE operand   The input records are fixed-length.

LENGTH operand   The records in the input data set are each 60 bytes long. You change the records to 40 bytes in the sort phase and to 50 bytes in the final merge phase.

```
r--------------------------------------------------------------------¬
|Column 1 must                                                       |
|be blank                                                            |
|      _____  |
|     /                                                              |
|    /  RECORD TYPE=V,LENGTH=(200,175,180,50,100)                    |
| |                                                                  |
| |                                                                  |
L--------------------------------------------------------------------┘
```

RECORD Statement Example 2.   Variable-length, Five Length Values

TYPE operand   The records in the input data set are variable-length.

LENGTH operand   The maximum length of the records in the input data set is 200
                      bytes. In the sort phase, you reduce the maximum record
                      length to 175 bytes. You add five bytes to each record in the
                      final merge phase, making the maximum record length in the
                      output data set 180 bytes. The minimum record length in the
                      input data set is 50 bytes and the most frequent record length
                      in the input data set is 100 bytes.


```
r--------------------------------------------------------------------¬
|Column 1 must                                                       |
|be blank                                                            |
|      _____  |
|     /                                                              |
|    /  RECORD TYPE=F,LENGTH=(76,,50)                                |
| |                                                                  |
| |                                                                  |
L--------------------------------------------------------------------┘
```

RECORD Statement Example 3.   Fixed-length, Two Length Values

TYPE operand   The records in the input data set are fixed-length.

LENGTH operand   The input records are 76 bytes long. You do not change record
                      length in the sort phase so you omit l_2 because sort/merge
                      will assume the proper value for it. In the final merge
                      phase, you change the record length to 50 bytes.

MODS CONTROL STATEMENT

The MODS statement is required only if you want the sort/merge program to transfer
control to your routine(s) at various points during sort/merge execution. The
statement associates your routines with specific exits in the sort/merge program
and provides the program with basic descriptions of your routines. For details
about exits in the sort/merge program and how to use them, refer to "Section 3:
Program Modification."

    Figure 10 shows the format of the MODS statement.


```
r--------------------------------------------------------------------¬
|Column 1 must                                                       |
|be blank                                                            |
|      _____  |
|     /                    ⎡(N)⎤                          ⎡(N)⎤      |
|    /  MODS exit=(n₁,m₁,s₁⎢,{ }⎥),....,exit=(n₁₇,m₁₇,s₁₇⎢,{ }⎥)     |
| |                        ⎣(S)⎦                          ⎣(S)⎦      |
| |                                                                  |
L--------------------------------------------------------------------┘
```

Figure 10. MODS Control Statement Format

Parameters

The sort/merge program provides seventeen exits at which control can be transferred to your routines. These exits are described in detail in "Section 3: Program Modification". The exits have three-character names such as E11, E15, E16, E28, etc. To use one of these exits, you substitute its three-character name for the word "exit" in the MODS statement format example. The values associated with the three-character name describe your routine. These values are:

n

       the name of your routine (member name if your routine is in a library). If your routine has been linkage edited previously and you do not want to have it linkage edited again, its name must be the same as the three-character exit name with which it is associated.

m

       the number of bytes, exact or approximate, of main storage that your routine uses.

s

       either the name of the DD statement in your sort/merge job step that defines the partitioned data set in which your routine is located, or SYSIN if your routine is in the input stream. If your routines are in a concatenated data set the value of s for all the routines must be the ddname of the data set.

$\begin{Bmatrix} N \\ S \end{Bmatrix}$

       indicates the linkage editor requirements of your routine.

  N means that your routine has already been linkage edited and can be used in the sort/merge run without further linkage editing. Your routines must be in the same library or in libraries defined as a concatenated data set with one DD name.

  S means that your routine requires linkage editing but that it can be linkage edited separately from the other routines you are using in a particular sort/merge program phase. Only routines at exits E11, E21, and E31 are eligible for separate linkage editing; see Section 3.

       Absence of these parameters means that your routine must be linkage edited together with the other routines you are using in a particular sort/merge program phase.

       Refer to the topic "Bypassing the Linkage Editor" in "Section 3: Program Modification" for details on how to design your routines.

When you are preparing your MODS statement, consider the following:

- The sort/merge program must know the amount of main storage your routine needs so that it can allocate main storage properly for its own use. If you do not know the exact number of bytes your program requires, make a slightly high estimate. The value of m in the MODS statement is written the same whether it is an exact figure or an estimate. In other words, you do not precede the value by E for an estimate as you did on the SORT or MERGE statement.

- If the routines you are using for a particular sort/merge run are in several system libraries, you need a DD statement for each library. DD statements required for sort/merge are described later in this section.

- If your routines are in the system input stream (SYSIN), you must arrange them in numerical order (the E11 routine before the E15 routine, etc.). If you use the same routine in several sort/merge program phases, you must provide a separate copy of the routine for each use.

- Your routines can also reside in private libraries. The use of private libraries is described in the publication OS JCL Reference.

MODS Statement Examples

```
┌────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                 │
│be blank                                                                      │
│                                                                              │
│  ┌─────────────────────────────────────────────────────────────────────     │
│ /  MODS E15=(E15,554,MODLIB,N),E35=(E35,11032,MODLIB,N)                      │
│││                                                                            │
└────────────────────────────────────────────────────────────────────────────┘
```

MODS Statement Example 1.   Two Routines in a Library, No Linkage Editing

E15   At exit E15, the sort/merge program will transfer control to your routine.
      Your routine is in the library defined by the MODLIB DD statement. Its
      member name is E15, it is 554 bytes long, and has been linkage edited
      previously, and does not require further linkage editing.

E35   At exit E35, the program will transfer control to your routine. Your
      routine is in the library defined by the MODLIB DD statement, its member
      name is E35, it is 11032 bytes long and has been linkage edited previously.

```
┌────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                 │
│be blank                                                                      │
│                                                                              │
│  ┌─────────────────────────────────────────────────────────────────────     │
│ /  MODS E17=(CLSE,348,SYSIN)                                                 │
│││                                                                            │
└────────────────────────────────────────────────────────────────────────────┘
```

MODS Statement Example 2.   One Routine in SYSIN, Linkage Editing is Needed

E17   At exit E17, the sort/merge program will transfer control to your routine
      which is named CLSE. Your routine is in object form in the system input
      stream and will be linkage edited together with other routines in the sort
      phase of the sort/merge program.

```
┌────────────────────────────────────────────────────────────────────────────┐
│Column 1 must                                                                 │
│be blank                                   Column 72 ──────┐                  │
│                                                           │                  │
│  ┌────────────────────────────────────────────────────────▼───────          │
│ /  MODS E16=(NMAXERR,1000,MYLIB),E21=(E21,550,MODLIB,N),        X            │
│││                                                                            │
│  ┌─────────────────────────────────────────────────────────────────         │
│ /        E31=(E31,450,MODLIB,N),E35=(SUMUP,5000,SYSIN)                       │
│││                                                                            │
└────────────────────────────────────────────────────────────────────────────┘
```

MODS Statement Example 3.   Four Routines

E16   The sort/merge program will transfer control to your routine at exit E16.
      Your routine is named NMAXERR, is located in the library defined by the
      MYLIB DD statement, and is approximately 1000 bytes long.

E21   At exit E21, the program will transfer control to your routine which resides
      in the library defined by the MODLIB DD statement under the member name E21.
      Your routine is 550 bytes long and does not require additional linkage
      editing.

E31 Another of your routines in the library defined by the MODLIB CD statement will gain control at exit E31. Its member name is E31, it is 450 bytes long and does not require additional linkage editing.

E35 You have placed a routine named SUMUP in object form in the input stream. It is approximately 5000 bytes long, must be linkage edited together with other routines in its phase, and will receive control at exit E35.

```
r--------------------------------------------------------------------------1
|Column 1 must                                                             |
|be blank                                                                  |
|                                                                          |
|  _____  |
| /                                                                        |
|/  MODS    E11=(E11,500,MYLIB,S)                                          |
||                                                                         |
||                                                                        |
|                                                                          |
L--------------------------------------------------------------------------J
```

MODS Statement Example 4. One Routine, Separate Linkage Editing

E11 At exit E11 on the sort phase, the sort/merge program will transfer control to your routines. Your routine, named E11, is located in a library defined on a statement with the ddname MYLIB, is 500 bytes long, and can be linkage edited separately from other routines in the sort phase. After the sort phase is initialized, your E11 routine will be overlaid. Because you have specified S for separate linkage editing, your routine can have no external references.

END CONTROL STATEMENT

The END statement marks the end of all sort/merge control statements and continuation statements for a particular sort/merge run. The END statement must be used whenever the sort/merge control statements are not immediately followed in the input stream by a /* statement. For example, if you include you own routines in the input stream, they are placed between the sort/merge control statements and the /* statement, so you must use an END statement.

The format of the END statement is shown in Figure 11. The statement has no operands.

```
r--------------------------------------------------------------------------1
|Column 1 must                                                             |
|be blank                                                                  |
|                                                                          |
|  _____  |
| /                                                                        |
|/  END                                                                    |
||                                                                         |
||                                                                        |
|                                                                          |
L--------------------------------------------------------------------------J
```

Figure 11. END Control Statement Format

CONTROL STATEMENT COMPATIBILITY

There are eight control statement types used by System/360 sort/merge programs. The System/360 Operating System sort/merge program acts upon the SORT, MERGE, RECORD, MODS, and END statements described above. The three remaining control statement types, INPFIL, OUTFIL, and OPTION, are used only by other System/360 sort/merge programs. The operating system sort/merge program recognizes INPFIL, OUTFIL, and OPTION as valid control statements, but does not act upon them.

The information contained in INPFIL and OUTFIL statements is supplied to the operating system sort/merge program in DD statements. The information contained in the OPTION statement is specified at system generation time.

The operating system sort/merge program accepts SORT, MERGE, RECORD, and END statements used by other System/360 sort/merge programs. If these statements contain parameters not recognized by the operating system sort/merge program, the program ignores those parameters. However, because of differences in the way parameters are specified, the operating system sort/merge program will not accept MODS statements used by other System/360 sort/merge programs.

Summary of Sort/Merge Control Statements

SORT
$$\text{FIELDS} = \left(p_1, m_1, f_1, s_1, p_2, m_2, f_2, s_2, \cdot \cdot \cdot, p_{64}, m_{64}, f_{64}, s_{64} \right)$$
$$\text{FIELDS} = \left(p_1, m_1, s_1, p_2, m_2, s_2, \cdot \cdot \cdot, p_{64}, m_{64}, s_{64} \right), \text{FORMAT} = xx$$
$$\left[, \text{SIZE} = y \right] \left[, \text{SKIPREC} = z \right] \left[, \text{CKPT} \right]$$

MERGE
$$\text{FIELDS} = \left(p_1, m_1, f_1, s_1, p_2, m_2, f_2, s_2, \cdot \cdot \cdot, p_{64}, m_{64}, f_{64}, s_{64} \right)$$
$$\text{FIELDS} = \left(p_1, m_1, s_1, p_2, m_2, s_2, \cdot \cdot \cdot, p_{64}, m_{64}, s_{64} \right), \text{FORMAT} = xx$$
$$\left[, \text{SIZE} = y \right]$$

SORT and MERGE Statement Parameters

PARAMETER	EXPLANATION	LIMITATIONS	EXAMPLE	DEFAULT
p	Control field position within record.	All fields except binary must start on a byte boundary. No field may extend past byte 4092.	4.2 – a binary field starting on the 3rd bit of the 4th byte.	
m	Control field length.	Character 1 – 256 bytes Zoned Decimal 1 – 32 bytes Packed Decimal 1 – 32 bytes Fixed – Point 1 – 256 bytes Floating – Point 1 – 256 bytes Binary – 1 bit – 256 bytes	32 – a maximum length packed decimal field	
f	Control field data format.	Must be one of the following: CH, ZD, PD, FI, FL, or BI.	ZD – the code for a zoned decimal field	
s	Sequencing desired	Must be one of the following: A – ascending D – descending E – user modification then absolute ascending	E – exit E61 will modify the control field to achieve a unique sequencing.	
FORMAT = xx	Optional. Used when all control field data formats are the same.	XX must be CH, ZD, PD, FI, FL, or BI.	FORMAT = PD – all control fields are packed decimal.	
SIZE = y	Optional. The number of records in the input data set. May be an estimate.	If y is an estimate, precede value with the character E.	E40200 – an estimate of 40200 records.	
SKIPREC = z	Optional. Program will skip z records before sorting.	Not valid for a merge.	SKIPREC = 900 – the first 900 input records are ignored.	
CKPT	Optional. Checkpoints are taken.	Not valid for a merge.	CKPT	

END

The END statement must be used when user routines or data are in the input stream. The statement has no parameters.

RECORD TYPE = x, LENGTH = $\left(\ell_1, \ell_2, \ell_3, \ell_4, \ell_5 \right)$

RECORD Statement Parameters

PARAMETER	EXPLANATION		LIMITATIONS	EXAMPLE	DEFAULT
TYPE =x	Tells program whether input records are fixed or variable length.		x must be F or V	TYPE = V – input is variable – length	
ℓ_1	Fixed	Variable	ℓ_1 may not be less than 18 bytes See Figures 1 & 2 for maximum.	80 – fixed – length input records are 80 bytes, or maximum variable – length input record is 80.	
	Input record length	Maximum input record length			
ℓ_2	Length of input to sort phase	Maximum record length of input to sort phase	ℓ_2 is not used for a merge.	60 – input to sort phase is 60, or a maximum of 60. User routine has modified original input record length.	$\ell_2 = \ell_1$
ℓ_3	Record length of output records	Maximum record length of output records.	If specified, must be same as LRECL for output, or else LRECL is used.	90 – output records are all 90 bytes or a maximum of 90.	$\ell_3 = \ell_2$ for a sort $\ell_3 = \ell_1$ for a merge
ℓ_4		Minimum input record length.	Not used for fixed – length records.	30 – minimum variable – length record is 30 bytes	ℓ_4 = sum of control field lengths, or 18 bytes, whichever is greater.
ℓ_5		Modal input record length	Not used for fixed – length records.	50 – in a variable-length input data set, 50 bytes is the most frequently occurring length.	$\ell_5 = \dfrac{\ell_1 + \ell_4}{2}$

MODS exit = $\left(n_1, m_1, s_1 \left[, \begin{Bmatrix} N \\ S \end{Bmatrix} \right] \right), \ldots, exit = \left(n_{17}, m_{17}, s_{17} \left[, \begin{Bmatrix} N \\ S \end{Bmatrix} \right] \right)$

MODS Statement Parameters

PARAMETER	EXPLANATION	LIMITATIONS	EXAMPLE	DEFAULT
exit = xx	The name of an exit to be activated.	Must be a valid exit name.	E28	
n	Name of the routine. Member name if routine is in a library.		CHANGE1	
m	Size, in bytes, of the routine.		514	
s	Location of the routine.	Either the ddname of data set containing routines, or SYSIN.	USERLIB – the routine is in a data set defined by the DD statement named USERLIB.	
N	Tells if no additional link editing or separate link editing is required.	Must be the character N or S.	N – no additional link editing is required.	If not used, assumes link editing together.

40

Sort/Merge Control Statement Examples

Following are a number of examples showing groups of sort/merge control
statements. Each example shows all the sort/merge control statements that are
necessary to accomplish a particular job. However, these control statements must
be accompanied by job control language statements before the job can be run.
Later in this section the JCL required for sort/merge execution is discussed. At
the end of that discussion is a group of complete JCL and sort/merge control
statement examples. The operands and values of the sort/merge control statements
shown there are the same as the ones in these examples.

Example 1 - Simple Sort

This example shows a simple sorting application. No modification routines are
included so neither the RECORD nor the MODS statement is required.

```
    SORT FIELDS=(1.0,6.0,A,28,5,D),FORMAT=CH,SIZE=E10000
    END
```

SORT statement	The FIELDS operand describes two fields. The first begins on byte 1 of each record, is 6 bytes long, contains character data, and is to be sorted into ascending order. The second field begins on byte 28, is 5 bytes long, contains character data, and is to be sorted into descending order. The optional FORMAT operand is used because both fields contain data of the same format.
END statement	This statement is shown for completeness. It is not necessary since no modification routines which would come between the SORT statement and the /* statement are included.

Example 2 -- Simple Merge

This example shows a simple merge application. The values of the FIELDS operand
are the same as those on the SORT statement in Example 1. No modification
routines are included in this application.

```
    MERGE FIELDS=(1.0,6.0,A,28,5,D),FORMAT=CH,SIZE=E10000
    END
```


Example 3 -- Sorting With Modification Routines

This example shows a more complicated sorting application. Modification routines
are included, therefore a MODS statement is required. Some of the modification
routines change record lengths during sort/merge program execution, therefore a
RECORD statement is required.

```
    SORT FIELDS=(3.0,8.0,ZD,E,40.0,6.0,CH,D),SIZE=E30000

    RECORD TYPE=F,LENGTH=(120,100,80)

    MODS E15=(E15,780,MODLIB,N),E16=(E16,1024,MODLIB,N),           X
             E35=(ADDUP,912,SYSIN),E61=(CHGE,1000,SYSIN)

    END
```

SORT Statement	The FIELDS operand describes two control fields. The first will be changed by a modification routine (at exit E61, see the MODS statement) before sort/merge orders it into absolute ascending sequence. The second control field will not be modified and will be placed in descending sequence.

RECORD Statement	The fixed-length records in the input data set are 120 bytes long. A modification routine (at exit E15) changes them to 100 bytes during the sort phase. A modification routine (at exit E35) changes them again during the final merge phase (to 80 bytes each).
MODS Statement	The statement describes four modification routines. The first two are in a library that is defined on the MODLIB DD statement with member names of E15 and E16 respectively. Neither routine requires additional linkage editing. The next two routines are in object form in the input stream. Their names are ADDUP and CHGE, respectively. They must be linkage edited together with other routines in their phases that require linkage editing.
END Statement	This statement is required because of the modification routines in the input stream.

Example 4 - Merging With Modification Routines

This example is a merging application. Modification routines that change record lengths and control fields are included.

```
MERGE FIELDS= (5,6,CH,E) ,SIZE=8150
RECORD TYPE=V,LENGTH= (240,,200,,160)
MODS E35= (CALC,800,USERLIB) ,E61= (E61,450,MODLIB,N)
END
```

MERGE Statement	The FIELDS operand describes one control field that will be modified (by the routine at exit E61) before it is examined by the merge. The exact size of the input data sets is given.
RECORD Statement	All the records in the input data sets are variable-length. The maximum record length in the input data sets is 240. A modification routine (at exit E35) shortens all records by 40 bytes making the maximum record length in the output data set 200 bytes. The most frequent record length in the input data set is 160 bytes.
MODS Statement	A routine named CALC receives control at exit E35. CALC is approximately 800 bytes long, resides in the library defined on the USERLIB DD statement and must be linkage edited together with other routines in its phase which require linkage editing. At exit E61, the sort/merge program transfers control to a routine from the library defined by the MODLIB DD statement. The member name of this routine is E61. This routine is 450 bytes long and does not need further linkage editing.
END Statement	The END statement is not required because there are no modification routines in the input stream, but it is shown here for completeness.

Example 5 - Sort

This example shows a one-field sort with fixed-length records whose length is changed during the course of sort/merge execution by a routine at exit E35.

    SORT FIELDS=(10,5,CH,A),SIZE=10000

    RECORD TYPE=F,LENGTH=(80,,50)

    MODS E35=(E35,534,SYSIN)

    END

SORT Statement   The FIELDS operand describes one control field that begins on
                      byte 10 of each record, is 5 bytes long, contains character data,
                      and is to be sorted into ascending order. The optional SIZE
                      operand indicates that there are exactly 10,000 records in the
                      input data set.

RECORD Statement   This statement indicates that the input data set contains 80-byte
                      fixed-length records and that the records will be shortened to 50
                      bytes each as they leave the final merge.

MODS Statement   The statement describes a modification routine that will receive
                      control at sort/merge program exit E35. The name of the routine
                      is E35, it is 534 bytes long, appears in object form in SYSIN,
                      and must be linkage edited together with other routines in its
                      phase which require linkage editing.

END Statement   This statement is required because the sort/merge control
                      statements are not followed immediately by a /* statement. (The
                      E35 object deck follows the END statement in the input stream.)

Example 6 - Sort

This example shows a one-field sort with variable-length records. Modification routines receive control at exits E11 and E16.

    SORT FIELDS=(20,5,CH,A),SIZE=E25500

    RECORD TYPE=V,LENGTH=(120,,,60,80)

    MODS E11=(E11,500,USERLIB,S),E16=(E16,554,USERLIB,N)

    END

SORT Statement   The FIELDS operand describes one control field that begins on
                      byte 20 of each record, is 5 bytes long, contains character data,
                      and is to be sorted into ascending order. The optional SIZE
                      operand indicates that there are approximately 25,500 records in
                      the input data set.

RECORD Statement   This statement indicates that the input data set contains
                      variable-length records with a maximum record length of 120
                      bytes, a minimum record length of 60 bytes and a modal (most
                      frequent) length of 80 bytes. The RECORD statement is not
                      required for this example, but without it, sort/merge would
                      assume a minimum record length of 24 bytes (large enough to
                      contain the specified control field) and a modal length of 72
                      bytes (the average of maximum and minimum lengths).

MODS Statement   The statement describes two modification routines. One will
                      receive control at exit E11. It is named E11, is 500 bytes long
                      and can be linkage edited separately. (See "Bypassing the
                      Linkage Editor" in "Section 3: Program Modification", for a
                      description of the requirements for separate linkage editing.)

The E11 routine is in a library described on a DD statement with the ddname USERLIB. The other modification routine, named E16 will receive control at exit E16. The routine is 554 bytes long and the library in which it resides is described on the DD statement USERLIB. The E16 routine has been linkage edited previously and does not require further linkage editing prior to its use in this application.

END statement   This statement is not required in this example. It is shown for completeness only.

Example 7 - Sort

This example shows a two-field sort. A modification routine at E35 places part of the output data set on a device other than SORTOUT.

    SORT FIELDS=(1,10,CH,A,11,6,PD,D) ,SIZE=E15000

    MODS E35= (SUBSET,1024,SYSIN)

    END

SORT Statement   The FIELDS operand describes two control fields. The first is a 10-byte field beginning on byte 1. It contains character data which is to be sorted into ascending order. The second is a 6-byte field which begins on byte 11 and contains packed decimal data to be placed in descending order. The input data set contains approximately 15,000 records.

MODS Statement   A routine named SUBSET will receive control at sort/merge exit E35. The routine is 1024 bytes long, must be linkage edited together with other routines in the final merge phase of the program, and will appear in object form in SYSIN.

END Statement   This statement is required for this example because the SUBSET routine will appear in the input stream between the sort/merge control statements and the /* statement.

Determining Intermediate Storage Requirements

If you are performing a sorting application, you must calculate the amount of intermediate storage the sort/merge program needs to sort your data. The basic factors to consider are the type of device on which you assign intermediate storage and the number of records in your input data set. Another factor which must sometimes be weighed is the amount of main storage assigned to the sort/merge program. In general, the less main storage sort/merge has to operate in, the more intermediate storage it needs to complete a sorting application.

INTERMEDIATE STORAGE DEVICES

You can assign intermediate storage either on magnetic tape or direct access devices, but not on a mixture of both.

IBM 2400 Series or IBM 3400 Series Magnetic Tape Units can be used for intermediate storage. The sort/merge program can operate with a mixture of 7-track and 9-track tapes. If the sort input data set is on 7-track tape, you can use any combination of 7-track and 9-track tapes for intermediate storage and output, or intermediate storage and output can be on 2311 disks, 2314 storage facilities, or 2301 drums. However, if 7-track tape is not used for input, it cannot be used for intermediate storage or output. When 7-track tape is used for intermediate storage, variable length records cannot be handled.

If you assign 7-track tapes for input, you can use the data converter. If you assign 7-track tape for intermediate storage, you cannot use the data converter, nor can you use the translation feature for anything but character data.

If you use direct access devices for intermediate storage, use only one type of direct access device as intermediate storage for a given sorting application. The types of direct access devices available for intermediate storage are:

- IBM 2311 Disk Storage Drive.
- IBM 2301 Drum Storage Drive.
- IBM 2314 Direct Access Storage Facility.
- IBM 2319 Direct Access Storage Facility.

INTERMEDIATE STORAGE SPACE REQUIREMENTS

Use the following formulas to calculate the amount of intermediate storage necessary for a given sorting application, device type, and sequence distribution technique. Unless you force a sequence distribution technique, you do not know which one sort will use. This causes no difficulty, however. The amount of intermediate storage you assign may affect the sort/merge program's choice of a technique. In other words, you may implicitly rule out one technique by not providing enough intermediate storage for its use. To avoid this possibility, calculate the intermediate storage required by all the techniques and provide the largest amount needed.

Tape Intermediate Storage

If you use tape for intermediate storage, the following formulas give the number of tapes needed to complete a tape sort for a given data set size and sequence distribution technique:

Formula 1 $n = 2(x+1)$ -- balanced tape technique -- maximum input is 15 reels.

Formula 2 $n = x+2$ -- oscillating tape technique -- maximum input is 15 reels.

Formula 3 $n = 3$ reels -- polyphase tape technique -- maximum input is 1 reel.

The x represents the minimum number of volumes required to contain the input data set with a blocking factor equal to that used for intermediate storage by the sort/merge program. If input is spanned, or uses a more efficient blocksize, you will need more space. For an approximate sort blocking figure refer to the publication OS Sort/Merge Timing Estimates under your particular configuration and record length.

The maximum number of tape units that can be used for intermediate storage are:

32 for the balanced technique.

17 for the oscillating technique.

17 for the polyphase technique.

These maximums permit the sorting of 15 reels of input with the balanced and oscillating techniques. The polyphase technique allows only one reel of input.

2311, 2301, and 2314 (Balanced Technique) Intermediate Storage

Use the following formula to calculate the approximate number of tracks (T) required to complete a direct access sort for a given data set size when intermediate storage is on 2311 or 2314 disk or 2301 drum. If the data set tends to be ordered in reverse of the sequence you want the output to be in, more intermediate storage may be necessary. Conversely, if the input data set tends to be ordered in the desired sequence, less intermediate storage is necessary.

Formula 4
$$T = \frac{S(N)}{k(N-1)} + 2N$$

where:

N   is the number of intermediate storage areas. You must have at least three, but no more than six.

S   is the number of records in the input data set, exact or approximate.

$$k = \frac{B}{L}$$

where:

B is   3,400 for the 2311
        18,000 for the 2301
        7,000 for the 2314

L is the length in bytes of each record in the input data set. For variable-length records, L is the maximum length.

Only the integer portion of k is used for calculating T. Disregard the remainder, whatever its value. If the formula yields k = 0, use the value 1.

You must make at least three intermediate storage areas available to the sort and define each as a separate data set. Assign at least three tracks to the smallest area (five for the 2314). All tracks in an area must be contiguous. You can use up to six areas. Divide the number of tracks (T) among the areas you select. The formula is based on areas of equal size. More tracks will be needed if T is not divided equally.

Intermediate Storage Assignment Example

Determine T for 2301 using 4 intermediate storage data sets, variable-length records; maximum length 120, estimated input data set size 25500 records.

$$T = \frac{25500(4)}{\frac{18000(3)}{120}} + 8 = \frac{102000}{450} + 8 \approx 235$$

Divide T among the 4 data sets: 59, 59, 59, 58.

If the sort/merge program has less than 44K bytes of main storage to execute in, you may have to increase the value of T. If sort/merge has 12K bytes of main storage, you should increase T by about 50%. If main storage is between 12K and 44K, the percentage of increase is correspondingly less.

For information on assigning intermediate storage for efficient program operation, refer to "Section 4: Efficient Program Use".

46

2314 (Crisscross Technique) Intermediate Storage

Use the following formula to calculate the approximate total number of tracks (T) required to complete a sort when intermediate storage is on a 2314 and the crisscross sequence distribution technique is used:

Formula 5 $$T = \frac{1.25S}{k}$$

where:

S   is the number of records in the input data set, either actual or approximate.

$$k = \frac{B}{L}$$

where:

B   is 7,000

L   is the number of bytes in each record in the input data set.

   For variable-length records, L is the maximum record length. Use only the integer portion of k. Disregard the remainder, whatever its value. If the formula yields k=0, use the value 1.

When the input data set is on 2314, and you know how much space it occupies, you do not need to use the above formula to determine intermediate storage space. Assign intermediate storage space that is at least 25% larger than the space occupied by the input data set.

If the data set tends to be ordered in reverse of the desired output sequence, more intermediate storage space is necessary. Conversely, if the data set tends to be ordered in the desired sequence, less space is required. Also, if the sort/merge program is assigned less than 44K bytes of main storage in which to execute, you may have to increase the value of T. If sort/merge has 24K bytes of main storage, you should increase T by about 50%. If main storage is between 24K and 44K, the percentage of increase is correspondingly less.

The sort/merge program requires a minimum of six 2314 areas when the crisscross technique is used and permits a maximum of 17. (When the balanced technique is used, the minimum number of 2314 areas is three.) Each area must contain at least five tracks. All tracks in an area must be contiguous.

Efficient assignment of 2314 space is discussed in "Section 4: Efficient Program Use".

Intermediate Storage Assignment Formulas—Summary

Device Types for Intermediate Storage

INPUT	INTERMEDIATE STORAGE
7-track tape	7- and/or 9-track tape or 2311 disk or 2301 drum or 2314 facility
Any device but 7-track tape	9-track tape or 2311 disk or 2301 drum or 2314 facility

NUMBER OF TAPES REQUIRED FOR INTERMEDIATE STORAGE (N)

Formula 1 $n = 2(x+1)$ -- for the balanced technique, maximum n=32, maximum input 15 reels.

Formula 2 $n = x+2$ -- for the oscillating technique, maximum n=17, maximum input 15 reels.

Formula 3 $n = 3$ -- for the polyphase technique, maximum n=17, maximum input 1 reel.

where:

x   is the number of tapes that would be required to contain the input data set at sort blocking.

TOTAL NUMBER OF TRACKS REQUIRED FOR DIRECT ACCESS INTERMEDIATE STORAGE

Formula for 2301, 2311 and 2314 with balanced technique

Formula 4 $T = \dfrac{S(N)}{k(N-1)} + 2N$

Formula for 2314 with crisscross technique

Formula 5 $T = \dfrac{1.25S}{k}$

where:

N   is the number of intermediate storage areas
$3 \leq N \leq 6$ for 2311, 2301 and 2314 with the balanced technique
$6 \leq N \leq 17$ for 2314 with crisscross technique

S   is the number of input records

$k = \dfrac{B}{L}$

     B   is 3,400 for the 2311
           18,000 for the 2301
           7,000 for the 2314
     L   is the input record length (maximum length for variable-length records)

Note: Use only the integer portion of k. Never round upwards. If k = 0, use 1.

Job Control Language for Sort/Merge

When the sort/merge program is initiated via the system input stream, it requires a JOB statement, an EXEC statement, and DD statements.

JOB STATEMENT

The JOB statement for a sort/merge job is a standard System/360 Operating System JOB statement.

```
//jobname   JOB   accounting info,programmer name, etc.
```

EXEC STATEMENT

The EXEC statement identifies either a sort/merge cataloged procedure or the sort/merge program. The statement

```
//stepname   EXEC  { PGM=SORT     }  [ ,PARM='optional parameters' ]
                   { PGM=IERRCO00 }  [        DISCUSSED LATER       ]
```

identifies the sort/merge program. The statement

```
//stepname   EXEC  { PROC=SORT  }   [ ,PARM='optional parameters' ]
                   { PROC=SORTD }   [        discussed later       ]
                   { SORT       }
                   { SORTD      }
```

identifies a sort/merge cataloged procedure. The procedures, SORT and SORTD are shown later in this section under "Initiating Sort/Merge". The PROC= notation merely serves as a reminder that a cataloged procedure is being used.

PARM Field Options

```
          ' [ [ BALN ] ] [  xxxxxx (an        ] [    [ NO ] ]
PARM=       [ [ OSCL ] ] [ ,CORE=optional main ] [ ,MSG={ CC } ]  [ ,DIAG ]
            [ [ POLY ] ] [   storage value)    ] [    { CP } ]
            [ [ CRCX ] ]                          [    { AC } ]
                                                  [    [ AP ] ]
```

The first PARM field option specifies a sequence distribution technique to be used by the sort/merge program. If the intermediate storage medium is tape, BALN means use the balanced tape technique, OSCL means use the oscillating tape technique, and POLY means use the polyphase tape technique. If the intermediate storage medium is on a 2314 storage facility, BALN means use the balanced direct access technique, CRCX means use the crisscross direct access technique.

Note: You cannot choose a sequence distribution technique if intermediate storage is on 2311 or 2301; sort/merge always uses the balanced technique. There are certain restrictions on your choice of a technique for the 2314:

- If less than six work areas are provided, the sort/merge program always uses the balanced technique.

- If more than six work areas are provided, the program uses the crisscross technique.

- If exactly six work areas are provided, the program uses the balanced technique unless CRCX is specified in the PARM field.

You should be extremely cautious when forcing the sort/merge program to use a specific technique. The program tries to select the most efficient technique for a given application. If it is forced to use another, performance may not be as efficient. Refer to Figure 5 in Section 1 for information about the requirements of the sequence distribution techniques.

The second PARM field option is an optional main storage value which will temporarily override the sort/merge storage allocation set up at system generation time. Refer to "Altering the Main Storage Allocation" in Section 4.

You can use the third PARM field option to override temporarily the message option specified at system generation time. The option is requested by MSG=xx. Valid entries for xx are:

- NO - no messages are printed.

- CC - critical messages only are printed. They appear on the system console.

- CP - critical messages only are printed. They appear on the printer.

- AC - all messages are printed. They appear on the system console.

- AP - all messages are printed. They appear on the printer.

Note: When using ASP you cannot specify MSG=CP or MSG=AP.

The DIAG PARM field option is only for use in customer problem determination. It should never be used for normal sort or merge jobs, as it degrades sort/merge performance. It causes the program to print diagnostic messages and control cards. If the program terminates in the sort or merge phases with a critical message, use of this parameter also produces an 0C1 abnormal termination dump for diagnostic use.

DD STATEMENTS

If you do not use a sort/merge cataloged procedure to invoke the sort/merge program, you must include system DD statements in the input stream. These are the DD statements that would be contained in the cataloged procedure. They are:

| //SYSPRINT | DD | used by the linkage editor. Include this statement when your routines that require linkage editing are included in the application. |

//SYSLMOD   DD   defines a data set that contains output from the linkage editor. Include this statement when your routines that need linkage editing are included in the application.

//SYSUT1   DD   defines a data set used as a work area by the linkage editor. Use this statement when your routines that must be linkage edited are included.

//SYSLIN   DD   defines a data set created by the program, for input to the linkage editor. Use this statement when your routines that require linkage editing are included.

//SORTLIB   DD   defines a data set that contains load modules for the sort/merge program. Always include this statement.

//SYSOUT   DD   defines a data set used as the system output data set. Always use this statement.

The following DD statements are required whether sort is initiated directly or through a cataloged procedure:

```
//SORTIN       DD    defines the input data set for a sorting application.  Not
                     required for a merge-only application.

//SORTIN01     DD    define the input data sets for a merging application.
----------           Not required for a sorting application.

----------
//SORTIN16     DD

//SORTWK01     DD    define intermediate storage data sets for a sorting
----------           application.  Not required for a merging application.

----------           Refer to the section "Determining Intermediate Storage
//SORTWK32     DD    Requirements" for information on how many SORTWKxx DD
                     statements are needed.

//SORTOUT      DD    defines the output data set for sorting and merging
                     applications.

//SORTMODS     DD    defines a temporary partitioned data set large enough to
                     contain all of your modification routines that appear in the
                     input stream for a given application.  If your routines are
                     not in the input stream, this statement is not required.  If
                     your routines are on libraries, DD statements defining the
                     libraries must be included.

//SORTCKPT     DD    defines a data set for checkpoint records.  If you are not
                     using the checkpoint facility this statement is not required.
```

REQUIRED DD STATEMENT PARAMETERS

The sort/merge program requires that certain parameters be included in the DD statements described above. These parameters, the conditions under which they are required, a summary of the information contained in them, and the value assumed (default) if the parameter is not included are shown in Figure 12. The parameters and subparameters which are not required are not discussed.

PARAMETER	CONDITION UNDER WHICH REQUIRED	SUMMARY OF PARAMETER VALUE	DEFAULT VALUE
DSNAME	When the DD statement defines a labeled input data set (e.g., SORTIN), or when the data set being created is to be kept or cataloged (e.g., SORTOUT), or passed to another step.	Specifies the fully qualified name or the temporary name of the data set.	The system assigns a unique name.
DCB	Always required for the input and output data sets; and when 7-track tape is used for intermediate storage.[1]	Specifies information used to fill the data control block (DCB) associated with the data set.	----
UNIT	When the input data set is neither cataloged nor passed or when the data set is being created.	Specifies (symbolically or actually) the type and quantity of I/O units required by the data set.	----
SPACE	When the DD statement defines a new direct access data set.	Specifies the amount of space needed to contain the data set.	----
VOLUME	When the input data set is neither cataloged nor passed, for multi-reel input, or when the output data set is on direct access and is to be kept or cataloged.	Specifies information used to identify the volume or volumes occupied by the data set.	----
LABEL	When the default value is not applicable.	Specifies information about labeling and retention for the data set.	The system assumes standard labeling.
DISP	When the default value is not applicable.	Indicates the status and disposition of the data set.	The system assumes (NEW, DELETE)
[1]Input, on direct access devices only, is an exception.			

Figure 12. Summary of DD Statement Parameters Required by the Sort/Merge Program

A full description of other DD statement parameters and subparameters is contained in the publication <u>OS JCL Reference</u>.

Figure 13 is a summary of the DCB subparameters that are required by the sort/merge program if the DCB parameter is used. A more detailed discussion of these and other DCB subparameters is contained in the publication OS Data Management Macro Instructions.

SUBPARAMETER	CONDITION UNDER WHICH REQUIRED	SUMMARY OF SUBPARAMETER VALUE	DEFAULT VALUE
DEN	When the data set is located on a 7-track 2400-series tape unit.	Specifies the density at which the tape was recorded.	800 bpi
TRTCH	When the data set is located on a 7-track 2400-series tape unit.	Specifies the technique used to record 8-bit bytes on a 7-track tape.	Converter not used, translator not used, odd parity.
RECFM	When the DCB parameter is required, except on SORTWK statements.	Specifies the format of the records in the data set.	----
LRECL	When the DCB parameter is required, except on SORTWK statements. Not required for fixed-length unblocked records	Specifies the maximum length (in bytes) of the logical records in the data set.[1]	----
BLKSIZE	When the DCB parameter is required, except on SORTWK statements.	Specifies the maximum length (in bytes) of the physical records in the data set.	----

[1]When used in the SORTOUT DD statement to shorten the output records, care must be taken that the shortened records still include all the control fields.

Figure 13. Summary of DCB Subparameters Required by the Sort/Merge Program

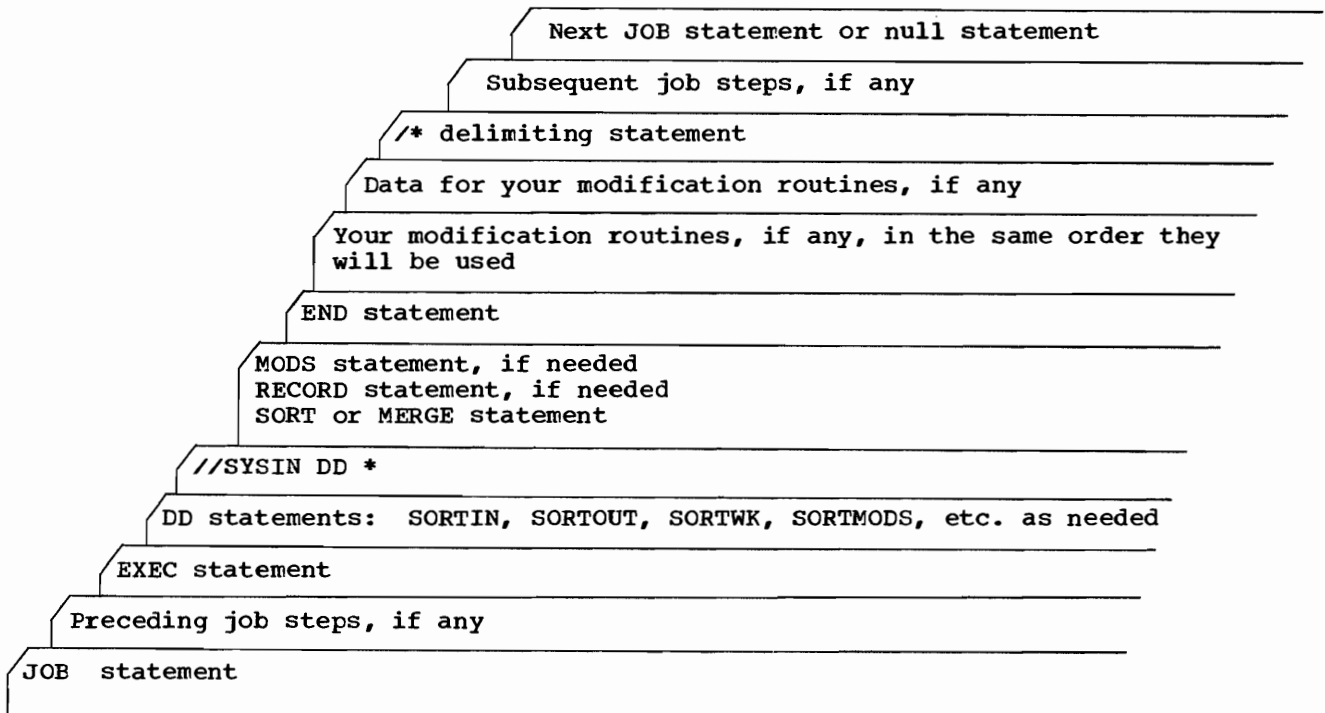Figure 14 illustrates the order in which control statements must be placed in the input stream.

```
              ┌─────────────────────────────────────────────────────────
             ╱  Next JOB statement or null statement
            ┌────────────────────────────────────────────────────────────
           ╱  Subsequent job steps, if any
          ┌─────────────────────────────────────────────────────────────
         ╱ /* delimiting statement
        ┌───────────────────────────────────────────────────────────────
       ╱  Data for your modification routines, if any
      ┌────────────────────────────────────────────────────────────────
     ╱  Your modification routines, if any, in the same order they
    │   will be used
   ┌──────────────────────────────────────────────────────────────────
  ╱  END statement
 ┌───────────────────────────────────────────────────────────────────
╱  MODS statement, if needed
│  RECORD statement, if needed
│  SORT or MERGE statement
┌────────────────────────────────────────────────────────────────────
╱//SYSIN DD *
┌─────────────────────────────────────────────────────────────────────
╱ DD statements:  SORTIN, SORTOUT, SORTWK, SORTMODS, etc. as needed
┌──────────────────────────────────────────────────────────────────────
╱ EXEC statement
┌───────────────────────────────────────────────────────────────────────
╱ Preceding job steps, if any
┌────────────────────────────────────────────────────────────────────────
╱JOB   statement
│
```

Figure 14.   Arrangement of Statements for Sort/Merge Execution

    Each of the DD statement types required by the sort/merge program are discussed
in the following text. Examples of the statements are included.

SORTIN DD Statement

For a sort, the SORTIN data set may be cataloged or uncataloged, or it may be
inserted by your routine at exit E15 (see "Section 3: Program Modification").
The SORTIN data set may not be a DD DUMMY.

DD Example 1: SORTIN DD Statement

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                                │
│    This example shows DD statement parameters that define a previously cataloged│
│ input data set:                                                                │
│                                                                                │
│ //SORTIN      DD  DSNAME=INPUT,DISP=(OLD,DELETE),                    X          │
│ //                DCB=(RECFM=FB,BLKSIZE=800,LRECL=80)                           │
│                                                                                │
└──────────────────────────────────────────────────────────────────────────────┘
```

DSNAME   causes the system to search the catalog for a data set with the
                name INPUT. When the data set is found, it is associated with the
                ddname SORTIN. The control program obtains the unit assignment and
                volume serial number from the catalog and types a mounting message
                to the operator if the volume is not already mounted.

DISP   indicates that the data set is passed or cataloged (OLD) and that
                it should be deleted (DELETE) after the current job step.

DCB   indicates that the data set contains fixed-length blocked records
                (RECFM=FB) with a block size of 800 bytes and a record length of 80
                bytes.

If the input data set is contained on more than one reel of magnetic tape, the
VOLUME parameter must be included on the SORTIN DD statement to indicate the
serial numbers of the tape reels. In the following volume parameter example, the
input data set is on three reels that have serial numbers 75836, 79661, and 72945.

DD Example 2: Volume Parameter on SORTIN DD

```
VOLUME=SER= (75836,79661,72945)
```

When input to the sort/merge program is a concatenated data set, all data sets
in the concatenation must have identical attributes. If they do not, results are
unpredictable. This causes sort to terminate if an actual data set size appears
in the SIZE parameter of the SORT control card because of the ensuing record count
off condition.

SORTIN01 -- SORTIN16 DD Statements

These DD statements define the input data sets for a merge operation. They must
be numbered in ascending sequence. SORTIN01 is the name of the first DD
statement; SORTIN02 is the name of the second DD statement, etc. No numbers can
be skipped. The maximum block size and the maximum record length of all the data
sets to be merged must be defined in the SORTIN01 DD statement. RECFM and LRECL
must be the same for all input data sets. Mixtures of fixed- and variable-length
records are not allowed. Fixed-length records must all be of the same length.

DD Example 3: SORTIN01 -- SORTIN03 DD Statements for a Merge

```
//SORTIN01    DD   DSNAME=MERGE1,VOLUME=SER=000111,DISP=OLD,      X
//                 LABEL= (,NL) ,UNIT=2400,                       X
//                 DCB= (RECFM=FB,LRECL=80,BLKSIZE=240)
//SORTIN02    DD   DSNAME=MERGE2,VOLUME=SER=000121,DISP=OLD       X
//                 LABEL= (,NL) ,UNIT=2400,                       X
//                 DCB= (RECFM=FB,LRECL=80,BLKSIZE=240)
//SORTIN03    DD   DSNAME=MERGE3,VOLUME=SER=000131,DISP=OLD,      X
//                 LABEL= (,NL) ,UNIT=2400,                       X
//                 DCB= (RECFM=FB,LRECL=80,BLKSIZE=240)
```

DD Example 4: SORTIN01 and SORTIN02 DD Statements for a Merge

```
//SORTIN01    DD   DSNAME=INPUT1,VOLUME=SER=000101,               X
//                 UNIT=2301,DISP=OLD,DCB= (RECFM=VB,             X
//                 LRECL=240,BLKSIZE=2400)
//SORTIN02    DD   DSNAME=INPUT2,VOLUME=SER=000201,               X
//                 UNIT=2301,DISP=OLD,DCB= (RECFM=VB,             X
//                 LRECL=240,BLKSIZE=2400)
```

SORTWK01 -- SORTWK32 DD Statements

These statements define the intermediate storage data sets for a sort operation.
For a merge-only operation, these statements are not required. Intermediate
storage data sets can be on tape or direct access devices but not on a mixture of
both. Your selection of an intermediate storage device type is not related to the
device types used for input or output with one exception: seven-track tape cannot
be used for intermediate storage unless the input device is also 7-track tape.
Refer to "Intermediate Storage Space Requirements" in this section for information
about how much intermediate storage is required for a particular application.

If you are using the checkpoint/restart facility and may be making a deferred restart, you must make the following two additions to each of your SORTWK DD statements so that the sort work data sets will not be lost:

    DSNAME=anyname
    DISP= (NEW,DELETE,KEEP)

Thus a complete SORTWK DD statement for deferred restart might be:

```
//SORTWK01     DD   DSNAME=WORK1,UNIT=2311,SPACE(TRK,(20),,CONTIG),      X
//                  DISP= (NEW,DELETE,KEEP)
```

With this DD statement, the data set will be kept, if the job step aborts, and will be in the system until the step has been successfully rerun or until the data set has been deleted by some other means.

When the intermediate storage data sets are on direct access devices, only the primary space allocation is used by sort/merge and the space must be contiguous. Partitioned data sets cannot be used for SORTWK.

The ddnames for intermediate storage data sets must be numbered in ascending sequence. SORTWK01 must be the first, SORTWK02, the second, etc., and no numbers can be skipped.

DD Example 5: SORTWK01 DD Statement Defining a Tape Intermediate Storage Data Set

```
//SORTWK01     DD   UNIT=2400,LABEL= (,NL)
```

These parameters specify an unlabeled data set on a 2400 series tape unit. The system assigns a unique name to the data set because the DSNAME parameter is omitted. Because the DISP parameter is omitted, the system assumes DISP= (NEW,DELETE) ; the data set has not been previously cataloged and it will be deleted at the end of the current job step. The disposition PASS is not allowed for a SORTWK data set.

DD Example 6: SORTWK01 DD Statement Defining a Direct Access Data Set for Intermediate Storage

```
//SORTWK01     DD   UNIT=2311,SPACE= (TRK, (200) ,,CONTIG)
```

UNIT   specifies a 2311 disk. The LABEL parameter is omitted. The default is
          standard labels.

SPACE   specifies 200 contiguous tracks for the data set.

    The omission of the DSNAME parameter causes the system to assign a unique name to the data set. The DISP parameter is omitted; the system assumes NEW, DELETE.

SORTOUT DD Statement

This DD statement is used to define all the characteristics of the output data set.

DD Example 7:   SORTOUT DD Statement

```
|//SORTOUT       DD  DSNAME=OUTPT,UNIT=2400,DISP=(NEW,CATLG),          X          |
|//                  DCB=(RECFM=FB,LRECL=90,BLKSIZE=900)                           |
```

DSNAME   The data set is to be called OUTPT.

DISP   The data set is unknown to the operating system (NEW) and it is to be cataloged (CATLG) under the name OUTPT.

UNIT   indicates that the data set is on a 2400 series tape unit.

DCB   specifies a fixed-length blocked data set with a record length of 90 bytes and a block size of 900 bytes.

SORTMODS DD Statement

This statement is required if your routines are included in the system input stream. It must define a temporary partitioned data set large enough to hold all your routines that appear in the input stream. The sort/merge program transfers your routines to the SORTMODS data set before they are linkage edited for execution. If all your routines are located in libraries, the SORTMODS DD statement is not required, but DC statements defining the libraries must be included.

DD Example 8:   SORTMODS DD Statement Defining a SORTMODS Data Set on 2311

```
|//SORTMODS      DD  UNIT=2311,SPACE=(TRK,(10,,3))                                 |
```

These parameters allot ten tracks of a 2311 disk to the SORTMODS data set. Space for three directory blocks is also requested.

SORTCKPT DD Statement

The SORTCKPT data set may be assigned on any device that operates with BSAM. Processing can be restarted from the last checkpoint taken. If the MOD disposition is specified for the checkpoint data set, processing can be restarted from the checkpoint taken at the start of the sort phase as well as the last checkpoint taken.

DD Example 9:   SORTCKPT DD Statement

```
|//SORTCKPT      DD  DSNAME=CHECK,VOLUME=SER=000123,DISP=(NEW,KEEP),   X          |
|//                  UNIT=2400                                                     |
```

Job Control Language Statements for Sort/Merge—Summary

Statement		Purpose	When Required
//jobname	Job	Introduces the job.	Always.
//stepname	EXEC	Introduces the step.	Always.
//SYSPRINT	DD	Used by linkage editor.	When you do not use a cataloged procedure and have modification routines that require link editing.
//SYSLMOD	DD	Defines linkage editor output data set.	Same as for SYSPRINT.
//SYSUT1	DD	Defines work area for linkage editor.	Same as for SYSPRINT.
//SYSLIN	DD	Defines input data set for linkage editor.	Same as for SYSPRINT.
//SORTLIB	DD	Defines data set that contains sort/merge program modules.	When you do not use cataloged procedures SORT or SORTD.
//SYSOUT	DD	Defines system output data set.	Same as SORTLIB.
//SORTIN	DD	Defines input data set for a sort.	For a sort, always unless LINK, ATTACH, or XCTL is used to invoke sort and the input data set is inserted by your routine at sort/merge exit E15. Not used for a merge.
//SORTIN01-16	DD	Define input data sets for a merge.	For a merge, always. Not used for a sort.
//SORTWK01-32	DD	Define intermediate storage data sets for a sort.	For a sort, always. Not used for a merge.
//SORTOUT	DD	Defines sort/merge output data set.	Always, unless LINK, ATTACH, or XCTL is used to invoke sort and your routine disposes of output via sort/merge exit E35.
//SORTMODS	DD	Defines a temporary data set for your modification routines in SYSIN.	When you supply modification routines through the system input stream.
//SORTCKPT	DD	Defines data set for checkpoint records.	When you use the checkpoint facility.
//SYSIN	DD *	Indicates that data set containing sort/merge control statements follows in input stream.	Always.
/*		Marks the end of SYSIN data set.	Always.
Shaded statements are provided by SORT or SORTD cataloged procedure.			

JCL and Sort/Merge Statement Examples

Following are a number of examples showing all the JCL and sort/merge statements necessary to accomplish a particular job. The sort/merge control statements shown have the same operands as those illustrated and explained at the end of the topic "Defining the Sort or Merge" in this section.

Example 1 -- Sort

Input	Output	Intermediate Storage	User Modifications	Options
Blocked fixed-length records on 9-track tape	Blocked fixed-length records on 9-track tape	Four 9-track tapes	None	FORMAT=xx for control fields of like format. Estimated data set size.

```
//EXAMP1          JOB A402,PROGRAMMER                                        01
//STEP1           EXEC SORTD                                                 02
//SORT.SORTIN      DD  DSNAME=INPUT,VOLUME=SER=000101,                       03
//                     UNIT=2400,DISP= (OLD,DELETE) ,                        04
//                     DCB= (RECFM=FB,LRECL=80,                              05
//                     BLKSIZE=800)                                          06
//SORT.SORTOUT     DD  DSNAME=OUTPUT,UNIT=2400,DISP= (NEW,                   07
//                     CATLG) ,VOLUME=SER=000102,DCB= (RECFM=FB,             08
//                     LRECL=80,BLKSIZE=800)                                 09
//SORT.SORTWK01 DD  UNIT=2400                                                10
//SORT.SORTWK02 DD  UNIT=2400                                                11
//SORT.SORTWK03 DD  UNIT=2400                                                12
//SORT.SORTWK04 DD  UNIT=2400                                                13
//SORT.SYSIN    DD  *                                                        14
  SORT   FIELDS= (1.0,6.0,A,28,5,D) ,FORMAT=CH,SIZE=E10000                   15
  END                                                                        16
/*                                                                           17
```

01   The JOB statement introduces this job to the operating system. The card contains accounting information and programmer identification. Message level 0, indicating that only incorrect control statements and associated diagnostic messages are to be printed, is specified by default.

02   The EXEC statement invokes the cataloged procedure SORTD. It can be written as shown or as EXEC PROC=SORTD. The contents of the two cataloged procedures supplied by IBM for sort/merge are shown in Section 2. The SORT cataloged procedure could be used for this example, but it causes allocation of linkage editor data sets which are not needed since no user-written modification routines that require linkage editing are included. The SORT procedure is therefore less efficient than the SORTD procedure for this example.

The remaining DD statements are being added to the SORTD procedure for this job step only. Therefore they are qualified by the stepname (SORT) of the SORTD procedure. The SORT procedure also has the stepname SORT.

03-06 The SORTIN DD statement describes an input data set named INPUT. The data set is on a 9-track tape that has the serial number 000101. The DISP parameter indicates that the data set is known to the operating system and that it should be deleted from the system after this job step. The DCB parameter shows that the data set consists of fixed-length records with a record size of 80 and a block size of 800.

07-09   The SORTOUT DD statement describes the output data set. OUTPUT will be
         recorded on a 9-track tape drive and will be cataloged after it is
         created. The data set will be placed on tape volume number 102.
         OUTPUT's format, record length and block size are the same as those for
         SORTIN.

10-13   These DD statements define temporary intermediate storage data sets. The
         three data sets are on 9-track tape drives. No other parameters are
         necessary since the standard system default options are acceptable for
         this application.

14   The SYSIN DD * statement informs the operating system that a data set
         follows in the input stream.

15-16   Sort/Merge control statements described in Example 1 at the end of the
         topic "Defining the Sort or Merge".

17   The /* delimiter statement marks the end of the SYSIN data set.

60

Example 2 -- Sort

```
|Example 2 is a sorting application exactly like that shown in Example 1 except |
|that a cataloged procedure is not used.  The sort/merge program is called      |
|directly.  Only the EXEC statement, which is different from Example 1, and the |
|two extra DD statements are described.  Note that the DD statements need not be |
|qualified by the word SORT.                                                    |
|-------------------------------------------------------------------------------|
|//EXAMP2      JOB  A402,PROGRAMMER                                             |
|//STEP1       EXEC PGM=IERRCO00,REGION=26K                              01     |
|//SYSOUT      DD   SYSOUT=A                                             02     |
|//SORTLIB     DD   DSNAME=SYS1.SORTLIB,DISP=SHR                         03     |
|//SORTIN      DD   DSNAME=INPUT,VOLUME=SER=000101,                            |
|//                 UNIT=2400,DISP=(OLD,DELETE),                               |
|//                 DCB=(RECFM=FB,LRECL=80,                                    |
|//                 BLKSIZE=800)                                               |
|//SORTOUT     DD   DSNAME=OUTPUT,UNIT=2400,DISP=(NEW,                         |
|//                 CATLG),VOLUME=SER=000102,DCB=(RECFM=FB,                    |
|//                 LRECL=80,BLKSIZE=800)                                      |
|//SORTWK01    DD   UNIT=2400                                                  |
|//SORTWK02    DD   UNIT=2400                                                  |
|//SORTWK03    DD   UNIT=2400                                                  |
|//SORTWK04    DD   UNIT=2400                                                  |
|//SYSIN       DD   *                                                          |
|  SORT  FIELDS=(1.0,6.0,A,28,5,D),FORMAT=CH,SIZE=E10000                       |
|  END                                                                        |
|/*                                                                           |
```

01   This EXEC statement initiates the sort/merge program and indicates that
        it needs a 26K region in which to operate.

02   This DD statement directs the system output to system output class A.

03   This DD statement defines the data set containing the sort/merge program
        modules.

Example 3 -- Merge

Input	Output	Intermediate Storage	User Modifications	Options
Blocked fixed-length records on four 9-track unlabeled tapes	Blocked fixed-length records on one 9-track tape	None is required for a merge	None	FORMAT=xx for control fields of like format. Estimated data set size.

```
//EXAMP3       JOB A402,PROGRAMMER                                01
//STEP1        EXEC SORTD                                         02
//SORT.SORTIN01  DD  DSNAME=MERGIN01,VOLUME=SER=000111,           03
//               DISP=OLD,LABEL=(,NL),UNIT=2400,                  04
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=240)              05
//SORT.SORTIN02  DD  DSNAME=MERGIN02,VOLUME=SER=000222,           06
//               DISP=OLD,LABEL=(,NL),UNIT=2400,                  07
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=240)              08
//SORT.SORTIN03  DD  DSNAME=MERGIN03,VOLUME=SER=000333,           09
//               DISP=OLD,LABEL=(,NL),UNIT=2400,                  10
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=240)              11
//SORT.SORTIN04  DD  DSNAME=MERGIN04,VOLUME=SER=000444,           12
//               DISP=OLD,LABEL=(,NL),UNIT=2400,                  13
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=240)              14
//SORT.SORTOUT   DD  DSNAME=MERGOUT,VOLUME=SER=000101,            15
//               DISP=(NEW,KEEP),LABEL=(,NL),UNIT=2400,           16
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=240)              17
//SORT.SYSIN     DD  *                                            18
 MERGE  FIELDS=(1.0,6.0,A,28,5,D),FORMAT=CH,SIZE=E10000           19
 END                                                              20
/*                                                                21
```

01-02   The basic JOB and EXEC statements. The EXEC statement invokes the cataloged procedure SORTD.

03-14   These DD statements describe the merge input data sets. They are all on 9-track unlabeled tape and consist of fixed-length records with a blocking factor of three. The total number of records on all of the data sets is about 10,000 as indicated by the SIZE parameter on the MERGE statement.

15-17   The result of the merge is recorded on 9-track tape at the same blocking factor and in the same format as the input data sets.

18   A data set follows in the input stream.

19-20   Sort/Merge control statements described in Example 2 at the end of the topic "Defining the Sort or Merge".

21   Marks the end of the SYSIN data set.

Example 4 -- Sort

Input	Output	Intermediate Storage	User Modifications	Options
Fixed-length blocked records on 9-track tape	Fixed-length blocked records on 9-track tape, same unit as input data set	Three 2311 areas of 540 tracks each	Four - two change record lengths, one changes control fields, one decides what to do if Nmax is exceeded	Estimated data set size

```
//EXAMP4          JOB  A402,PROGRAMMER                                    01
//STEP1           EXEC SORT                                               02
//SORT.SORTIN     DD   UNIT=2400,DSNAME=INPUT,VOLUME=SER=000101,          03
//                DCB= (RECFM=FB,LRECL=120,                               04
//                BLKSIZE=480) ,DISP= (OLD,DELETE)                        05
//SORT.SORTOUT    DD   UNIT=AFF=SORTIN,DSNAME=OUTPUT,                     06
//                VOLUME=SER=000101,DCB= (RECFM=FB,                       07
//                LRECL=80,BLKSIZE=320) ,DISP= (NEW,PASS)                 08
//SORT.SORTWK01   DD   UNIT=2311,SPACE= (TRK, (540) ,,CONTIG)            09
//SORT.SORTWK02   DD   UNIT=2311,SPACE= (TRK, (540) ,,CONTIG)            10
//SORT.SORTWK03   DD   UNIT=2311,SPACE= (TRK, (540) ,,CONTIG)            11
//SORT.MODLIB     DD   DSNAME=YOURRTNS,DISP=OLD                           12
//SORT.SORTMODS   DD   UNIT=2311,SPACE= (TRK, (10,,3) )                   13
//SORT.SYSIN      DD   *                                                  14
   SORT  FIELDS= (3.0,8.0,ZD,E,40.0,6.0,CH,D) ,SIZE=E30000               15
   RECORD  TYPE=F,LENGTH= (120,100,80)                                    16
   MODS   E15= (E15,780,MODLIB,N) ,E16= (E16,1024,MODLIB) ,              17
          E35= (ADDUP,912,SYSIN) ,E61= (CHGE,1000,SYSIN)                 18
   END                                                                    19
Object deck for ADDUP routine
Object deck for CHGE routine
/*                                                                        20
```

01-02   The basic JOB and EXEC statements. The EXEC statement specifies the SORT cataloged procedure because user-written routines that require linkage editing are included in the application.

03-05   This DD statement describes an input data set that consists of fixed-length blocked records on 9-track tape. Each record is 120 bytes long and the blocking factor is 4. The data set, which is already known to the operating system, will be deleted after this job step.

06-08   This DD statement describes the output data set. UNIT=AFF=SORTIN means that the data set is to be placed on the same unit as the input data set. The output records have the same format as the input records, but they are each 40 bytes shorter. The blocking factor is the same.

09-11   The next three DD statements describe three intermediate storage areas on 2311 disk. Each area contains 540 contiguous tracks.

12   Defines the data set that contains the E15 and E16 modification routines.

13   Defines a data set on which the ADDUP and CHGE routines will be placed.

14   A data set follows in the input stream.

15-19   Sort/Merge control statements described in Example 3 at the end of the topic "Defining the Sort or Merge".

Object decks for your modification routines must appear in the input stream in numerical exit number order. ADDUP is the routine for exit E35, so it appears first. CHGE, the routine used at exit E61, appears second.

20   Marks the end of the SYSIN data set.

Example 5 -- Sort

Input	Output	Intermediate Storage	User Modifications	Options
Fixed-length blocked records on two 9-track tape volumes	Fixed-length blocked records on one 9-track tape	Four 9-track tapes	Four - two change record lengths, one changes control fields, one decides what to do if Nmax is exceeded.	Estimated data set size, oscillating technique forced.

```
//EXAMP5      JOB   A402,PROGRAMMER                                              01
//STEP1       EXEC  SORT,PARM='OSCL'                                            02
//SORT.SORTIN DD    DSNAME=INPUT,VOLUME=SER= (000333,000343) ,                 03
//                  UNIT=2400,DISP= (OLD,DELETE) ,                             04
//                  DCB= (RECFM=FB,LRECL=120,                                   05
//                  BLKSIZE=480)                                                06
//SORT.SORTOUT DD   DSNAME=OUTPUT,UNIT=2400,DISP= (NEW,                         07
//                  CATLG) ,VOLUME=SER=456,DCB= (RECFM=FB,                      08
//                  LRECL=80,BLKSIZE=320)                                       09
//SORT.SORTWK01  DD  UNIT=2400                                                  10
//SORT.SORTWK02  DD  UNIT=2400                                                  11
//SORT.SORTWK03  DD  UNIT=2400                                                  12
//SORT.SORTWK04  DD  UNIT=2400                                                  13
//SORT.MODLIB DD     DSNAME=YOURRTNS,DISP=OLD                                   14
//SORT.SORTMODS  DD  UNIT=2311,SPACE= (TRK, (10,,3) )                           15
//SORT.SYSIN   DD    *                                                          16
  SORT FIELDS= (3.0,8.0,ZD,E,40.0,6.0,CH,D) ,SIZE=E30000                        17
  RECORD TYPE=F,LENGTH= (120,100,80)                                            18
  MODS E15= (E15,780,MODLIB,N) ,E16= (E16,1024,MODLIB,N) ,            X         19
           E35= (ADDUP,912,SYSIN) ,E61= (CHGE,1000,SYSIN)                       20
  END
Object deck for ADDUP routine                                                  22
Object deck for CHGE routine                                                   23
/*                                                                             24
```

01   The basic JOB statement.

02   The EXEC statement specifies the cataloged procedure SORT. OSCL in the
      PARM field directs the sort/merge program to use the oscillating tape
      sequence distribution technique if it possibly can, whether or not it
      considers the oscillating technique most efficient for this application.

03-06 Defines the input data set. Note that the SORTIN DD statement is
      prefaced by the step name of the SORT cataloged procedure because it and
      other DD statements so prefaced are being added to the procedure for this
      job step. The input data set consists of fixed-length blocked records on
      two 9-track tape volumes numbered 000333 and 000343, respectively.

07-09 Defines the output data set. The output data set also consists of
      fixed-length blocked records. It is on one 9-track tape.

10-13 Defines four intermediate storage data sets on 9-track tape. Since the
      DSNAME parameter is omitted, the system will assign unique names to the
      data sets.

14   Describes a data set that contains the E15 and E16 modification routines.

15   Defines a data set on which the ADDUP and CHGE routines will be placed.

16   A data set follows in the input stream.

17-21   Sort/merge control statements described in Example 3 at the end of the topic "Defining the Sort or Merge".

22   The object deck for the ADDUP routine comes before the deck for CHGE.

23   The object deck for the CHGE routine.

24   SYSIN data set delimiter.

Example 6 -- Sort

Input	Output	Intermediate Storage	User Modifications	Options
Blocked fixed-length records on 7-track unlabeled tape	Blocked fixed-length records on 7-track labeled tape	Six 7-track tapes	None	FORMAT=xx for control fields of like format, estimated data set size.

```
//EXAMP6         JOB A402,PROGRAMMER                                    01
//STEP1          EXEC SORT                                              02
//SORT.SORTIN    DD  DSNAME=INPUT,VOLUME=SER=000101,                    03
//               UNIT=2400-2,DCB=(DEN=2,RECFM=FB,                       04
//               LRECL=80,BLKSIZE=800,TRTCH=ET) ,                       05
//               DISP=(OLD,PASS) ,LABEL=(,NL)                           06
//SORT.SORTOUT   DD  DSNAME=OUTPUT,UNIT=2400-2,DISP=(NEW,               07
//               CATLG) ,VOLUME=SER=102,DCB=(RECFM=FB,                  08
//               LRECL=80,BLKSIZE=800,DEN=2,TRTCH=ET)                   09
//SORT.SORTWK01  DD  UNIT=2400-2,LABEL=(,NL) ,DCB=(DEN=2,               10
//               TRTCH=ET)                                              11
//SORT.SORTWK02  DD  UNIT=2400-2,LABEL=(,NL) ,DCB=(DEN=2,               12
//               TRTCH=ET)                                              13
//SORT.SORTWK03  DD  UNIT=2400-2,LABEL=(,NL) ,DCB=(DEN=2,               14
//               TRTCH=ET)                                              15
//SORT.SORTWK04  DD  UNIT=2400-2,LABEL=(,NL) ,DCB=(DEN=2,               16
//               TRTCH=ET)                                              17
//SORT.SORTWK05  DD  UNIT=2400-2,LABEL=(,NL) ,DCB=(DEN=2,               18
//               TRTCH=ET)                                              19
//SORT.SORTWK06  DD  UNIT=2400-2,LABEL=(,NL) ,DCB=(DEN=2,               20
//               TRTCH=ET)                                              21
//SORT.SYSIN     DD  *                                                  22
  SORT  FIELDS=(1.0,6.0,A,28,5,D) ,FORMAT=CH,SIZE=E10000               23
  END                                                                  24
/*                                                                     25
```

01-02   Standard JOB and EXEC statements. The EXEC statement invokes the SORT
         cataloged procedure. The SORTD procedure would be more efficient for
         this application since there are no modification routines that need
         linkage editing, but the SORT procedure can be used.

03-06   The SORTIN DD statement defines the input data set. The data set is
         named INPUT, it is on an unlabeled 7-track tape with a serial number
         000101. The DCB subparameters indicate that the tape was recorded at
         800 bpi, is composed of fixed-length blocked records. The TRTCH=ET sub-
         parameter indicates that the tape was recorded with even parity and that
         BCDIC to EBCDIC translation is required. The DISP parameter shows that
         the data set is in existence and that it should be retained after this
         job step. The data set is the first one or only one of this unlabeled
         volume.

07-09   The SORTOUT DD statement defines the output data set. It is named
         OUTPUT, and is recorded on 7-track tape on a volume that has the serial
         number 102. The other parameters on this statement are the same as
         those on SORTIN, with the exception of DISP. DISP indicates that this
         data set will be created in this job step and will be cataloged for
         future reference by another job.

10-21   These DD statements define intermediate storage for the sort/merge
         program. The storage is on six 7-track unlabeled tapes. These tapes
         are to be recorded with even parity and BCDIC to EBCDIC translation.

22   A data set follows in the input stream.

23-24   Sort/Merge control statements described in Example 1 at the end of the
         topic "Defining the Sort or Merge".
25   Delimiter statement marks the end of the SYSIN data set.

Example 7 -- Sort

Input	Output	Intermediate Storage	User Modifications	Options
Fixed-length unblocked records on 2311 disk	Fixed-length blocked records on 2311 disk	Three 2311 areas, 120 tracks each	Exit E35 routine shortens each record by 30 bytes as it leaves the merge	Exact data set size, message option

```
//EXAMP7         JOB A402,PROGRAMMER                                   01
//STEP1          EXEC PROC=SORT,PARM='MSG=CC'                          02
//SORT.SORTIN    DD  DSNAME=INFILE,VOLUME=SER=INP214,                  03
//               UNIT=2311,DCB=(RECFM=F,LRECL=80,                      04
//               BLKSIZE=80),DISP=(OLD,DELETE)                         05
//SORT.SORTOUT   DD  DSNAME=OUTFILE,VOLUME=SER=DLIB02,                 06
//               UNIT=2311,DCB=(RECFM=FB,LRECL=50,                     07
//               BLKSIZE=500),DISP=(NEW,KEEP),                         08
//               SPACE=(TRK,(500,5))                                   09
//SORT.SORTWK01  DD  UNIT=2311,SPACE=(TRK,(120),,CONTIG)               10
//SORT.SORTWK02  DD  UNIT=2311,SPACE=(TRK,(120),,CONTIG)               11
//SORT.SORTWK03  DD  UNIT=2311,SPACE=(TRK,(120),,CONTIG)               12
//SORT.SORTMODS  DD  UNIT=2311,SPACE=(TRK,(10,,3))                     13
//SORT.SYSIN     DD  *                                                 14
  SORT  FIELDS=(10,5,CH,A),SIZE=10000                                 15
  RECORD  TYPE=F,LENGTH=(80,,50)                                      16
  MODS    E35=(E35,534,SYSIN)                                         17
  END                                                                 18
Object deck for E35
/*                                                                    19
```

01   Standard JOB statement.

02   The EXEC statement invokes the SORT cataloged procedure and specifies
          that critical messages only are to be printed and they are to appear on
          the console typewriter.

03-05   The input data set consists of fixed-length unblocked records on volume
          INP214 on a 2311 disk storage drive. The data set will be deleted after
          this job step.

06-09   The output data set is composed of fixed-length blocked records that
          will require 500 tracks of 2311 disk. Each time space is exhausted, 5
          additional tracks will be allotted. The data set will be retained for
          future reference.

10-12   Intermediate storage consists of three 2311 areas of 120 contiguous
          tracks each.

13   This DD statement defines a data set large enough to contain the E35
          routine which appears in object form in SYSIN. Ten disk tracks are
          reserved for the partitioned data set plus three blocks of the
          directory.

14   A data set follows in the input stream.

15-18   Sort/Merge control statements described in Example 5 at the end of the
          topic "Defining the Sort or Merge".

19   Delimiter statement marks the end of the SYSIN data set.

Example 8 -- Sort

Input	Output	Intermediate Storage	User Modifications	Options
Variable-length blocked records on 2314	Variable-length blocked records on 2314	Six 2314 areas	Initialization routine at exit E11 and an Nmax error routine at E16	Crisscross technique forced, Message option, estimated data set size.

```
//EXAMP8          JOB A402,PROGRAMMER                                        01
//STEPONE         EXEC SORT,PARM='CRCX,MSG=CP'                               02
//SORT.SORTIN     DD   UNIT=2314,DSNAME=PAY413,                             03
//                VOLUME=SER=231401,DCB=(RECFM=VB,                          04
//                LRECL=120,BLKSIZE=840),DISP=(OLD,KEEP)                    05
//SORT.SORTOUT    DD   UNIT=2314,DSNAME=PAY414,                            06
//                VOLUME=SER=231404,DCB=(RECFM=VB,                          07
//                LRECL=120,BLKSIZE=840),DISP=(NEW,KEEP),SPACE=(CYL,(50,10)) 08
//SORT.SORTWK01   DD   UNIT=2314,SPACE=(CYL,(5),,CONTIG)                    09
//SORT.SORTWK02   DD   UNIT=(2314,SEP=(SORT.SORTWK01)),SPACE=(CYL,(5),,CONTIG) 10
//SORT.SORTWK03   DD   UNIT=(2314,SEP=(SORT.SORTWK01,SORT.SORTWK02)),       11
//                SPACE=(CYL,(5),,CONTIG                                    12
//SORT.SORTWK04   DD   SEP=SORT.SORTWK01,UNIT=2314,                        13
//                SPACE=(CYL,(5),,CONTIG)                                  14
//SORT.SORTWK05   DD   AFF=SORT.SORTWK04,UNIT=(2314,SEP=(SORT.SORTWK04)),  15
//                SPACE=(CYL,(5),,CONTIG)                                  16
//SORT.SORTWK06   DD   AFF=SORT.SORTWK04,UNIT=(2314,SEP=(SORT.SORTWK04,    17
//                SORT.SORTWK05)),SPACE=(CYL,(5),,CONTIG)                  18
//SORT.USERLIB    DD   DSNAME=JIMSMODS,DISP=OLD                            19
//SORT.SYSIN      DD   *                                                    20
  SORT   FIELDS=(20,5,CH,A),SIZE=E25500                                    21
  MODS   E11=(E11,500,USERLIB,S),E16=(E16,554,USERLIB,N)                   22
  RECORD   TYPE=V,LENGTH=(120,,,60,80)                                     23
  END                                                                      24
/*                                                                         25
```

01   The standard JOB statement.

02   The EXEC statement specifies the SORT cataloged procedure. The options
            in the PARM field indicate that the program is to use the crisscross
            sequence distribution technique if possible, that critical messages only
            are to be printed and that they are to appear on the printer.

03-05   The SORTIN DD statement describes the input data set. Its name is
            PAY413, it is on volume 231401 on a 2314, and consists of variable
            length blocked records. The data set is known to the operating system
            and is to be retained after use.

06-08   This statement describes the output data set. The data set, named
            PAY414, will be on volume 231404 of a 2314, will consist of variable
            length blocked records, is being created in this job step, and is to be
            retained in the system.

09-18   These statements define intermediate storage data sets. There are six
            data sets of 5 contiguous cylinders each and they are on 2314. Six data
            sets is the minimum required for the crisscross technique. SEP and AFF
            parameters are used to ensure each area will be on a different unit, for
            maximum efficiency.

19   Defines a data set called JIMSMODS which contains the E11 and E16
            modification routines described on the MODS statement. The data set is
            known to the operating system and is not to be deleted after this job
            step.

20   A data set follows in the input stream.

21-24   Sort/merge control statements described in Example 6 at the end of topic
            "Defining the Sort or Merge".

25   Delimiter statement marking the end of the SYSIN data set.

Example 9 -- Merge

Input	Output	Intermediate Storage	User Modifications	Options
Variable-length blocked records on 2301	Variable-length blocked records on 2301	None	E35 routine shortens records and E61 routine modifies control field	Exact input data set size

```
//EXAMP9          JOB A402,PROGRAMMER                            01
//STEP1           EXEC SORT                                      02
//SORT.SORTIN01   DD  DSNAME=WEEKLY,VOLUME=SER=000101,           03
//                UNIT=2301,DISP=OLD,DCB= (RECFM=VB,             04
//                LRECL=240,BLKSIZE=2400)                        05
//SORT.SORTIN02   DD  DSNAME=DAILY,VOLUME=SER=000113,            06
//                UNIT=2301,DISP= (OLD,DELETE) ,                 07
//                DCB= (RECFM=VB,LRECL=240,BLKSIZE=2400)         08
//SORT.SORTOUT    DD  DSNAME=WEEKA,VOLUME=SER=000111,            09
//                UNIT=2301,DISP= (NEW,KEEP) ,                   10
//                SPACE= (TRK, (75, 10) ) ,DCB= (RECFM=VB,       11
//                LRECL=200,BLKSIZE=2000)                        12
//SORT.USERLIB    DD  DSNAME=MYMODS,DISP=OLD                     13
//SORT.MODLIB     DD  DSNAME=XYZ,DISP=OLD                        14
//SORT.SYSIN      DD  *                                          15
  MERGE   FIELDS= (5,6,CH,E) ,SIZE=8150                          16
  RECORD  TYPE=V,LENGTH= (240,,200,,160)                         17
  MODS   E35= (CALC,800,USERLIB) ,E61= (E61,450,MODLIB,N)        18
  END                                                            19
/*                                                               20
```

01-02   The basic JOB and EXEC statements.

03-05   The SORTIN01 DD statement describes one of two input data sets for the merge. The data set, named WEEKLY, is on volume 000101 of a 2301. The data set is known to the operating system and is to be retained. It contains variable length blocked records with a maximum record length of 240 bytes and a blocksize of 2400.

06-08   The SORTIN02 DD statement describes the second of two inputs to the merge. It is named DAILY, is on volume 000113 of a 2301, is old and will be deleted after this job step, and contains records of the same format, length and block size as the WEEKLY data set.

09-12   The output from the merge will be a data set named WEEKA. It is new and will be retained in the system on volume 000111 of a 2301. The data set will be recorded on 75 drum tracks. If this space is not sufficient, additional space will be allotted in blocks of ten tracks. The data set consists of variable-length blocked records with a maximum record length of 200 (see l_3 on the RECORD statement) and a block size of 2000.

13   Defines the library on which the CALC routine for exit E35 resides.

14   Defines the library on which the E61 routine for exit E61 resides.

15   A data set follows in the input stream.

16-19   Sort/merge control statements described in Example 4 at the end of the topic "Defining the Sort or Merge".

20   Standard delimiter statement.

Example 10 -- Simple Merge

Input	Output	Intermediate Storage	User Modifications	Options
Blocked fixed-length records on 3 7-track tapes	Blocked fixed-length records on one 7-track tape	None	None	Estimated input data set size

```
//EXAMP10        JOB A714,PROGRAMMER                                    01
//STEPA          EXEC SORTD                                             02
//SORT.SORTIN01  DD  DSNAME=FILE1,VOLUME=SER=000123,                    03
//                   UNIT=2400-2,DCB=(DEN=2,RECFM=FB,                   04
//                   LRECL=80,BLKSIZE=800,TRTCH=ET),                    05
//                   DISP=(OLD,DELETE)                                  06
//SORT.SORTIN02  DD  DSNAME=FILE2,VOLUME=SER=000225,                    07
//                   UNIT=2400-2,DCB=(DEN=2,RECFM=FB,                   08
//                   LRECL=80,BLKSIZE=800,TRTCH=ET),                    09
//                   DISP=(OLD,DELETE)                                  10
//SORT.SORTIN03  DD  DSNAME=FILE3,VOLUME=SER=000179,                    11
//                   UNIT=2400-2,DCB=(DEN=2,RECFM=FB,                   12
//                   LRECL=80,BLKSIZE=800,TRTCH=ET),                    13
//                   DISP=(OLD,DELETE)                                  14
//SORT.SORTOUT   DD  DSNAME=FILE123,VOLUME=SER=000111,                  15
//                   UNIT=2400-2,DCB=(DEN=2,RECFM=FB,                   16
//                   LRECL=80,BLKSIZE=800,TRTCH=ET),                    17
//                   DISP=(NEW,KEEP)                                    18
//SORT.SYSIN     DD  *                                                  19
  MERGE  FIELDS=(1.0,6.0,A,28,5,D),FORMAT=CH,SIZE=E10000               20
  END                                                                  21
/*                                                                     22
```

01-02   Standard JOB and EXEC statements.

03-06   Defines one of three inputs to the merge. The data set's name is FILE1. It is on 7-track tape with a serial number of 000123, and consists of fixed-length blocked records. The TRTCH=ET DCB subparameter indicates that the tape was recorded with even parity and that BCDIC to EBCDIC translation is required.

07-10   Defines another of the inputs to the merge, a data set named FILE2.

11-14   Defines FILE3, the third input to the merge.

15-18   Defines the output data set which is named FILE123. The data set is to be recorded on 7-track tape, volume 000111. The other parameters are the same as those for SORTIN01, with the exception of DISP, which indicates that the data set is new and is to be retained for future reference.

19   Data set follows in the input stream.

20-21   Sort/merge control statements described in Example 2 at the end of the topic "Defining the Sort or Merge".

22   Delimiter statement.

Example 11 -- Sort

Input	Output	Intermediate Storage	User Modifications	Options
Fixed-length blocked records on 2314	Fixed-length blocked records on 2314	Eight 2314 areas of 20 tracks each	One routine shortens records as they leave the final merge phase	Exact data set size

```
//EXAMP11        JOB  B600,PROGRAMMER                            01
//STEPI          EXEC PROC=SORT                                  02
//SORT.SORTIN    DD   DSNAME=INPUT,UNIT=2314,VOLUME=SER=231401,  03
//               DCB= (RECFM=FB,LRECL=80,BLKSIZE=800) ,          04
//               DISP= (OLD,DELETE)                              05
//SORT.SORTWK01  DD   UNIT=2314,VOLUME=SER=231402,               06
//               SPACE= (TRK, (20) ,,CONTIG)                     07
//SORT.SORTWK02  DD   UNIT=2314,VOLUME=SER=231403,               08
//               SPACE= (TRK, (20) ,,CONTIG)                     09
//SORT.SORTWK03  DD   UNIT=2314,VOLUME=SER=231404,               10
//               SPACE= (TRK, (20) ,,CONTIG)                     11
//SORT.SORTWK04  DD   UNIT=2314,VOLUME=SER=231405,               12
//               SPACE= (TRK, (20) ,,CONTIG)                     13
//SORT.SORTWK05  DD   UNIT=2314,VOLUME=SER=231406,               14
//               SPACE= (TRK, (20) ,,CONTIG)                     15
//SORT.SORTWK06  DD   UNIT=2314,VOLUME=SER=231407,               16
//               SPACE= (TRK, (20) ,,CONTIG)                     17
//SORT.SORTWK07  DD   UNIT=2314,VOLUME=SER=231408,               18
//               SPACE= (TRK, (20) ,,CONTIG)                     19
//SORT.SORTWK08  DD   UNIT=2314,VOLUME=SER=231409,               20
//               SPACE= (TRK, (20) ,,CONTIG)                     21
//SORT.SORTOUT   DD   DSNAME=OUTPUT,UNIT=2314,                   22
//               VOLUME=SER=231410,DCB= (RECFM=FB,               23
//               LRECL=50,BLKSIZE=500) ,DISP= (NEW,KEEP) ,       24
//               SPACE= (TRK, (200,10) ,RLSE)                    25
//SORT.SORTMODS  DD   UNIT=2314,SPACE= (TRK, (10,,2) )           26
//SORT.SYSIN     DD   *                                          27
  SORT  FIELDS= (10,5,CH,A) ,SIZE=10000                          28
  RECORD  TYPE=F,LENGTH= (80,,50)                                29
  MODS  E35= (E35,534,SYSIN)                                     30
  END                                                            31
Object deck for E35 routine
/*
```

01-02   Standard JOB and EXEC statements.

03-05   Defines the input data set. It is named INPUT, is on 2314 volume
         231401, consists of fixed-length, blocked records with a length of 80
         bytes and a blocking factor of 10.

06-21   These statements describe eight 2314 work areas. Each area consists of
         20 contiguous tracks.

22-25   Defines the output data set. The data set, named OUTPUT, will be on
         volume 231410 of a 2314 and will contain fixed-length blocked records.
         Two hundred tracks are requested for the data set; if the space is
         exhausted, additional tracks are to be assigned in blocks of ten. When
         the output data set is closed, unused tracks are to be released.

26   Defines a temporary data set on 2314 for the E35 routine.

27   A data set follows in the input stream.

28-31   Sort/merge control statements described in Example 5 at the end of the
         topic "Defining the Sort or Merge".

32   Delimiter statement.

72

Example 12 -- Sort

Input	Output	Intermediate Storage	User Modifications	Options
Variable-length records on 2301	Variable-length records on 2301	Four 2301 areas of 60 tracks each	E11 routine performs initialization for the E16 NMAX routine	Estimated data set size

```
//EXAMP12        JOB B999,PROGRAMMER                              01
//STEP0          EXEC SORT                                        02
//SORT.SORTIN    DD  DSNAME=XFILE,VOLUME=SER=000230,              03
//               UNIT=2301,DISP=OLD,DCB=(RECFM=V,                 04
//               LRECL=120,BLKSIZE=124)                           05
//SORT.SORTWK01  DD  UNIT=2301,VOLUME=SER=230102,                 06
//               SPACE=(TRK,(60),,CONTIG)                         07
//SORT.SORTWK02  DD  UNIT=2301,VOLUME=SER=230197,                 08
//               SPACE=(TRK,(60),,CONTIG)                         09
//SORT.SORTWK03  DD  UNIT=2301,VOLUME=SER=000106,                 10
//               SPACE=(TRK,(60),,CONTIG)                         11
//SORT.SORTWK04  DD  UNIT=2301,VOLUME=SER=000145,                 12
//               SPACE=(TRK,(60),,CONTIG)                         13
//SORT.SORTOUT   DD  DSNAME=YFILE,VOLUME=SER=230198,              14
//               UNIT=2301,DCB=(RECFM=V,LRECL=120,                15
//               BLKSIZE=124),SPACE=(TRK,(170,10),RLSE),          16
//               DISP=(NEW,CATLG)                                 17
//SORT.USERLIB   DD  DSNAME=MYRTNS,DISP=OLD                       18
//SORT.SYSIN     DD  *                                            19
  SORT  FIELDS=(20,5,CH,A),SIZE=E25500                            20
  MODS  E11=(E11,500,USERLIB,S),E16=(E16,554,USERLIB,N)           21
  RECORD  TYPE=V,LENGTH=(120,,,60,80)                             22
  END                                                            23
/*                                                               24
```

01-02   Standard JOB and EXEC statements.

03-05   Defines the input data set. It is named XFILE, resides on volume 000230 of a 2301, is known to the operating system and is not to be deleted, and consists of variable-length unblocked records.

06-13   Define four intermediate storage areas on 2301. Each area consists of 60 contiguous tracks.

14-17   Defines the output data set. It is named YFILE, and is to be placed on volume 230198 of a 2301. It will contain records of the same format as the input data set. One hundred seventy tracks are requested for the data set. If they are not sufficient to contain it, additional tracks are requested in blocks of ten. The data set is being created in this job step and is to be cataloged.

18   Defines the library that contains the E11 and E16 modification routines.

19   A data set follows.

20-23   Sort/merge control statements described in Example 6 at the end of the topic "Defining the Sort or Merge".

24   Delimiter statement.

Initiating Sort/Merge

There are two ways to initiate a sorting operation:

- By including sort/merge control statements and job control language statements in the input stream. You can use a cataloged procedure to supply some of the job control language statements.

- By using ATTACH, LINK, or XCTL macro instructions issued by another program.

There is only one way to initiate a merging operation: by placing sort/merge control statements and JCL statements in the input stream. As with a sort, a cataloged procedure can be used to supply some of the JCL.

USING THE SYSTEM INPUT STREAM

When sort/merge program execution is initiated by control statements in the input stream, it is treated as an ordinary task being executed under operating system control. You must provide a JOB statement, an EXEC statement and several DD statements to communicate with the operating system and the sort/merge program.

The job that initiates sort/merge requires a JOB statement. Each job step within that job requires an EXEC statement. (Other job steps may precede and follow the sort/merge job step.) The EXEC statement that introduces the sort/merge job step can initiate execution either directly or through a cataloged procedure. DD statements are required to define data sets used by the sort/merge program, the system, and, if necessary, the linkage editor.

Cataloged Procedure SORT

The SORT cataloged procedure is designed to be used in sorting and merging applications that have modification routines that require linkage editing. You can use this procedure for all sort/merge applications, but it is inefficient for those that do not have modification routines that require linkage editing, because it causes unnecessary linkage editor data sets to be allocated.

The SORT cataloged procedure is:

```
//SORT        EXEC   PGM=IERRCO00,REGION=98K                        01
//SYSOUT      DD     SYSOUT=A                                       02
//SYSPRINT    DD     DUMMY                                          03
//SYSLMOD     DD     DSNAME=&GOSET,UNIT=SYSDA,SPACE=(3600,(20,20,1))  04
//SYSLIN      DD     DSNAME=&LOADSET,UNIT=SYSDA,SPACE=(80,(10,10))   05
//SORTLIB     DD     DSNAME=SYS1.SORTLIB,DISP=SHR                   06
//SYSUT1      DD     DSNAME=&SYSUT1,SPACE=(1000,(60,20)),           07
//                   UNIT=(SYSDA,SEP=(SORTLIB,SYSLMOD,SYSLIN))      08
```

01   The stepname of the procedure is SORT. This EXEC statement initiates the sort/merge program, which is named IERRCO00. A 98K region, large enough to contain the largest linkage editor, is requested.

02   This DD statement defines an output data set for system use (messages). It is directed to system output class A.

03   SYSPRINT is defined as a dummy data set because linkage editor diagnostic output is not required.

04   This DD statement defines a data set for linkage editor output. Any system direct access device is acceptable for the output. Space for 20 records that have an average length of 3,600 bytes is requested; this is the primary allocation. Space for 20 more records is requested if the primary space allocation is not sufficient; this is the secondary allocation, which is requested each time space is exhausted. The last value is space for a directory, which is required because SYSLMOD is a new partitioned data set.

05   The SYSLIN data set is used by the sort/merge program to create control
        records for the linkage editor. It is created on any system direct
        access device, and it has space for 10 records with an average length of
        80 bytes. If the primary space allocation is exhausted, additional
        space is requested in blocks large enough to contain 10 records. No
        directory space is necessary.

06   The SORTLIB DD statement defines the data set that contains the
        sort/merge program modules. It has the qualified name SYS1.SORTLIB, and
        it is cataloged.

07-08   The SYSUT1 DD statement defines a work data set for the linkage editor.

Cataloged Procedure SORTD

The SORTD cataloged procedure is designed for sorting and merging applications
that have no modification routines, or have modification routines that do not
require linkage editing. It cannot be used for applications having modification
routines that need linkage editing.

The SORTD cataloged procedure is:

```
//SORT          EXEC  PGM=IERRCO00,REGION=26K                              01
//SYSOUT        DD    SYSOUT=A                                             02
//SORTLIB       DD    DSNAME=SYS1.SORTLIB,DISP=SHR                         03
```

01   The stepname of the SORTD procedure is SORT. A 26K region is the
        smallest in which the program can operate.

02   System output is directed to system output class A.

03   This DD statement defines the data set containing sort/merge program
        modules.

USING ATTACH, LINK OR XCTL

You can use ATTACH, LINK, or XCTL macro instructions in another program to
initiate operation of a sorting application (but not a merging application).
(For a full description of ATTACH, LINK, and XCTL, see the publication OS
Supervisor Services and Macro Instructions.)

   There are four differences between initiating sort in the input stream and
initiating it by a macro instruction:

1. Sort DD statements must be placed in the input stream with the job step that
    issues the macro instruction.
2. Information normally contained on sort/merge control statements must be
    passed to the sort/merge program in a parameter list.
3. Only two sort/merge program exits for modification routines (E15 and E35,
    see "Section 3: Program Modification") can be used when the sort is
    initiated by a macro instruction.
4. If ATTACH is used, checkpoints cannot be taken.

Supplying the Needed DD Statements

When you ATTACH, LINK, or XCTL to the sort/merge program, you must supply the
following DD statements in the input stream with the job step that issues the
macro instruction:

```
//SORTLIB       DD  DSNAME=SYS1.SORTLIB,DISP=SHR
```

        to define the data set that contains sort/merge program modules.

```
//SORTIN         DD   with appropriate parameters
```

       (See the examples at the end of "Job Control Language for Sort/Merge") to
        define the data set(s) to be sorted.

```
//SORTWK01-32 DD   with appropriate parameters
```

       to define the intermediate storage data sets required by the sort.

```
//SYSOUT         DD   SYSOUT=A
```

       to define an output data set (messages) for system use.

```
//SORTOUT        DD   with appropriate parameters
```

       to define the sort/merge output data set.

Note: If you activate sort/merge exit E15, the SORTIN DD statement is not
necessary because your routine must supply all input for the sort. If you
activate exit E35, the SORTOUT DD statement is not necessary because your routine
must handle output from the sort. You may need DD statements to describe your
sort input and to set up a data set for your output, but they need not be called
SORTIN or SORTOUT.

Passing Parameters to the Sort

The parameters you pass to sort/merge consist of two control statement images --
SORT and RECORD -- in main storage, and the entry point addresses of your
modification routines (E15 and E35). These are the only modification routines
permitted when sort/merge is initiated by ATTACH, LINK, and XCTL, and they are
optional. You need not use any modification routines.

    Your routine must construct the following parameter list and place a pointer
to it in general register 1 before issuing the control-passing macro instruction:

X'80'	Pointer to list of addresses and options
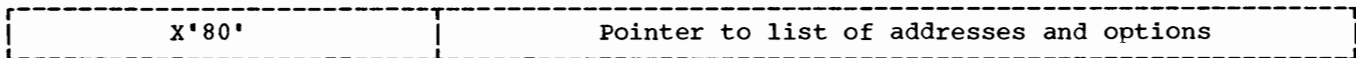
The format of the address list is:

Unused	Number of bytes in the following list
Starting address of the SORT statement	
Ending address of the SORT statement	
Starting address of the RECORD statement	
Ending address of the RECORD statement	
Address of the E15 routine or zeros if no routine is provided	
Address of the E35 routine or zeros if no routine is provided	
Optional characters for ddnames	
X'00'	Optional main storage value
Optional sequence distribution technique	

X'FF'	Unused	Message option
Characters for DIAG (diagnostic message option)		

The address list is variable in length. The first halfword shown in the above illustration is not considered part of the list. The next halfword, which is pointed to by the parameter list pointer, contains the number of bytes in the parameter list excluding the two bytes occupied by the number itself. The list must contain at least 24 bytes because none of the addresses can be omitted. (The E15 and E35 routine addresses are zeros if the routines are not used.) The list can be as long as 40 bytes if all the options are included.

The first address in the address list must begin on a fullword boundary. Each address is contained in the low order three bytes of a fullword.

The following rules apply to the SORT and RECORD statement images whose starting and ending addresses appear in the address list:

- The first and last bytes of each statement image must contain a blank, and a blank (one only) must follow SORT and RECORD. No other blanks are allowed.

- The contents and formats of the SORT and RECORD statements are the same as those described in Section 2 under "Defining the Sort or Merge" except that continuation characters are not allowed. In other words, the statement images are not set up in card image format. Each statement image can be up to 1,100 bytes long.

- No comments are permitted.

The six addresses (or four addresses and two words of zeros) must appear in the order shown in the list. The options following the addresses can appear in any order and any of them can be omitted. For example, to specify only the optional main storage value, construct the list as follows:

Unused	Count
Address	
Address	
Address	
Address	
Address or zeros	
Address or zeros	
X'00'	Optional Main Storage Value

To specify only the balanced sequence distribution technique, construct:

Unused		Count	
Address			
Address			
Address			
Address			
Address or zeros			
Address or zeros			
B	A	L	N

OPTIONAL CHARACTERS FOR DDNAMES: You must select this option if you are
operating in a multiprogramming environment and your task initiates two or more
sort applications via ATTACH, LINK, or XCTL. The four characters you place in
this word of the address list will replace the characters "SORT" in the DD names
of the standard DD statements that define input, intermediate storage, and
output. For the four characters, you can use any alphameric characters and the
special characters $, #, and @, but the first must be alphabetic. If it is not,
the characters are ignored. For example, if you use the characters ABC# as
replacement characters, the statements SORTIN, SORTWK01 - SORTWK32, and SORTOUT
from the input stream will be converted internally to ABC#IN, ABC#WK01 -
ABC#WK32, and ABC#OUT.

Caution: Do not use characters that conflict with other ddnames; do not use the
characters BALN, OSCL, POLY, CRCX, or DIAG.

OPTIONAL MAIN STORAGE VALUE: This parameter serves the same purpose as the CORE
parameter in the EXEC statement PARM field. With it, you can specify the amount
of main storage sort/merge can use for this application. The value you specify
temporarily overrides the main storage assigned to the sort at system generation.
The value must be a binary number and must appear right justified in the last
three bytes of the field. As shown in the address list format, the high-order
byte must contain zeros. The new value must not be less than 12,000, the minimum
number of bytes needed for sort/merge operation. If it is, the number 12,000 is
chosen by default. Refer to the topic "Altering the Main Storage Allocation" in
Section 4 for further information.

OPTIONAL SEQUENCE DISTRIBUTION TECHNIQUES: This parameter takes the place of
another PARM field option. With it you can force the sort/merge program to
choose the balanced, oscillating, or polyphase technique for tape intermediate
storage or the balanced or crisscross technique for disk. The four valid entries
for this parameter are BALN, OSCL, POLY, and CRCX. Refer to the topic "Sequence
Distribution Techniques" in Section 1 for further information.

   This parameter may be ignored under the following conditions:

Tape Sorting

  • Only three intermediate storage tape drives are assigned. With only three
    drives, the polyphase technique is always used.

  • No input data set size, exact or estimated, is specified on the SORT
    statement. When the sort/merge program is not given an input data set size,
    it always uses the balanced technique if more than three work tapes are
    available.

  • The tape drive containing the input data set is also specified as an
    intermediate storage unit. In this case, the oscillating technique cannot be
    used, so the sort/merge program chooses either the balanced or polyphase
    technique.

Disk Sorting

  • Technique forcing can occur only on a 2314 facility. All direct access
    sorting on 2311 disks and 2301 drums uses the balanced technique.

  • Whenever less than six work areas are available, only the balanced technique
    can be used on the 2314.

  • Whenever more than six work areas are available, only the crisscross
    technique can be used on the 2314.

MESSAGE OPTION: This parameter takes the place of the third EXEC statement PARM
field option, MSG. The parameter temporarily overrides the message option
selected at system generation.

78

The high-order byte of this parameter must be X'FF'. The next byte is unused. The last two bytes must contain one of the following codes:

        NO -- no messages printed
        CC -- critical messages only, printed on the system console
        CP -- critical messages only, printed on the printer
        AC -- all messages, printed on the system console
        AP -- all messages, printed on the printer

DIAG OPTION: Serves the same purpose as the DIAG parameter in the EXEC statement PARM field.

Considerations When Using XCTL

When you initiate sort/merge via XCTL, you must give special consideration to the area where the parameter list, address list, and optional parameters, and modification routines (if you use them) are stored. This information must not reside in the module that issues the XCTL because the module is frequently overlaid by the sort/merge program.

There are two ways to overcome this problem. First, the control information can reside in a task that attaches the module that issues the XCTL. Second, the module issuing the XCTL can first issue a GETMAIN macro instruction and place the control information in the main storage area it obtains. This area is not overlaid when the XCTL is issued. The address of the control information in the area must be passed to the sort/merge program in general register 1.

The following text contains two examples. The first illustrates passing parameters to the sort. The second is an assembler language coding example that shows how to set up the parameter list, address list, and optional fields.

Example 1

Figure 15 shows how the parameter list, address list, and optional fields might appear in main storage.

General register 1 contains a pointer to the parameter list, which is at location 1000. The parameter list points to the address list which begins at location 1006. The first halfword of the address list contains, right adjusted, in hexadecimal, the number of bytes in the list (40 decimal).

The first two fullwords in the address list point to the beginning (location 1036) and end (location 105A) of the SORT control statement. The next two fullwords point to the beginning (location 105B) and end (location 1074) of the RECORD statement.

The fourth and fifth fullwords in the list contain the entry point addresses of modification routines for exit E15 (2000) and exit E35 (3000).

The next fullword in the list contains four characters to replace the letters "SORT" in the DD names of standard DD statements.

The next three fullwords in the list specify a main storage value for this application, a sequence distribution technique, and a message option.

The control statement images must be represented in EBCDIC. The symbol b in the figure stands for a blank character.

```
General Register 1        Parameter List          Address List

     Contents            Location Contents        Location Contents
   ┌─────────┐
   │  1000   ├──────────► 1000   Byte 1
   └─────────┘                 ┌─────────┬────────┐
                               │ X'80'   │ 001006 ├─┐      1004 ┌──┬──┬──┬──┐
                               └─────────┴────────┘ │           │00│00│00│28│
                                                    │           ├──┼──┼──┼──┤
  1036                        105A ┐                │      1008 │00│00│10│36│
   ↓                              ↓                 │           ├──┼──┼──┼──┤
   bSORT FIELDS=(10,15,CH,A),SIZE=47806b            │      100C │00│00│10│5A│
                                                    │           ├──┼──┼──┼──┤
                                                    │      1010 │00│00│10│5B│
  105B                        1074 ┐                │           ├──┼──┼──┼──┤
   ↓                              ↓                        1014 │00│00│10│74│
   bRECORD LENGTH=100,TYPE=Fb                             ├──┼──┼──┼──┤
                                                     1018 │00│00│20│00│
                                                          ├──┼──┼──┼──┤
                                                     101C │00│00│30│00│
                                                          ├──┼──┼──┼──┤
                                                     1020 │ A│ B│ C│ #│
                                                          ├──┼──┼──┼──┤
                                                     1024 │00│00│65│90│
                                                          ├──┼──┼──┼──┤
                                                     1028 │ O│ S│ C│ L│
                                                          ├──┼──┼──┼──┤
                                                     102C │FF│00│ A│ C│
                                                          └──┴──┴──┴──┘
```
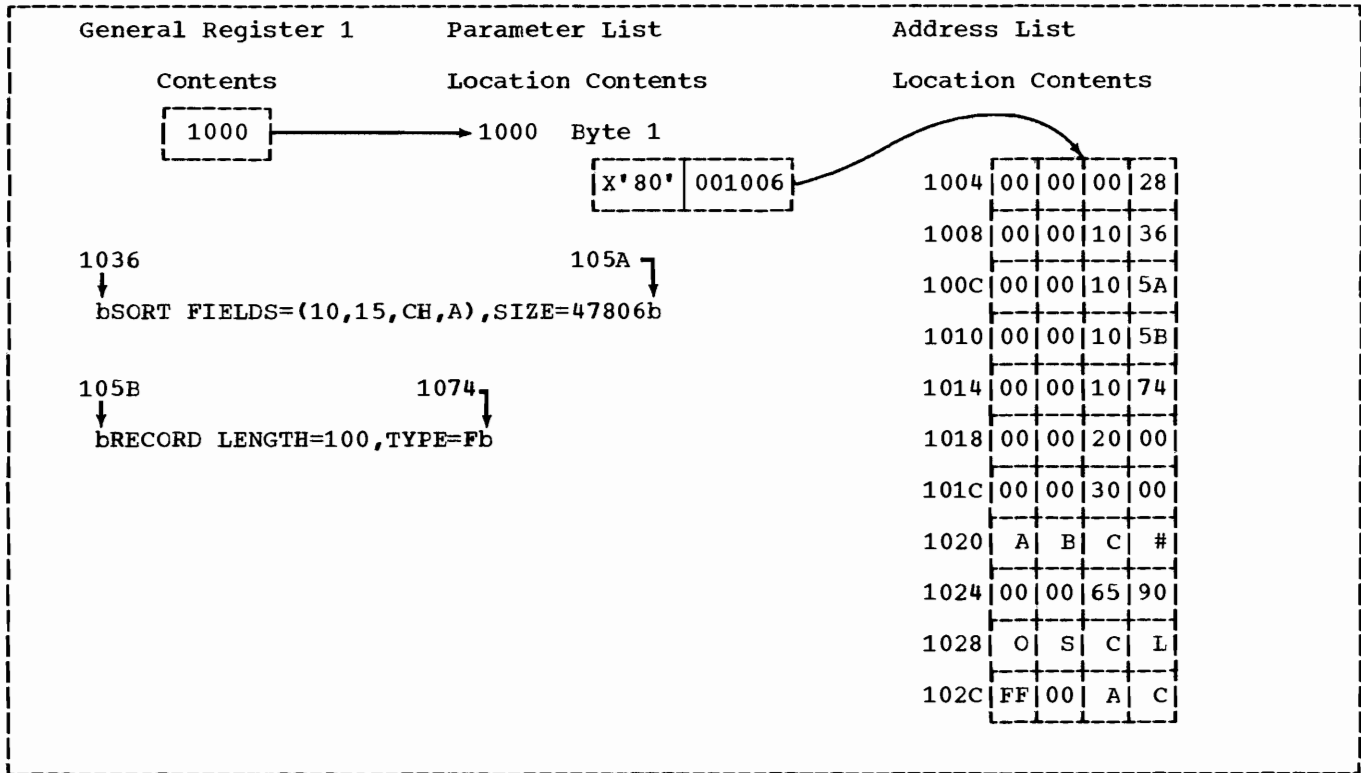
Figure 15. Passing Parameters to the Sort

Example 2

The following example shows, in assembler language coding, how to set up the
parameters and card images in Example 1, and how to pass control to the
sort/merge program.

```
             LA    1,PARLST
             ATTACH EP=SORT,MF=(E,(1))
               .

               .

               .
             CNOP  0,8
   PARLST DC     X'80'
          DC     AL3(ADLST)
          DC     X'0000'
   ADLST  DC     X'0028'
          DC     A(SORTCD)
          DC     A(STCDED)
          DC     A(RCDCD)
          DC     A(RDCDED)
          DC     A(MOD1)
          DC     A(MOD2)
          DC     C'ABC#'
          DC     X'0000'
          DC     X'6590'
          DC     C'OSCL'
          DC     X'FF00'
          DC     C'AC'
   SORTCD DC     C' SORT FIELDS=(10,15,CH,A),'
          DC     C'SIZE=4780'
   STCDED DC     C' '
   RCDCD  DC     C' RECORD LENGTH=100,TYPE=F'
   RDCDED DC     C' '
          CNOP  0,8
          USING *,15
   MOD1   routine for exit E15
               .

               .
          CNOP  0,8
          USING *,15
   MOD2   routine for exit E35
```

Further Considerations When Using ATTACH, LINK, or XCTL

If you provide a modification routine for exit E15, sort/merge ignores the SORTIN
data set. Your E15 routine must pass all input records to the sort/merge
program. This means that your routine can only issue a return code of 12 (insert
record) until the input data set is completed and then a return code of 8 (do not
return).

Similarly, sort/merge ignores the SORTOUT data set if you provide a
modification routine for exit E35. Your routine is responsible for disposing of
all output records. Your routine must issue a return code of 4 (delete record)
for each record in the output data set. When sort/merge has deleted all the
records, your routine issues RC = 8 (do not return).

When sort/merge completes execution, it passes control back to the routine
that invoked it or to the operating system.

COMPLETION CODES

The sort/merge program returns a completion code to the operating system (or other invoking program) upon termination. This code may be interrogated by succeeding job steps. The codes are:

    0 - Successful completion of sort/merge
  16 - Unsuccessful completion of sort/merge

Successful Completion: When a sort/merge application has been successfully executed, a completion code of zero is returned, and the sort terminates.

Unsuccessful Completion: If the sort, during execution, encounters an error that will not allow it to complete successfully, it returns a completion code of 16 and terminates. (Such errors include an out-of-sequence condition or an uncorrectable I/O error.)

User-written routines can be used during a sort/merge program execution to perform a variety of functions, such as deleting, inserting, altering, and summarizing records.

Control is passed to your routines at predesignated places in the executable code of the sort/merge program called sort/merge program exits. Because these exits are located in particular program phases (and in one case, in a particular module), a general understanding of how the sort/merge program operates is necessary to understand sort/merge program exits.

Program Description

The sort/merge program is a segmented program; that is, it is composed of parts that can operate independently. Generally, there are two levels of segmentation:

1. Phases -- large program components that accomplish a certain task.

2. Modules -- the independent routines of which phases are composed.

The sort/merge program is composed of five phases. All five phases are used for sorting applications, but only the first two and the last phases of the program are used for merging applications. The first two phases -- the definition and optimization phases -- are strictly initialization phases. Each of the remaining three phases -- the sort, intermediate merge, and final merge -- is divided into two components:

1. An assignment component that initializes for the operation of the phase.

2. A running component that performs the actual sorting or merging.

Figure 16 is a phase-level flowchart of the program. Each phase is explained in the following text.

DEFINITION PHASE

The definition phase reads and interprets sort/merge control statements and decides which phases, and which modules of each phase, should be used. This phase also decides which of your routines, if any, must be linkage edited. This phase has no exits for passing control to your routines.

```
            ┌─────────────┐
            │ Definition  │
            │   Phase     │
            └──────┬──────┘
                   │
                   ▼
                  ╱ User ╲
                ╱ Routines ╲   No
               ◄ to be Link ►────────┐
                ╲  Edited  ╱         │
                  ╲      ╱           │
                   │                 │
                   ▼ Yes             │
            ┌─────────────┐          │
            │  Linkage    │          │
            │  Editor     │          │
            └──────┬──────┘          │
                   │◄────────────────┘
                   ▼
            ┌─────────────┐
            │Optimization │
            │   Phase     │
            └──────┬──────┘
                   │
                   ▼
    Merge        ╱ Sort ╲
   ┌───────────◄   or    ►
   │             ╲ Merge ╱
   │               ╲   ╱
   │                │
   │                ▼ Sort
   │         ┌─────────────┐
   │         │   Sort      │
   │         │   Phase     │
   │         └──────┬──────┘
   │                │◄────────────────┐
   │                ▼                 │
   │             ╱ Interm.╲       ┌───────────┐
   │           ◄   Merge    ► Yes │ Intermed. │
   │             ╲ Needed  ╱──────│  Merge    │
   │               ╲  ?  ╱        │  Phase    │
   │                 │            └───────────┘
   └─────────────────┤
                     ▼ No
            ┌─────────────┐
            │   Final     │
            │   Merge     │
            │   Phase     │
            └─────────────┘
```
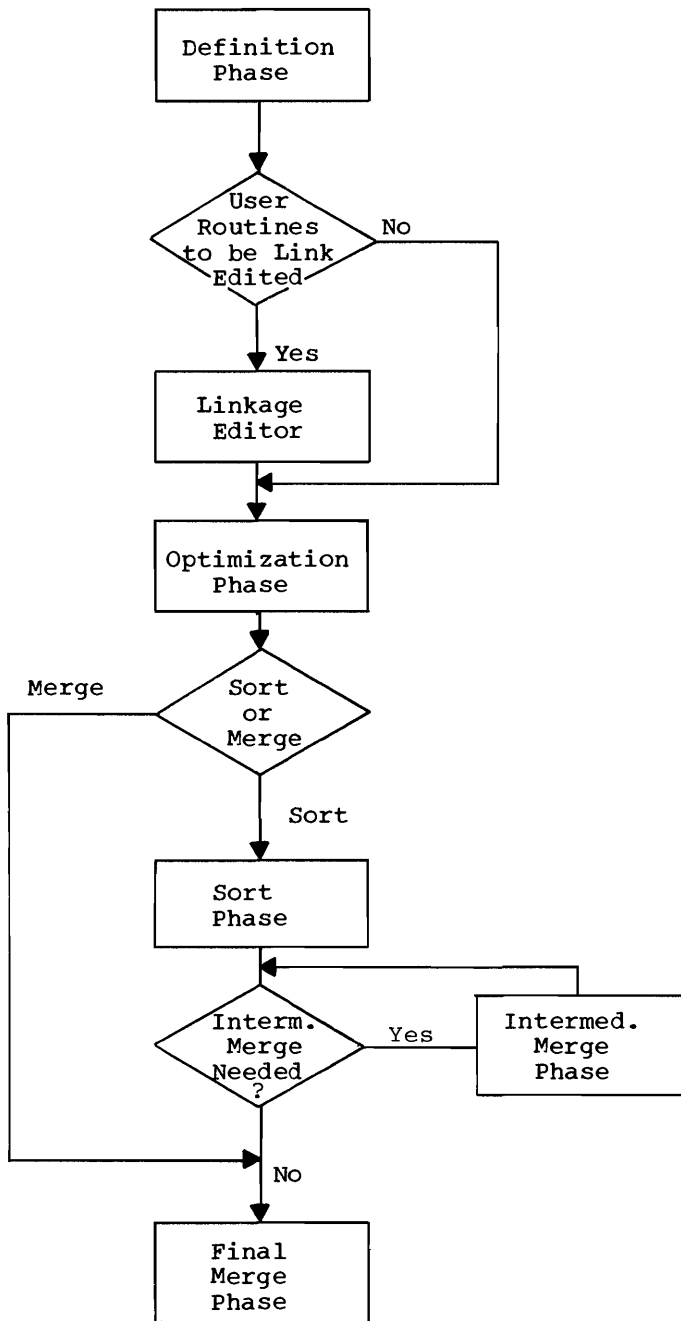
Figure 16. Phase-level Flowchart

OPTIMIZATION PHASE

The optimization phase, using information obtained from the operating system and
from DD statements, determines the optimum method of using the CPU and I/O
configuration available.

   This phase also generates special routines, if necessary, to perform record
comparisons. One of two routines -- the equals module or the extract module --
may be generated to make record comparisons. (Neither routine is used when
sorting or merging is based on a single control field containing character data
or binary data beginning and ending on a byte boundary.) If one of these
routines is used, it remains in main storage throughout execution of the program.

84

Equals Module

The equals module is used when there are from two to twelve control fields and all control fields contain character data or binary data beginning and ending on a byte boundary. It is executed to resolve the collating of records when an equal comparison arises between two major control fields. This is done by comparing successive minor control fields until an unequal compare is made, thus determining the proper order of the two records. If all control fields are equal, the records are taken in the order which requires the least internal processing time.

Extract Module

The function of the extract module is to extract and group all of the control fields into one field so that a single compare instruction can be executed to collate the record.

The extract module is loaded for one of two reasons:

1. If more than one control field is used and the equals module cannot be used to resolve collating.

2. If specified by the user in either the SORT or MERGE control statement. (User specifications is accomplished by taking the E option for the s parameters of the FIELDS operand. See the topic "Defining the Sort or Merge" in Section 2 for further information.)

When the extract module is used, it is executed each time a logical record is processed. This is done to avoid carrying the extracted information with the records, which would increase I/O time and, therefore, total sort or merge time.

SORT PHASE

The job of the sort phase is to order the input data set into sequences and distribute these sequences onto intermediate storage data sets. The method of distribution is determined by the sequence distribution technique being used.

If tape is being used as intermediate storage, the sequences may be put out in both ascending and descending order. This enables the intermediate merge phase, using the read-backward feature, to merge the sequences without rewinding tapes.

If direct access intermediate storage is used, the sequences are distributed among the areas assigned to the program so that they may be merged in a minimum number of passes.

This phase has a number of exits at which control can be passed to your routines.

INTERMEDIATE MERGE PHASE

The intermediate merge phase is loaded and executed following completion of the sort phase. There are several exits in this phase at which your routines can receive control. It performs successive merges of the strings produced by the sort phase. The merges are carried out from intermediate storage device to intermediate storage device, each successive merge decreasing the number of strings and increasing the average string length. When one more merge is required to create one long string (the output data set), control is given to the final merge phase; note that if only one merge pass is required after the sort phase, the intermediate merge phase (and any associated routines) will be skipped entirely, and message IER049I issued.

FINAL MERGE PHASE

The final merge phase has two uses:

1. It makes the final merge pass of a sorting application, thus creating the output data set.

2. It merges the input data sets for a merging application to create the output data set.

   Output from this phase can be on any output device supported by QSAM. After the execution of this phase, the sort system control component returns control to the operating system via the RETURN macro instruction. Your routines can receive control at a number of exits in this phase.

General Information

There are two types of exits available with the sort/merge program.

1. Assignment component exits, one each for the sort, intermediate merge, and final merge phases.

2. Running component exits, a number of which are associated with the running component of each program phase.

You can use assignment component exits to initialize your routines in each phase or to open data sets needed by your routines. The sort/merge assignment components are overlaid and used as buffer areas by the running components. Your routines at assignment component exits are also overlaid unless you linkage edit them together with the other routines in the phase.

   You can use running component exits for a variety of purposes including the deletion, summarization, insertion, or any other alteration of the records coming into or out of the phase. You can also use running component exit routines to correct some of the errors that may occur during sort/merge execution, including I/O errors and exceeding Nmax. These exits also give you an opportunity to provide a routine that will close any data sets used by your other routines.

   When altering or shortening a record, the sort/merge control fields in the record itself must not be changed or deleted except in exit E35 when the user can inhibit further sequence checking.

   You can use a running component extract module exit (E61) to alter control fields temporarily before the program collates them. This is the exit you would use to normalize floating-point control fields.

   Figure 17 is a summary of the sort/merge program exits and their uses. The first digit of the exit number represents the phase in which the exit is located -- 1 for the sort phase, 2 for the intermediate merge phase, and 3 for the final merge phase. The second digit represents the type of function your routine can perform at the exit.

EFFICIENCY CONSIDERATIONS

When you consider using sort/merge program exits, you should weigh the following factors:

- Your modification routines occupy main storage that would otherwise be available to the sort/merge program. Because its main storage is restricted, the program may need to execute extra intermediate merge phase passes. This, of course, increases sorting time.

- The execution of your routine adds time to the overall sort/merge program execution time. Later, in the description of exits, you will note that most of the exits give your routine(s) control once for each record until you pass

a "do not return" return code to the program. You should design your modification routines with this in mind.

To use the sort/merge program exits, you must associate your routines with the appropriate exits using the MODS control statement. (Refer to the topic "Defining the Sort or Merge" in Section 2.)

<u>Note</u>: If you use the 18K linkage editor with the minimum amount of sort/merge main storage, your routines are limited to 10 external references.

Possible Use for Exit	Sort						Intermediate Merge					Final Merge					Extract
	E11	E15	E16	E17	E18	E19	E21	E25	E27	E28	E29	E31	E35	E37	E38	E39	E61
Assignment	X						X					X					
Nmax Error			X														
Logical Record Change		X						X					X				
Control Field Change																	X
Opening Data Sets	X						X					X					
Closing Data Sets				X					X					X			
Read Error Routine					X					X					X		
Write Error Routine						X					X					X	

Figure 17. Summary of Functions Permitted at Sort/Merge Program Exits

BYPASSING THE LINKAGE EDITOR

To save execution time, you should design your routines so that they do not require linkage editing each time they are used in a sort/merge application. To avoid use of the linkage editor at sort/merge execution time, your routines must meet the following requirements:

* Each routine must be a load module on a partitioned data set (library). Its member name must be the same as its exit number, e.g., E16. The value s on the MODS statement that defines the routine must be the name of the DD statement that defines the library, e.g.,

  MODS E16=(E16,500,MYLIB,N)
  //MYLIB DD DSNAME=MYRTN,etc.

* Each routine must have only one entry point which is the module name.

* The routines cannot have external references.

* All routines must be on the same library or must be defined as a concatenated data set with one ddname.

You should code the parameter N on the MODS statement for each routine that meets the above requirements. This indicates that the routine was previously linkage edited and does not require further linkage editing.

If you use routines at assignment exits (E11, E21, and E31) that do not meet the requirements for bypassing the linkage editor, you can still save execution time by designing them for separate linkage editing. To be eligible for separate linkage editing, your assignment component routines must meet the following requirements:

- Each routine must be separate.

- The routines cannot contain external references.

- The routines can have several entry points, but one entry point must be the same as the exit number e.g., E11.

- The routine must be designed so that it can be overlaid after assignment time.

To indicate that the routine is eligible for separate linkage editing, code the parameter S for that routine on the MODS statement. If your routine opens data sets or communicates with running component routines, it will contain external references and therefore cannot be linkage edited separately.

When your routine does not meet the requirements for bypassing the linkage editor or for separate linkage editing, do not code a fourth parameter for that routine on the MODS statement. The routine is then linkage edited together with all other routines in its phase which do not meet the requirements. In any phase, you can mix routines that do not require additional linkage editing, routines that can be linkage edited separately, and routines that must be linkage edited together.

OPERATING CONSIDERATIONS

Each of your routines must be assembled or compiled as a separate program and placed either in a partitioned data set (library) or in the system input stream. The sort/merge general assignment phase includes the names and locations of your routines in the list of modules to be executed during each program phase. Your routines are loaded and executed with their associated program phase.

None of your routines may appear more than once in a program phase, but the same routine can appear in several phases. For example, you can use the same read error routine in all three phases, but not twice in any one phase. If a routine is to be used more than once and the routines are on SYSIN, you must supply a copy of the routine for each use.

Only one load module is allowed at each sort/merge program exit. If you need more than one routine at an exit, the routines must be assembled, compiled or linkage edited as one load module.

All your routines in a phase that require linkage editing can be placed in one partitioned data set member. The member must have an entry point for each of the routines you use. When routines are arranged in one member, their individual lengths specified on a MODS statement are not important, but the sum of the lengths must be the total length of the module. All but one length can be specified as zero, with the total member length specified for the remaining routine.

The exit routines must not be arranged to overlay each other. But if you want to build a single exit routine as an overlay structure, you must perform the linkage editing before the sort or merge and code N in the MODS statement.

ROUTINES IN THE SYSTEM INPUT STREAM

The routines that you place in the system input stream are copied into the SORTMODS data set; they then become input to the linkage editor. Under the MVT configuration, the entire contents of SYSIN, including control statements, is first moved to a system direct access data set. Sort/merge strips away the sort control cards and then copies your routines on SORTMODS.

When data follows your routines, it is also written on the system data set. When one of your routines opens SYSIN to read the data, it will start reading from the beginning of the SYSIN data set.

LINKAGE CONSIDERATIONS

Your routine must save and restore all general registers it uses at the modification exit. The general registers used by the sort/merge program for linkage and communication of parameters follow operating system conventions. The registers used are:

- General register 1 -- used to pass the address of a parameter list to the called routine.

- General register 13 -- contains the address of an area, set aside by the sort/merge program, in which your routine may save the contents of any general registers it needs for operation.

- General register 14 -- This register contains the address of the sort/merge program return point.

- General register 15 -- contains the address of your routine. Your routine can use it as a base register. General register 15 is also used as a return-code register whereby your routine communicates information to the sort/merge program.

The sort/merge program uses a CALL macro instruction expansion to enter your routines. You can also use the RETURN macro instruction to set return codes when multiple actions are available at an exit. You can use the SAVE macro instruction to save all general registers that the routine uses. If you save registers, you must also restore them. You can do this with the RETURN macro instruction.

All of your routines must contain an entry point defined by an ENTRY or CSECT statement. The name of the entry point must be the number of the associated sort/merge program exit.

Linkage Examples

The CALL macro instruction used by the sort/merge program to link to your routines is written as follows:

        CALL   E11

This macro instruction is expanded to form assembler language instructions and, when executed, places the return address in general register 14 and your routine's entry point address in general register 15. The sort/merge program has already placed the register save area address in general register 13.

Your routine for the sort phase assignment component exit could incorporate the following instructions:

```
        ENTRY    E11
          .
          .
          .
E11     SAVE     (5,9)
          .
          .
          .
        RETURN   (5,9)
```

This coding saves and restores the contents of general registers 5 through 9. The macro instructions are expanded into the following assembler language code:

```
        ENTRY    E11
          .
          .
          .
E11     STM      5,9,40(13)
          .
          .
          .
        LM       5,9,40(13)
        BR       14
```

If multiple actions are available at a sort/merge program exit, your routine sets a return code in general register 15 to inform the sort/merge program of the action it is to take. The following macro instruction could be used to return to the sort/merge program with a return code of 12 in general register 15:

```
        RETURN RC=12
```

(A full explanation of linkage conventions and the macro instructions discussed in this section can be found in the publication OS Data Management for System Programmers.)

Assignment Component Exits (E11, E21, E31)

PHASE IN WHICH USED:

    E11 -- Sort phase
    E21 -- Intermediate merge phase
    E31 -- Final merge phase

POSSIBLE USES OF ROUTINES: You might use routines at these exits to open data sets needed by your other routines in the associated phases, or to initialize your other routines.

RETURN CODES: None.

LINKAGE CONVENTIONS:

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge
CALL E11	ENTRY E11 E11 SAVE (5,9) [1] RETURN (5,9) [1]
[1]This coding saves and restores the contents of registers 5 through 9. You would save and restore whatever registers you use.	

FURTHER CONSIDERATIONS: These are the only three routines you can design for separate
linkage editing. Refer to the topic "Bypassing the Linkage Editor" earlier in this
section.

Running Component Exits

Each sort/merge program phase has a number of running component exits associated with it.
Many of these exits perform the same function in each of the program's three phases.
They are explained in the following text according to exit function.

RECORD CHANGE EXITS (E15, E25, E35)

The record change exits can be used to insert, delete, alter, or summarize records.

Exit E15

PHASE IN WHICH USED: Sort phase before any records are sorted.

POSSIBLE USES OF ROUTINE: Add records to the input data set, create the entire input
data set, delete records from the input data set, change records in the input data
set (except control fields). Use exit E61 for control field change.

INFORMATION SUPPLIED TO YOUR ROUTINE BY SORT/MERGE: Your routine at exit E15 is
executed each time a new record is brought into the sort phase. Sort/merge places
the address of a parameter list in general register 1. The parameter list contains
the address of the new record. The parameter list starts on a fullword boundary and
is one fullword long. The high order byte of the word is not used; it is represented
by xx in the figure below. The format of the parameter list is:

XX	Address of the new record

When sort/merge reaches the end of the input data set, it passes an address of zero
in the parameter list. If there are no records in the input data set, sort/merge
passes a zero address the first time it uses exit E15.

RETURN CODES: Your routine must pass one of the following return codes to the sort merge program informing it what to do with the record you have been examining or changing:

        0 -- Alter or no action
        4 -- Delete record
        8 -- Do not return
       12 -- Insert record

0 - No Action: If you want sort/merge to retain the record as is, place the address of the record in general register 1 and return to sort/merge with a zero return code.

0 - Alter Record: If you want to change the record before passing it back to sort/merge, your routine must move the record into a work area, perform whatever modification you desire, place the address of the modified record in general register 1, and return to sort/merge with a zero return code. If your routine changes record size, you must communicate that fact to the program on a RECORD statement. (See "Defining the Sort or Merge" in Section 2 and the publication OS Data Management for System Programmers for further information about the length indicator and the Record Descriptor Word.)

4 - Delete Record: If you want sort/merge to delete the record from the input data set, return with a return code of 4. You need not place the address of the record in general register 1.

8 -- Do Not Return: Sort/merge keeps returning to your routine until you pass a return code of 8. After that, the exit is closed and not used again during the sort/merge application. You need not place an address in general register 1 when you return with RC = 8. Unless you are inserting records after end-of-data set, you must pass a return code of 8 when sort/merge indicates end-of-data set by passing your routine a zero address in the parameter list.

12 -- Insert Record: If you want sort/merge to add a record to the input data set, before the record whose address was just passed to your routine, place the address of the record to be added in general register 1 and return to sort/merge with a return code of 12. Sort/merge then returns to your routine with the same record address as before so that your routine can insert more records at that point or alter the current record. You can make insertions after the last record in the input data set (after sort places a zero address in the parameter list). Sort/merge keeps returning to your routine until you pass a return code of 8.

LINKAGE CONVENTIONS: Linkage conventions for exit E15 are shown in the following table:

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge

LA   1,param
    CALL E15
    .
    .
    .
param DC   A(radrs)
	ENTRY E15
	    .
	    .
	E15 SAVE (5,9) [1]
	    .
	    .
	    LA 1,nwrec
	    RETURN (5,9), [1]
	           RC=x

[1]This coding saves and restores the contents of registers 5 through 9. You would save and restore whatever registers you use.
radrs refers to the record passed by the sort
nwrec refers to the record returned to the sort
x is the return code

RESTRICTIONS: If you ATTACH, LINK, or XCTL to the sort/merge program, and use exit E15, the sort/merge program ignores the SORTIN data set. Your E15 routine must pass all input records to the program by placing their addresses one by one into general register 1 and returning to sort/merge with RC = 12. When sort/merge returns to your routine after you have passed the last record, return to sort with RC = 8 indicating "do not return". Since exit E15 is associated with the sort phase, it cannot be used during a merge-only operation.

Exit E25

PHASE IN WHICH USED: Intermediate merge phase, after the records have been merged.

POSSIBLE USES OF ROUTINE: Change (except control fields) or delete records leaving the intermediate merge phase.

INFORMATION SUPPLIED TO YOUR ROUTINE BY SORT/MERGE: Your E25 routine is executed each time sort/merge prepares to place a record (except the first record in each sequence) in an intermediate merge output sequence. Sort/merge passes two record addresses to your routine. These are:

- The address of the record leaving the merge, which would normally follow the record in the output area.

- The address of a record in the output area.

In general register 1, sort/merge places the address of a parameter list that contains these two record addresses. The parameter list starts on a full word boundary and is two fullwords long. The high order bytes of both words are not used (contain zeros). The format of the parameter list is:

00	Address of Record Leaving Merge
00	Address of Record in Output Area

RETURN CODES: Your routine must pass one of the following return codes to the sort/merge program informing it what to do with the record leaving the merge:

    0 -- Alter or no action
    4 -- Delete record or summarize and delete

0 - No Action: If you want sort/merge to retain the record as is in the intermediate merge sequence, load the address of the record leaving the merge into general register 1 and return to sort/merge with a zero return code. The next time sort/merge transfers control to your routine, the record whose address you just passed will be the record in the output area.

0 - Alter Record: If you want to change the record (except its control field) before passing it back to sort/merge, move the record to a work area, make the change, place the address of the modified record in general register 1, and return to sort/merge with a zero return code.

4 - Delete Record: If you want to delete the record leaving the merge, return to sort/merge with a return code of 4. You need not place an address in register 1.

4 - Summarize and Delete: You can summarize records by changing the record in the output area and then deleting the record leaving the merge. Sort/merge then returns to your routine with a new record (leaving the same record in the output area so that you can summarize further.) If you want to perform summarization without deletion, you should do it at exit E35 rather than E25 because it is more efficient. The sort/merge program does not test for equal control fields before taking exit E25. If you want to summarize records with equal control fields, you must test the fields.

<u>LINKAGE CONVENTIONS</u>: Linkage conventions for exit E25 are shown in the following

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge

<pre>   LA   1,param
      CALL E25
      .
      .
param DC   A(rcara)
      DC   A(otara)</pre>
	<pre> ENTRY E25
	          .
	          .
	E25 SAVE (5,9) ˈ
	          .
	          .
	          LA 1,modrc
	          RETURN (5,9) ,ˈ
	                 RC=x</pre>

ˈThis coding saves and restores the contents of registers 5 through 9. You would save and restore whatever registers you use.
<u>rcara</u> refers to the record leaving the merge
<u>otara</u> refers to the record in the output area
<u>modrc</u> refers to the record returned to the merge
<u>x</u> is the return code

<u>RESTRICTIONS</u>: You must not retain status information in the exit routine; you must carry it in the records being merged. The entire intermediate merge phase (including your E25 exit routine) is reloaded into main storage for each intermediate merge phase pass when the balanced tape or balanced direct access sequence distribution techniques are used by the program. Your routine would not work properly when sort/merge chooses either of the balanced techniques, if it depended upon status information saved within it. Since exit E25 is associated with the intermediate merge phase, it cannot be used during a merge-only operation.

<u>Exit E35</u>

<u>PHASE IN WHICH USED</u>: Final merge phase after the records have been merged.

<u>POSSIBLE USES OF ROUTINE</u>: Add records to, delete records from, or change records in the output data set.

<u>INFORMATION SUPPLIED TO YOUR ROUTINE BY SORT/MERGE</u>: Your E35 exit routine is executed each time sort/merge prepares to place a record (including the first record) in the output area after the final merge. Sort/merge passes two record addresses to your routine. These are:

• The address of the record leaving the merge which would normally follow the record in the output area. (This address is zero at end-of-data set.)

• The address of a record in the output area. (This address is zero the first time your routine is entered because there is no record in the output area at that time. It will of course remain zero so long as you delete all records leaving the merge.)

Sort/merge also passes your routine a third parameter which is used to control sequence checking. In general register 1, sort/merge places the address of a parameter list that contains the two record addresses and the sequence check switch. The list is three full-words long and begins on a full-word boundary. The high order bytes of the first two words are not used. The format of the parameter list is:

XX	Address of Record Leaving Merge		
XX	Address of Record in Output Area		
	00	00	Sequence Check Switch - 00 or 04

The sort/merge program tests the sequence check switch before each record is written on the output data set. If the word contains all zeros, sort/merge performs a sequence check. If the low order byte of the word contains a 4, sort/merge does not perform a sequence check. This switch is initially set to zero. Your routine can set it and reset it as necessary. If your routine is altering control fields which would not collate properly in the output data set, it should set the low order byte of the switch to 4 to eliminate the sequence check for the records whose control fields have been changed.

RETURN CODES: Your routine must pass one of the following return codes to sort/merge informing it what to do with the record leaving the merge:

    0 -- Alter or no action
    4 -- Delete record
    8 -- Do not return
   12 -- Insert record

0 -- No Action: If you want the program to retain the record as is in the output data set, load the address of the record leaving the merge into general register 1 and return to sort/merge with a zero return code.

0 -- Alter Record: If you want to change the record before having it placed in the output data set, move the record to a work area, make the change, load the address of the modified record into general register 1, and return to sort/merge with a zero return code. If you change record size, you must communicate that fact to sort/merge on a RECORD statement.

4 -- Delete Record: Your routine can delete the record leaving the merge by returning to sort/merge with a return code of 4. You need not place an address in general register 1.

8 -- Do Not Return: Sort/merge keeps returning to your routine until you pass a return code of 8. After that, the exit is closed and not used again during the sort/merge application. When you return with RC = 8, you need not place an address in general register 1. Unless you are inserting records after end-of-data set, you must pass RC = 8 when sort/merge indicates end-of-data set by passing your routine zero as the address of the record leaving the merge.

12 -- Insert Record: If you want to add a record to the output data set before the record leaving the merge, place the address of the new record in general register 1 and return to sort/merge with a return code of 12. Sort/merge returns to your routine with the same addresses as before so that you can make more insertions at that point, or delete the record leaving the merge, etc. Sort/merge does not perform a sequence check on records that you insert unless you delete the record leaving the merge and insert a record to replace it. If your new record will not collate properly, set the sequence check switch to 4 to eliminate the sequence check for that record.

Summarize Records: You can summarize records in the output data set by changing
the record in the output area and then, if you desire, deleting the record leaving
the merge. Sort/merge returns to your routine with the address of a new record
leaving the merge and leaves the same record in the output area, so that you can
summarize further. If you do not delete the record leaving the merge, that record
takes the place of the record in the output area and sort/merge returns with the
address of a new record leaving the merge. As with exit E25, sort/merge does not
check for equal control fields.

LINKAGE CONVENTIONS: Linkage conventions for exit E35 are shown in the following
table:

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge

LA   1,param
     CALL E35
     .
     .
param DC   A (rcara)
     DC   A (otara)
     DC   A (0)
	ENTRY E35
	E35 SAVE (5,9) ¹
	     .
	     .
	E35 MVI 11(1) ,
	     . X'04'
	     .
	     LA 1,nwrec
	     RETURN (5,9) ,¹
	           RC=x

¹This coding saves and restores the contents of registers 5 through 9. You would save and restore whatever registers you use. rcara refers to the record leaving the merge otara refers to the record in the output area nwrec refers to the record returned to the merge x is the return code	

RESTRICTIONS: If you ATTACH, LINK, or XCTL to the sort/merge program and use exit
E35, the sort/merge program ignores the SORTOUT data set. Your E35 routine must
dispose of all the output records by writing them out on a data set (you must
supply a DD statement defining that data set), and returning to sort/merge with RC
= 4. When sort/merge returns to your routine after you have disposed of the last
record, return to sort with RC = 8 indicating "do not return."

NMAX EXIT (E16)

PHASE IN WHICH USED: Sort phase.

POSSIBLE USES OF ROUTINE: You would use a routine at this exit to decide what to
do if sort exceeds its calculated estimate of the number of records it can handle
for a given amount of main storage and intermediate storage.

RETURN CODES: Your routine can choose among three actions, and must use one of
the following return codes to communicate its choice to sort/merge:

     0 -- Sort current records only.
     4 -- Try to sort additional records.
     8 -- Terminate the program.

0 -- Sort Current Records Only: If you want sort/merge to continue with only that
part of the input data set it estimates it can handle, return with RC = 0.
(Message IER054I contains the number of records that sort is continuing with. You
can sort the remainder of the data set on another run, using the SKIPREC operand
on the SORT statement to skip over the records already sorted. Then you can merge
the two sort outputs to complete the operation.)

4 -- Try to Sort Additional Records: If you want sort/merge to continue with all of the input data set, return with RC = 4. (Enough space may be available for the program to complete processing. If enough is not available, the program generates a message and terminates. Refer to "Further Considerations" below.)

8 -- Terminate the Program: If you want sort/merge to terminate, return with RC = 8. The job steps following the sort step are executed.

LINKAGE CONVENTIONS: Linkage conventions for this exit appear in the following table:

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge

CALL E16	ENTRY E16
	      .
	      .
	E16 RETURN RC=x

x is the return code	

FURTHER CONSIDERATIONS: For variable-length input records, sort/merge calculates Nmax using the maximum record length. Therefore, Nmax tends to be lower than the actual number of records the program can handle. If the maximum record length is much larger than the average record length, Nmax is considerably lower than the number of records the program can handle.

Sort/merge can calculate Nmax very accurately for fixed-length records. When Nmax is reached, usually little additional space remains.

If the input data set has no natural ordering, and if direct access devices (balanced technique only) are used for intermediate storage, Nmax tends to be larger than the number of records the program can handle.

Nmax is recalculated during the sort phase (balanced direct access technique only) and the final value may be less than the original estimate.

Note: Nmax for magnetic tape is calculated for 2400-foot tapes. For shorter tapes the figure should be reduced proportionately.

EXITS FOR CLOSING DATA SETS (E17, E27, E37)

Your routines at these exits are executed once at the end of the phase with which they are associated. They can be used to close data sets used by your other routines in the phase or to perform any housekeeping functions for your routines.

PHASE IN WHICH USED:

    E17 Sort phase
    E27 Intermediate merge phase
    E37 Final merge phase

LINKAGE CONVENTIONS: The linkage conventions used with these exits appear in the following table:

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge
CALL E17	ENTRY E17 . . E17 CLOSE . . RETURN

READ/WRITE ERROR ROUTINES

You can use the six read/write error exits to incorporate your own or your installation's I/O error recovery routines into the sort/merge program. When the sort/merge program encounters an uncorrectable I/O error, it passes the same parameters as those passed by QSAM. The following information is passed to your synchronous error routine:

General Register 0: This register always contains X'10' in the high-order byte. The remaining three bytes contain the address of the input/output block (IOB) associated with the error, as follows:

X'10'	IOB address

General Register 1: The high-order byte of this register always contains zeros. The remaining three bytes contain the address of the data control block (DCB) associated with the error, as follows:

00	DCB address

General Register 14: This register contains the return address of the sort/merge program.

General Register 15: This register contains the address of your error routine.

   Your read and write error routines can reside on a library, or can be placed in SYSIN. Your library or SYSIN routines are brought into main storage with their associated phases. (The E28 and E29 routines are reloaded for each pass of the intermediate merge phase.)

Read Error Exits (E18, E28, E38)

PHASE IN WHICH USED:

     E18 -- Sort phase
     E28 -- Intermediate merge phase
     E38 -- Final Merge phase

POSSIBLE USE OF ROUTINES: Your routines at these exits can pass a parameter list containing the specifications for three data control block fields -- SYNAD, EXLST, and EROPT -- to the sort/merge program. Your E18 exit routine can pass a fourth DCB field -- EODAD -- to sort/merge.

98

Your routines are entered first during the assignment component of each phase so that the sort/merge program can obtain the parameter lists. The routines are entered again during the running components at the points indicated in the parameter lists. For example, if you choose the EXLST option for your E18 routine, sort/merge enters your E18 routine during the execution of the sort phase assignment component. Sort picks up the parameter list, including the EXLST address. During the running component, sort/merge enters your routine at the EXLST address when the data set is opened.

INFORMATION YOUR ROUTINE PASSES TO SORT/MERGE: Your routine passes the DCB fields to sort/merge in a parameter list, the address of which it places in general register 1 before returning to the sort/merge program. The parameter list must begin on a fullword boundary and be a whole number of fullwords long. The high order byte of each word must contain a character code that identifies the parameter. One or more of the words can be omitted. A word of all zeros marks the end of the list. The format of the list is:

Byte 1	Byte 2	Byte 3	Byte 4
01	SYNAD field		
02	EXLST field		
03	0	0	EROPT code
04	EODAD field		
00	0	0	0

A full description of these DCB fields is in the publication Data Management for System Programmers. A brief description of these fields follows:

SYNAD: This field contains the address of your read synchronous error routine. This routine is entered only after the operating system has tried unsuccessfully to correct the error. The routine must be assembled as part of your E18, E28, or E38 exit routine. When the routine receives control, it must not store registers in the save area pointed to by general register 13.

EXLST: This field contains the location of a list which contains pointers to your routines that you want used to check labels and perform other functions not done by data management. The list and the routines to which it points should be included in your read error routine.

EROPT: The EROPT code is a means whereby you can specify what action sort/merge should take if an uncorrectable read error is encountered. The three possible actions and the codes associated with them are:

    X'80' -- Accept the record (block) as is
    X'40' -- Skip the record (block)
    X'20' -- Terminate the program

If you include this parameter in the DCB field list, you must place one of the above codes in the low-order byte of the word. Bytes 2 and 3 of the word must contain zeros.

When you use the EROPT option, the SYNAD field (and the EODAD field of exit E18) must contain either of the following:

- The address of your read synchronous error routine (or end-of-file routine in the EODAD field). These must be addresses within your exit routine.

- If you do not provide a read synchronous error routine or an end-of-file routine, the fields must contain X'01' in byte 4; bytes 2 and 3 must contain zeros. You can use the instruction DC AL3(1) to set up the field.

<u>EODAD</u>: This field is the address of your end-of-file routine. You can specify
this parameter at exit E18 only. If you specify it, your end-of-file routine must
be included in your exit routine. The end-of-file routine is used only for the
SORTIN data set.

<u>LINKAGE CONVENTIONS</u>: Linkage conventions for these exits are shown in the
following table:

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge		
CALL E18		ENTRY	E18
		.	
		.	
	E18	LA	1,parm
		RETURN	
		CNOP	0,4
	parm	DC	X'01'
		DC	AL3 (ser)
		DC	X'02'
		DC	AL3 (1st)
		DC	X'03'
		DC	XL3 (x)
		DC	A (0)
		.	
		.	
	ser	error routine	
		.	
		.	
	1st	address list	

<u>ser</u> refers to the read synchronous error routine
<u>1st</u> refers to the EXLST address list
<u>x</u> is EROPT code

<u>Write Error Exits (E19, E29, E39)</u>

<u>PHASE IN WHICH USED</u>:

    E19 -- Sort phase
    E29 -- Intermediate merge phase
    E39 -- Final merge phase

<u>POSSIBLE USES OF ROUTINE</u>: Your routines at these exits can pass a parameter list
containing the specifications for two DCB fields -- SYNAD and EXLST -- to the
sort/merge program.

    Your routines are entered first during the assignment component of each phase
so that the sort/merge program can obtain the parameter lists. The routines are
entered again during the running components at the points indicated by the options
in the parameter lists.

INFORMATION YOUR ROUTINE PASSES TO SORT/MERGE: Your routine passes the DCB fields
to sort/merge in a parameter list, the address of which it places in general
register 1 before returning to the sort. The list must begin on a fullword
boundary and must be a whole number of fullwords long. The high-order byte of
each word must contain a character code that identifies the parameter. Either
word can be omitted. A word of all zeros indicates the end of the list. The
format of the list is:

Byte 1	Byte 2	Byte 3	Byte 4
01	SYNAD field		
02	EXLST field		
00	0	0	0

A full description of these DCB fields can be found in the publication OS Data
Management for System Programmers. A brief description follows:

SYNAD: This field contains the location of your write synchronous error routine.
This routine is entered only after the operating system has unsuccessfully tried
to correct the error. It must be assembled as part of your exit routine.

EXLST: The EXLST field contains the location of a list that contains pointers to
your routines that you want used to check labels and perform other functions not
done by data management. This list and the routines to which it points must be
included as part of your exit routine.

LINKAGE CONVENTIONS: Linkage conventions for these exits are shown in the
following table:

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge
CALL E19	ENTRY E19 . . E19 LA 1,parm RETURN CNOP 0,4 parm DC x'01' DC AL3(ser) DC x'02' DC AL3(1st) DC A(0) . . ser error routine . . 1st address list
ser refers to the write synchronous error routine 1st refers to the EXLST address list	

CONTROL FIELD MODIFICATION EXIT (E61)

You can use a routine at this exit to lengthen, shorten, or alter any control
field within a record. The E option for the s parameters on the SORT or MERGE
control statement must be specified for control fields changed by this routine.
(Refer to the topic "Defining the Sort or Merge" in Section 2.)

PHASE IN WHICH USED: Your routine is loaded with the running portion of each phase and is executed whenever sort/merge encounters a control field specified by the E option.

POSSIBLE USES OF ROUTINE: Your routine can normalize floating point control fields or change any other type of control field in any way that you desire. You should be familiar with the standard data formats used in System/360 before modifying control fields.

INFORMATION SUPPLIED TO YOUR ROUTINE BY SORT/MERGE: Sort/merge places the address of a parameter list in general register 1. The list begins on a fullword boundary and is two fullwords long. It contains the number (in hexadecimal) of the control field in the low-order byte of the first word, and the address of the control field in the three low-order bytes of the second word as follows:

Byte 1	Byte 2	Byte 3	Byte 4
00	00	00	01-40
00	Control Field Address		

   Before it passes your routine the control field address, sort/merge moves the control field to an extract area, an area apart from the record. Your routine, in effect, changes an image of the control field and not the control field itself. The format of the field is unchanged.

   For all fields except binary, the number of bytes sort/merge passes to your routine is equal to the length specified in the m parameters of the SORT or MERGE statement. All binary fields passed to your routine, however, contain a whole number of bytes. If a binary field does not begin and end on a byte boundary, sort/merge pads it with zeroes at beginning and/or end.

   Your routine cannot physically change the length of the control field. If you increase the length for collating purposes, you must specify that length in the m parameter of the SORT or MERGE statement. If you must shorten the control field, you must pad the field to the specified length before returning it to sort/merge. The field your routine returns to sort/merge must contain the same number of bytes as when it was received.

   Sort/merge collates the modified control field in absolute ascending order.That is, if all major control fields are equal, the "winner" record chosen by the compare network is the one in which the modified copy of a control field is lowest in the collating order when compared logically with the modified copy of the same field in other records.

LINKAGE CONVENTIONS: Linkage conventions for exit E61 are shown in the following table:

Code Sort/Merge Uses to Enter Your Routine	Code Your Routine Uses to Return to Sort/Merge	
CALL E61		ENTRY E61
		.
		.
	E61	SAVE (5,9) [1]
		.
		.
		RETURN (5,9) [1]
[1]This coding saves and restores the contents of registers 5 through 9. You would save and restore whatever registers you use.		

EXIT	USE	PHASE
E11	Assignment Opening Data Sets	Sort
E15	Record Change	Sort
E16	NMAX	Sort
E17	Closing data sets	Sort
E18	Read Error Routine	Sort
E19	Write Error Routine	Sort
E21	Assignment Opening Data Sets	Int. Merge
E25	Record Change	Int. Merge
E27	Closing Data Sets	Int. Merge
E28	Read Error Routine	Int. Merge
E29	Write Error Routine	Int. Merge
E31	Assignment Opening Data Sets	Final Merge
E35	Record Change	Final Merge
E37	Closing Data Sets	Final Merge
E38	Read Error Routine	Final Merge
E39	Write Error Routine	Final Merge
E61	Change Control Field Contents	All Phases

Register Conventions

REGISTER	USE
1	Sort/Merge places address of parameter list in this register.
13	Sort/Merge places address of a save area in this register. Area may be used to save contents of registers used by routine.
14	Contains address of sort/merge return point.
15	Contains address of your routine. May be used as base register for routine. This register is also used by routine to pass return codes to sort/merge.

.

Section 4: Efficient Program Use

Once you become familiar with the basic functions of the sort/merge program, you will be concerned with program efficiency -- how to get a faster sort or merge. In this section the following subjects involving program efficiency are discussed:

- Information you can supply to the sort/merge program to optimize its operation.

- Intermediate storage assignment for optimum performance.

- Multiprogramming efficiency considerations.

- System generation options and requirements.

Supplying Information to the Program

The information you give the sort/merge program about the application it is to perform helps the sort and merge phases to produce a fast, efficient sort or merge. When you do not supply information such as data set size and record format, the program must make assumptions, which, if incorrect, lead to inefficiency.

DATA SET SIZE

The most important information you can give the program is an accurate data set size using the SIZE parameter of the SORT or MERGE statement. If you know the exact number of records in the input data set, use that number as the value of the SIZE parameter. If you do not know the exact number, estimate it as closely as you can.

When the sort/merge program has accurate information about data set size, it can make the most efficient use of both main storage and intermediate storage.

BLOCKING INPUT RECORDS

Sort performance is improved if you block input records as this saves time in reading your data set.

RECORD FORMAT

When your input data set consists of variable length records, use the RECORD statement to supply maximum, minimum, and modal (most frequent) lengths to the sort/merge program. This information allows the program to calculate the optimum sort.

Intermediate Storage Assignment

If you can, avoid assigning the bare minimum amount of intermediate storage for a given application. When a small amount of intermediate storage is assigned to the program, more intermediate merge phase passes are necessary because only a small number of record sequences can be merged at one time. Naturally, these extra passes increase sorting time.

Likewise, when the program has only a small amount of main storage to operate in, more intermediate merge phase passes are necessary because only a small number of records can be sorted internally and more sequences are produced.

The sort/merge program operates efficiently when at least two selector channels are available. A tape switching device also improves program performance, if the device is connected so that two channel paths exist between each device and the central processing unit that is running the sort/merge program.

ASSIGNING DIRECT ACCESS INTERMEDIATE STORAGE

Program performance is improved if you use devices, storage areas, and channels efficiently. If you use UNIT=2311, 2314, or 2301 on the DD statements that define intermediate storage data sets, the program assigns areas, and some optimization occurs automatically. But maximum performance is achieved if you follow these recommendations:

* Use as many physical devices as you have available. (If you place more than one intermediate storage data set on a disk, place them as close together as possible to minimize access arm movement.)

* Use channel overlap whenever you can.

* On 2311 and 2301 and using the balanced technique on 2314, assign as few data sets as possible. (You need at least three. Three large data sets are more efficient than six smaller ones.) Using the criss-cross technique on 2314, assign as many data sets as possible, (17 maximum) but not more than one for each device.

* Assign data sets of similar sizes.

Assigning more than three intermediate storage data sets (the minimum number) on a 2311 disk or a 2301 drum decreases program efficiency unless you assign the data sets to different devices. Sometimes you may need the maximum (six for the 2311 and 2301) number of data sets to handle a large input data set. To preserve efficiency, assign them on separate physical devices.

For example, if a 100-track area is available on each of three 2311 disk drives, you can handle more records if you define six data sets, each 50 tracks long, two on each device, but you decrease efficiency. If the size of the input data set permits, you can increase efficiency by defining fewer areas. For maximum efficiency, define three 100-track areas, each on a different device.

If your intermediate storage is on 2314, you can obtain maximum efficiency by assigning one data set per access arm. Also, efficiency decreases as the size of your input data set approaches sort capacity.

If you use channel overlap program performance is improved because the program can read input while writing output, etc.

Figure 18 shows a method for specifying channel overlap. The SEP parameter on the SORTWK01 DD statement requests that the operating system assign that data set to a channel other than the channel assigned to the SORTIN data set. The AFF parameter on the SORTWK03 and SORTOUT DD statements requests that the SORTWK03, and SORTOUT data sets, also be on a channel that is different from SORTIN. The channel assigned to SORTWK02 and SORTWK04 is not necessarily the same as the one assigned to SORTIN.

The operating system will honor your channel assignment requests when the necessary channel and device resources are available. If the requests cannot be filled, the system assigns channels according to the resources it has. Therefore, specifying channel overlap will never impair performance.

```
//SORTIN     DD    DSNAME=INPUT,VOLUME=SER=000001,UNIT=2311,DISP=(OLD,KEEP),     X
//                 DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SORTWK01 DD      DSNAME=WORK1,VOLUME=SER=000002,UNIT=2311,                      X
//                 SEP=SORTIN,SPACE=(TRK,(15),,CONTIG)
//SORTWK02 DD      DSNAME=WORK2,VOLUME=SER=000003,UNIT=2311,                      X
//                 SPACE=(TRK,(15),,CONTIG)
//SORTWK03 DD      DSNAME=WORK3,VOLUME=SER=000004,UNIT=2311,                      X
//                 AFF=SORTWK01,SPACE=(TRK,(15),,CONTIG)
//SORTWK04 DD      DSNAME=WORK4,VOLUME=SER=000005,UNIT=2311,                      X
//                 SPACE=(TRK,(15),,CONTIG)
//SORTOUT    DD    DSNAME=OUTPUT,VOLUME=SER=000006,UNIT=2311,DISP=(NEW,KEEP),     X
//                 DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),                          X
//                 AFF=SORTWK01,SPACE=(TRK,(50),,CONTIG)
```

Figure 18. DD Statements Illustrating Channel Overlap

ASSIGNING TAPE INTERMEDIATE STORAGE

You can use the timing tables in the publication OS Sort/Merge Timing Estimates as
guide lines for assigning tape intermediate storage.

Multiprogramming the Sort/Merge Program

You should consider the following factors when you execute the sort/merge program
with other programs:

 • The sort/merge program may use many I/O devices for input, output, and
   intermediate storage. You should assign it a relatively high priority to be
   sure that it gets control of the central processing unit frequently and does
   not tie up the I/O devices while it waits for CPU time.

 • The sort/merge program tends to be I/O bound. Therefore, you should
   multiprogram the sort with programs that are CPU bound.

 • When a single task attaches two or more sort applications by ATTACH, LINK, or
   XCTL, you must modify the standard DD names (SORTIN, SORTOUT, etc.) so that
   they are unique. Do this by specifying four letters in the parameter list
   passed to the sort/merge program. These characters replace the letters SORT
   in the references to standard DD names in sort/merge program modules. (For
   more information, see the topic "Passing Parameters to the Sort" in Section
   2.)

System Generation Options and Requirements

When the operating system for your installation is generated, certain sort/merge
facilities may be included; others may not be. You should be aware of what is
available at your installation. The following list is a summary of the sort/merge
facilities that can be included when the program is generated:

 • Sort or merge fixed-length records.

 • Sort or merge variable-length records.

 • Sort or merge records over 256 bytes long.

 • Operate with any or all allowable intermediate storage devices. (Only one
   type can be used for a specific sort run.)

- Sort or merge multiple control fields.

- Use sort/merge program exits.

- Print non-critical program-generated messages.

- Use (a specific number) of bytes of main storage as a maximum for sort/merge execution.

Selecting only the required program facilities conserves library space. If you attempt to execute an option that was not selected, the program terminates abnormally. System generation is described in the publication OS Sysgen.

LIMITING MAIN STORAGE

If the amount of main storage to be used by the sort/merge program was not specified when the system was generated, the program assumes it may use 12,000 bytes. A further 3500-5000 bytes will be needed for system functions. For reasonably efficient execution of the program 45,000 bytes of main storage are needed; in general, performance improves as more main storage is made available.

The maximum amount of main storage that can be made available to the program can be determined by subtracting the amount of storage required by system functions from the total amount available. The amount of main storage required for the execution of various operating components is given in the publication OS Storage Estimates. The publication OS Sysgen gives a formula for calculating the maximum amount of main storage.

On an execution by execution basis, you can change two of the system generation specifications: main storage size and types of messages printed.

ALTERING THE MAIN STORAGE ALLOCATION

You can override the amount of main storage specified at system generation time by using the PARM field of the EXEC statement. Write the field as follows:

    PARM='CORE=xxxxxx'

where xxxxxx is the amount of main storage in bytes that you want to operate with. xxxxxx cannot be less than 12,000, and at this value, some combinations of I/O devices and record lengths make a successful sort impossible. A formula for determining the maximum possible core size is given in the section describing the SORTMERG macro in the publication OS Sysgen. For MVT, the region size must be bigger than the sort size. Region size should be approximately 1.2 times the sort size + 8K. The main storage value is changed only for the current job step; afterwards, the value reverts to the one specified at system generation time.

Changing the main storage allocation is useful when you are running a sort/merge application in a multiprogramming environment. By reducing the amount of main storage allocated, you impair sort/merge performance so that other programs can have the storage they need to operate simultaneously. By increasing the allocation, you can run large sort/merge applications efficiently at the expense of other jobs sharing the multiprogramming environment.

ALTERING THE MESSAGE SPECIFICATION

You can override the message option selected at system generation by using the PARM field of the EXEC statement. Write the field as follows:

    PARM='MSG=xx'

where xx is a two-character code that specifies what kind of messages you want printed and where you want them to appear.

NO   means that you want no messages to be printed.

CC   means that you want critical messages only to be printed and you want them to appear on the console.

CP   means critical messages only and that you want them to appear on the printer.

AC   means that you want all messages (critical and informational) printed on the console .

AP   means that all messages are to be printed on the printer.

   The time factor involved in printing messages is relatively small. The printer is faster than the console so you save a few seconds by specifying CP or AP rather than CC or AC.

The following terms and phrases are defined as they are used in this publication.

ascending sequence: A sequence of records such that the control word of each successive record collates equal to or greater than that of the preceding record.

assignment component: A sort/merge program component that establishes the basic constants needed for program execution and initializes running components for a specific application.

block: A group of contiguous data read or recorded by an I/O device as one unit.

collating sequence: A predetermined sequence into which data can be sorted or merged.

control field: A group of contiguous data within a record that forms all or part of a control word.

control word: A group of control fields used to order records according to the collating sequence during a sort or merge.

descending sequence: A sequence of records such that the control word of each successive record collates equal to or less than that of the preceding record.

input data set: The data set (or data sets) used as input to the sort/merge program.

intermediate storage data set: A partially sequenced data set that is either input to or output from an intermediate merge phase pass.

major control field: The control field that is most significant in determining the collating sequence of a record.

merge: The process used to form one sorted sequence of records from two or more previously sorted sequences. Also, a program or routine that performs this function.

merge pass: The passing of all the records used as input to the sort/merge through a program phase which merges previously sorted sequences into fewer, longer sequences.

minor control field: A control field which is less significant than the major control field in determining the collating sequence

of a record. Successive minor control fields are considered to be in decreasing order of significance.

modal length: The record length that occurs most frequently in a variable-length record data set used as input to the sort/merge program.

nmax: The estimated maximum number of records of a given length that can be sorted using a given amount of intermediate storage.

output data set: The sequenced data set which is produced by a sort/merge program execution.

phase: A portion of the sort/merge program that is designed to perform one of the following functions: definition, optimization, sorting, intermediate merging, or final merging.

program exit: A place in the executable code of the sort/merge program component at which a user-written routine may be given control to perform various functions.

record: A group of contiguous characters which is processed as a unit by the sort/merge program.

running component: A sort/merge program component that performs a sorting or merging operation on the data set used as input to the program. Running components are initialized by assignment components.

sequence: A group of records that have been collated into a predesignated order.

sequence distribution technique: One of several methods used by the sort/merge program to combine previously sorted sequences of records into fewer, longer sequences.

sort: The process used to collate the records in a data set of unknown order. Also, a program or routine that performs this function.

sort blocking factor: The blocking factor used by the sort/merge program for intermediate storage data sets.

user-written routine: A routine written by the user to perform various functions at a sort/merge program exit.

Appendix A: Summary of How to Use the Sort/Merge Program

The following is a summary of what you need to do to use the sort/merge program:

- Prepare sort/merge control statements defining the sorting or merging application. (Refer to the topic "Defining the Sort or Merge" in Section 2.)

- For a sorting application, determine the amount of intermediate storage the sort/merge program will need for your data set. (Refer to the topic "Determining Intermediate Storage Requirements" in Section 2.)

- Prepare job control language statements to accompany the sort/merge statements. (Refer to the topic "Required Job Control Language Statements" in Section 2.)

The fold-out chart at the back of this manual shows the three points mentioned above in greater detail.

The chart does not cover the following points:

- EXEC statement PARM field options: forcing a sequence distribution technique. (Refer to "Sequence Distribution Techniques" in Section 1 for descriptions of the techniques; and "Job Control Language for Sort/Merge" in Section 2 for how to code the option), message option (refer to "Job Control Language for Sort/Merge" in Section 2), core value option (refer to "Job Control Language for Sort/Merge in Section 2).

- The checkpoint option. (Refer to "Defining the Sort or Merge" in Section 2 for how to select the option, and "Job Control Language for Sort/Merge" for information on the required SORTCKPT DD statement.)

- Achieving maximum sort/merge efficiency. (Refer to "Section 4: Efficient Program Use".)

Appendix B: Considerations for MVT Users—Summary

REGION SIZE

The SORT cataloged procedure requests a region size of 98K. The SORTD cataloged procedure requests 26K.

   A formula for determining region size is given in Introduction: "Determining Region Size".

OPTIONAL CHARACTERS FOR DD NAMES

If a task initiates two or more sort/merge applications via ATTACH, LINK, or XCTL, this option must be selected. It is discussed in the topic "Passing Parameters to the Sort" in Section 2.

ALTERING THE MAIN STORAGE ALLOCATION

The amount of main storage assigned to sort/merge at system generation can be changed. It can be temporarily increased to improve sort/merge preformance or temporarily decreased to permit other programs to obtain main storage. Refer to "Altering the Main Storage Allocation" in Section 4 for further details.

OTHER

Refer to "Multiprogramming the Sort/Merge Program" in Section 4.

Appendix C: Standard System/360 Operating System Collating Sequence

The following table shows the collating sequence for character and unsigned decimal data. The bit configuration shown is EBCDIC. The collating sequence is based on the EBCDIC representation of the graphic and ranges from low (00000000) to high (11111111). The bit configurations which do not correspond to graphics (e.g., 0 - 73, 81 - 89, etc.) are not shown. Some of these correspond to control commands for the printer and other devices.

Packed decimal, zoned decimal, fixed-point, and normalized floating-point data is collated algebraically; i.e., each quantity is interpreted as having a sign.

Collating Sequence	Bit Configuration	Graphic	Meaning
	00000000		
•			
•			
74	01001010	¢	Cent sign
75	01001011	.	Period, decimal point
76	01001100	<	Less than sign
77	01001101	(	Left parenthesis
78	01001110	+	Plus sign
79	01001111	|	Vertical bar, Logical OR
80	01010000	&	Ampersand
•			
•			
90	01011010	!	Exclamation point
91	01011011	$	Dollar sign
92	01011100	*	Asterisk
93	01011101	)	Right parenthesis
94	01011110	;	Semi colon
95	01011111	¬	Logical not
96	01100000	-	Minus, hyphen
97	01100001	/	Slash
•			
•			
107	01101011	,	Comma
108	01101100	%	Percent sign
109	01101101	_	Underscore
110	01101110	>	Greater than sign
111	01101111	?	Question mark
•			
122	01111010	:	Colon
123	01111011	#	Number sign
124	01111100	@	At sign
125	01111101	'	Apostrophe, prime
126	01111110	=	Equals sign
127	01111111	"	Quotation marks
•			
129	10000001	a	
130	10000010	b	
131	10000011	c	
132	10000100	d	
133	10000101	e	
134	10000110	f	
135	10000111	g	
136	10001000	h	
137	10001001	i	
•			
•			
145	10010001	j	
146	10010010	k	

Collating Sequence	Bit Configuration	Graphic	Meaning
147	10010011	l	
148	10010100	m	
149	10010101	n	
150	10010110	o	
151	10010111	p	
152	10011000	q	
153	10011001	r	
.			
.			
162	10100010	s	
163	10100011	t	
164	10100100	u	
165	10100101	v	
166	10100110	w	
167	10100111	x	
168	10101000	y	
169	10101001	z	
.			
.			
193	11000001	A	
194	11000010	B	
195	11000011	C	
196	11000100	D	
197	11000101	E	
198	11000110	F	
199	11000111	G	
200	11001000	H	
201	11001001	I	
.			
.			
209	11010001	J	
210	11010010	K	
211	11010011	L	
212	11010100	M	
213	11010101	N	
214	11010110	O	
215	11010111	P	
216	11011000	Q	
217	11011001	R	
.			
.			
226	11100010	S	
227	11100011	T	
228	11100100	U	
229	11100101	V	
230	11100110	W	
231	11100111	X	
232	11101000	Y	
233	11101001	Z	
.			
.			
240	11110000	0	
241	11110001	1	
242	11110010	2	
243	11110011	3	
244	11110100	4	
245	11110101	5	
246	11110110	6	
247	11110111	7	
248	11111000	8	
249	11111001	9	

Appendix D: Sort/Merge Messages

The sort/merge program generates two kinds of messages -- those which result from
critical error conditions and those which give information about the program's
operation. The printing of either all messages or only critical messages can be
specified at system generation. The messages can either be printed on a printer
or at the appropriate console.

When a system includes multiple console support, all messages except IER061A
are directed to the system master console. IER061A is directed to the tape, DASD,
or unit record pool console to which it applies. If an output error occurs on the
printer, the IER061A message is directed to the console with Routing Code = 11
independent of the message option specified.

The message options set up at system generation can be overridden on a job step
by job step basis by coding the MSG parameter in the PARM field of the EXEC
statement. Refer to the topic "EXEC Statement" in "Section 2: How to Use the
Sort/Merge Program", for a complete discussion of the MSG parameter.

The sort/merge program analyzes control statements in two stages. Stage 1
analyzes the general format of control statements. Stage 2 analyzes the
information contained in the sort/merge control statements and job control
language statements. Stage 2 checks for sort syntax and contents errors. Each
statement is scanned for errors. The first error detected stops the scan for that
statement. The program prints a message and continues the scan on successive
statements.

When the program encounters a critical error in either stage, it prints a
message and continues to analyze control information until the current stage is
completed, then the program terminates. Thus, if a critical error is found in
Stage 1, the program terminates at the end of Stage 1; if the error is encountered
in Stage 2, the program terminates at the end of Stage 2. The system action that
results from encountering a critical control information error is described in the
messages as either "Stage 1 termination" or "Stage 2 termination".

Component Name	IER
Program Producing Message	Sort/merge program.
Audience and Where Produced	For programmer: SYSPRINT data set or console (system generation option).
Message Format	IERnnns - text
	nnn
	Message serial number.
	s
	Severity code:
	A Error message; programmer action is required.
	I Information message; no programmer action is required.
	text
	Message text.
Comments	None.

IER001A - COL 1 OR 1-15 NOT BLANK

Explanation: Critical. Column 1 of a sort/merge control statement is not blank, or columns 1 through 15 of a sort/merge continuation card are not blank.

System Action: Stage 1 termination.

Programmer Response: Probable user error. Check control statements for nonblank characters in column 1, and continuation cards for nonblank characters in columns 1 through 15. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER002A - EXCESS CARDS

Explanation: Critical. This message is generated for one of four reasons:
- More than 33 control cards are supplied to the sort/ merge program.
- A sort/merge control statement type appears more than once. (For example, there is more than one SORT statement.)
- The control statements passed to the sort/merge program during an ATTACH, LINK, or XCTL operation contain more information than is allowed for the statements passed.
- A control statement occupies too many cards. The maximum number of cards for

a control statement is as follows:
```
SORT,MERGE,MODS    20 cards
RECORD              6 cards
END                 1 card
```

System Action: Stage 1 termination. The program does not analyze control cards above the 33 limit or duplicate type statements. If the sort was activated by an ATTACH, LINK, or XCTL, no information is processed.

Programmer Response: Probable user error. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER003A - NO CONTIN CARD

Explanation: Critical. A continuation card has been indicated by a nonblank character in column 72 of the previous card and no card follows.

System Action: Stage 1 termination.

Programmer Response: Probable user error. Check for a keypunching error, an overflow of parameters into column 72, or a missing continuation card. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the

EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
was included for the failing job step.
- Have the associated job stream and
master console log available.
- Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER004A - INVALID OP DELIMITER

Explanation: Critical. A control
statement ends with a comma or other
incorrect delimiter.

System Action: Stage 1 termination.

Programmer Response: Probable user error.
Check for operands that are incorrectly
split between control and continuation
cards. Correct any errors and execute the
job step again. If the problem recurs, do
the following before calling IBM for
programming support:
- Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
- Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
was included for the failing job step.
- Have the associated job stream and
master console log available.
- Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER005A - STMT DEFINER ERR

Explanation: Critical. A control
statement that should contain an operation
definer (SORT, MERG, RECORD, MODS, or END)
does not contain an acceptable one.

System Action: Stage 1 termination.

Programmer Response: Probable user error.
Check all statements for incorrect,
misplaced, or misspelled operation
definers. Correct any errors and execute
the job step again. If the problem
recurs, do the following before calling
IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
- Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
was included for the failing job step.
- Have the associated job stream and
master console log available.
- Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER006A - OP DEFINER ERR

Explanation: Critical. The first operand
of a control statement does not begin on
the same statement as the operation
definer.

System Action: Stage 1 termination.

Programmer Response: Probable user error.
Check for statements that contain only the
operation definer. Correct any errors and
execute the job step again. If the
problem recurs, do the following before
calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
- Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
was included for the failing job step.
- Have the associated job stream and
master console log available.
- Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER007A - SYNTAX ERR - xxx

Explanation: Critical. A control
statement contains an error in syntax.
xxx is a 3-character code ("S/M," "REC,"
or "MOD") that indicates the control
statement in which the error occurred.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check the control statements for syntax
errors. Some of the more common syntax
errors are:
- Unbalanced parentheses.
- Missing commas.
- Embedded blanks.
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
- Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
- Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters . (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
was included for the failing job step.
- Have the associated job stream and
master console log available.
- Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER008A - FLD OR VALUE GT 8 CHAR - xxx

Explanation: Critical. A parameter of
greater than 8 characters has been
specified. xxx is a 3-character code
("S/M," "REC," or "MOD") that indicates
the control statement in which the error
occurred.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check control statements for parameters
with more than eight characters. Correct
any errors and execute the job step again.
If the problem recurs, do the following
before calling IBM for programming
support:
• Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
• Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters . (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
was included for the failing job step.
• Have the associated job stream and
master console log available.
• Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER009I - EXCESS INFO ON CARD - xxx

Explanation: More information than
necessary appears in a control statement.
This could possibly be caused by a syntax
error which cannot be diagnosed by the
program. xxx is a 3-character code
("S/M," "REC," or "MOD") that indicates
the control statement in which the error
occurred. This message is also printed
when comments appear on a card.

System Action: The excess information is
treated as a comment.

Programmer Response: Probable user error.
Check control statements, unless comments
are intended. Correct any errors and
execute the job step again. If the
problem recurs, do the following before
calling IBM for programming support:
• Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
• Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters . (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
was included for the failing job step.
• Have the associated job stream and
master console log available.
• Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER010A - NO S/M CARD

Explanation: Critical. All control
statements have been processed and no SORT
or MERGE control statement has been found.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Supply a SORT or MERGE control statement.
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
• Make sure that MSGLEVEL=(1,1) was

specified in the JOB statement.
• Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters . (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
was included for the failing job step.
• Have the associated job stream and
master console log available.
• Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER011A - TOO MANY S/M KEYWORDS

Explanation: Critical. More than the
maximum of 5 keyword operands are defined
on a SORT or MERGE control statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that the SORT or MERGE control
statement does not contain too many
keyword operands. Valid keywords are as
follows:
FIELDS, FORMAT, SIZE
    (for SORT or MERGE statement)
    and
SKIPREC and CKPT
    (for SORT statement only.)
Correct any errors and execute the job
step again. If the problem recurs,do the
following before calling IBM for
programming support:
• Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
• Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
was included for the failing job step.
• Have the associated job stream and
master console log available.
• Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER012A - NO FLD DEFINER

Explanation: Critical. A SORT or MERGE
control statement does not contain a
control field definition.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check SORT or MERGE control statement for
lack of a control field definition (FIELDS
operand). Correct any errors and execute
the job step again. If the problem
recurs, do the following before calling
IBM for programming support:
• Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
• Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters . (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
was included for the failing job step.

• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER013A - INVALID S/M KEYWORD

Explanation: Critical. An invalid
keyword operand has been detected on a
SORT or MERGE control statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that the SORT or MERGE control
statement does not contain an invalid
keyword operand. Valid keywords are
FIELDS, FORMAT, and SIZE. (In SORT
statements, SKIPREC and CKPT are also
valid.) Correct any errors and execute
the job step again. If the problem
recurs, do the following before calling
IBM for programming support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER014A - DUPLICATE S/M KEYWORD

Explanation: Critical. A keyword operand
is defined twice on a SORT or MERGE
control statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check SORT or MERGE control statement for
a multiply-defined keyword operand.
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters . (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER015A - TOO MANY PARAMETERS

Explanation: Critical. Too many
parameters are associated with a keyword
operand on a SORT or MERGE control
statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check SORT or MERGE control statement
keyword operands for too many parameters.
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER016A - INVALID VALUES IN FLD

Explanation: Critical. An invalid number
of values is specified with a FIELDS
operand on a SORT or MERGE control
statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that the FIELDS operand of a
SORT or MERGE statement is specified
correctly. Valid formats are:
    FIELDS=(location, length, data format,
        order,...)
    or
    FIELDS=(location, length, order,...),
        FORMAT=
Correct any errors and execute the job
step again. If the problem recurs,do the
following before calling IBM for
programming support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER017A - ERR IN DISP LENGTH VALUE

Explanation: Critical. An invalid length
or displacement (position) value is
specified in a control field definition on
a SORT or MERGE control statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that the length and position
values in the FIELDS operand of a SORT or
MERGE control statement were specified
correctly. Make sure that the length

value plus the position value does not exceed 4096, and that bit positions are specified for binary fields only. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER018A - CTL FLD ERR

Explanation: Critical. An error in specifying the type of control field defined in a SORT or MERGE control statement has been detected.

System Action: Stage 2 termination.

Programmer Response: Probable user error. Make sure that all control field types are specified as either CH, ZD, PD, FI, BI, or FL. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER019A - SIZE/SKIPREC ERR

Explanation: Critical. An error in specifying the SIZE operand in either a SORT or MERGE control statement, or the SKIPREC operand in a SORT control statement, has been detected.

System Action: Stage 2 termination.

Programmer Response: Probable user error. Make sure that the SORT or MERGE control statement does not contain an invalid SIZE or SKIPREC operand. (If E is specified in the SIZE operand, it must precede the number.) Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER020A - INVALID REC KEYWORD

Explanation: Critical. An invalid keyword operand has been found in a RECORD control statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error. Make sure that the keyword operands in the RECORD control statement are valid and that they are spelled correctly. Valid keywords are TYPE and LENGTH. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER021A - NO TYPE DEFINER

Explanation: Critical. A RECORD control statement has been found without a TYPE operand.

System Action: Stage 2 termination.

Programmer Response: Probable user error. Check RECORD control statement for lack of TYPE operand. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER022A - RCD FORMAT NOT F/V

    Explanation: Critical. An error in
specifying the value associated with the
TYPE operand of a RECORD control statement
has been detected.

    System Action: Stage 2 termination.

    Programmer Response: Probable user error.
Check the RECORD control statement for
keypunching or other errors resulting in
TYPE operand value being some character
other than F (fixed-length records) or V
(variable-length records). Correct any
errors and execute the job step again. If
the problem recurs, do the following
before calling IBM for programming
support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER023A - NO LENGTH DEFINER

    Explanation: Critical. The LENGTH
operand of a RECORD control statement is
not present.

    System Action: Stage 2 termination.

    Programmer Response: Probable user error.
Check RECORD control statement for lack of
LENGTH operand. Correct any errors and
execute the job step again. If the
problem recurs, do the following before
calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER024A - ERR IN LENGTH VALUE

    Explanation: Critical. An incorrect
value is associated with the LENGTH
parameter of a RECORD control statement.

    System Action: Stage 2 termination.

    Programmer Response: Probable user error.
Make sure that the RECORD control
statement does not contain the following
errors:

- Keypunching errors in length values.
  (length values must not contain
  non-numeric characters, negative
  numbers, etc.)
- More than three length values specified
  for fixed length records.
- Minimum length (l_4) greater than maximum
  length (l_2).
- Modal length (l_5) greater than maximum
  length (l_2).
- Modal length (l_5) greater than maximum
  input length (l_1) or the logical record
  length specified on the SORTIN DD
  statement.
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER025A - RCD SIZE GT MAX

    Explanation: Critical. The logical
record size specified on a RECORD control
statement is greater than the maximum
allowed by the program.

    System Action: Stage 2 termination.

    Programmer Response: Probable user error.
Make sure that the record length specified
in the RECORD control statement is
correct. Maximum length is 32,000 bytes
(or 27,400 bytes for spanned records).
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER026A - L1 NOT GIVEN

    Explanation: Critical. The LENGTH
operand of a RECORD control statement
lacks an l_1 value.

    System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check RECORD control statement for lack of
l_1 value in LENGTH operand. Correct any
errors and execute the job step again. If
the problem recurs, do the following
before calling IBM for programming
support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER027A - CF BEYOND RCD

Explanation: Critical. A control field
has been defined as extending beyond the
maximum record length specified in a
RECORD control statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check SORT or MERGE control statement for
incorrectly specified control field
displacement. Check RECORD control
statement for incorrectly specified
maximum record length (l_2). Correct any
errors and execute the job step again. If
the problem recurs, do the following
before calling IBM for programming
support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER028A - TOO MANY EXITS

Explanation: Critical. An attempt has
been made to activate more than the
maximum number of program exits allowed by
the program (17).

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that exit routines are specified
for valid exits only, and that each exit
is associated with only one exit routine.
Valid exits are E11, E15, E16, E17, E18,
E19, E21, E25, E28, E29, E31, E35, E37,
E38, E39, and E61. (Note: For a
merge-only application, only exits E31,
E35, E37, E38, E39, and E61 are valid.)

Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER029A - IMPROPER EXIT

Explanation: Critical. This message is
generated for one of two reasons:
• An exit other than the 17 allowed by the
  program has been activated on a MODS
  control statement.
• An exit in the sort or intermediate
  merge phase of the program has been
  activated during a merge application.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that the MODS control statement
does not contain keypunch or other errors
that resulted in the specification of an
invalid program exit number. Valid
numbers are E11, E15, E16, E17, E18, E19,
E21, E25, E27, E28, E29, E31, E35, E37,
E38, E39, and E61. (Note: For a
merge-only application, only exits E31,
E35, E37, E38, E39, and E61 are valid.)
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have value available.

IER030A - MULTIPLY DEFINED EXIT

Explanation: Critical. A program exit
has been defined twice in MODS control
statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check MODS control statement for multiply
defined exits. Correct any errors and
execute the job step again. If the
problem recurs, do the following before
calling IBM for programming support:

- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER031A - INVALID MODS OP CHAR

Explanation: Critical. An invalid
character in a parameter of a MODS control
statement has been found.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check the parameters of a MODS statement
for a length field containing something
other than numeric data, or a source or
name field beginning with something other
than an alphabetic character or one of the
special characters $, @, # or containing a
special character other than $, @, #.
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER032A - EXIT E61 REQUIRED

Explanation: Critical. A SORT or MERGE
control statement defines a control field
calling for user-written routine (this is
done by specifying E for the control field
sequence indicator), and exit E61 is not
activated by a MODS control statement.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check SORT or MERGE control statements for
keypunching errors resulting in the
specification of an E type parameter.
Check the MODS control statement, for lack
of an E61 specification. Correct any
errors and execute the job step again. If
the problem recurs, do the following
before calling IBM for programming
support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and

MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER033A - CF SEQ INDIC E REQUIRED

Explanation: Critical. Program exit E61
is activated and no control fields have
been specified for user modification (E
control field sequence parameter missing
on SORT or MERGE control statement).

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Check MODS, and SORT or MERGE control
statements for keypunching errors
resulting in the activation of exit E61
and the lack of an E type parameter on the
SORT or MERGE control statement. Correct
any errors and execute the job step again.
If the problem recurs, do the following
before calling IBM for programming
support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER034A - PARAM ERR FOR MODS

Explanation: Critical. An incorrect
number of parameters follow an operand
definer on a MODS control statement, or
SYSIN is specified on the MODS statement
as the source for user-written routines,
and no //SORTMODS card is present.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that any MODS control statements
have the following format:

$$\text{MODS exit= (name,size,} \begin{Bmatrix} \text{ddname of library} \\ \text{SYSIN} \end{Bmatrix} \begin{bmatrix} , \begin{Bmatrix} N \\ S \end{Bmatrix} \end{bmatrix})\text{,..}$$

If SYSIN has been specified, make sure
that a SORTMODS DD statement is also
included in the step. Correct any errors
and execute the job step again. If the
problem recurs, do the following before
calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,

the CORE parameter should also be
included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
was included for the failing job step.
- Have the associated job stream and
master console log available.
- Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER035A - DUPLICATE MOD RTN IN PHASE

Explanation: Critical. The same
user-written routine is being used for
more than one exit in a sort/merge program
phase, or two or more routines have the
same name.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that the MODS control statement
does not use duplicate names improperly.
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
- Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
- Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
was included for the failing job step.
- Have the associated job stream and
master console log available.
- Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER036I - B = xxxxxx

Explanation: This message communicates
the blocking used by the sort for
intermediate storage records. For
fixed-length records, the blocking factor
is substituted for xxxxxx in the message
text. For variable-length records, the
size of the buffer area is substituted for
xxxxxx in the message text.

System Action: None.

Programmer Response: None.

IER037I - G = xxxxxx

Explanation: This message communicates
the number of records that can fit into
the program's record storage area at one
time during a sort. The number of records
is substituted for the xxxxxx in the text
of the message as shown above.

System Action: None.

Programmer Response: None.

IER038I - NMAX = xxxxxx

Explanation: This message communicates an
estimate of the maximum number of records
that can be sorted using the intermediate
storage and main storage available to the
sort for the current application. The
number replaces the xxxxxx in the text of
the message as shown above.

Note: Nmax for magnetic tape is
calculated for 2400-foot tapes. For
shorter tapes the figure should be reduced
proportionately.

System Action: None.

Programmer Response: None.

IER039A - INSUFFICIENT CORE

Explanation: Critical. There is not
enough main storage available to the sort
to allow program execution.

System Action: The program terminates.

Programmer Response: Probable user error.
Increase the amount of main storage to be
used by the sort program by coding the
CORE parameter in the PARM field of the
EXEC statement.

To calculate the minimum value for this
parameter:

- For a sort, apply the formula below to
each of the SORTIN and SORTOUT data
sets, and take the greater of the two
results. LEN=LRECL. If input or output
records are spanned, add LRECL to the
SORTIN or SORTOUT result respectively.

- For a merge, apply the formula with
LEN=the largest input blocksize, and
BLKSIZE=the output blocksize. If any
records are spanned, add LRECL x total
no. of spanned files.

Formula:
Min = A x max BLKSIZE + (C x LEN)

	A	C	
For a sort		for SORTIN	for SORTOUT
BALN (tape)	12000	5	(WA+1)/2)
POLY	12000	5	WA
OSCL	18000	max (5,WA)	WA
BALN (disk)	12000	5	WA
CRCX	18000	WA	WA
For a merge	12000	no. of input files	
WA = no. of intermediate work areas			

If user exit routines are used, their size
should be added to this minimum value.
For efficient sorting, allow at least 50%
more storage than the minimum required.
If you are working in an MFT or MVT
environment, be sure to specify a region

or partition size of 1.2 x CORE value +
8K.

If it is not possible to increase the
amount of main storage to be used by the
sort program, try to reduce the storage
requirements by decreasing either the
input blocksize or the number of
intermediate storage areas.

Execute the job step again. If the
problem recurs, do the following before
calling IBM for programming support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER040A- INSUFFICIENT WORK UNITS

Explanation: Critical. There is not
enough intermediate storage available to
the sort to allow program execution. In a
merge-only application this message may be
caused by incorrect specification of one
or more input units. (SORTIN01, etc...)

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Make sure that the DD statements do not
contain errors and that the SORTWK DD
statements are not out of order or
missing. (The numbers must be in
sequence. If tape is used, make sure that
at least three intermediate storage units
were assigned to the sort program. If
2311 disk or 2301 drum devices were used,
make sure that at least three areas of at
least three tracks each are assigned. If
the 2314 storage facility is used, at
least three data sets of at least five
tracks each must be assigned.) Correct
any errors and execute the job step again.
If the problem recurs, do the following
before calling IBM for programming
support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER041A- N GT NMAX

Explanation: Critical. The number of
records specified in the SIZE operand of a
SORT control statement is greater than the
maximum sort capacity calculated by the
program.

System Action: The program terminates
unless data set size was estimated or not
given; then sort continues.

Programmer Response: Probable user error.
Check SIZE operand of SORT control
statement for error. If SIZE operand is
correct, check DD statements for an error
in assigning intermediate storage. If DD
statements are correct, assign more
intermediate storage to the program and
rerun. Correct any errors and execute the
job step again. If the problem recurs, do
the following before calling IBM for
programming support:
• Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
• Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
• Make sure that a SYSABEND DD statement
  was included for the failing job step.
• Have the associated job stream and
  master console log available.
• Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER042A- UNITS ASGN ERROR

Explanation: Critical. A. Different
types of intermediate storage devices, or
an invalid combination of input, work, and
output devices have been assigned to the
sort. B. Duplicate ddnames have been
specified.

System Action: Stage 2 termination.

Programmer Response: Probable user error.
For case A, assign intermediate storage so
that all units are of the same type of
direct access device or are all 9 track
tape units. However if the sort input
unit is a 7 track tape device, then any
combination of 7 and 9 track tape units
may also be used.

Input Device	Intermediate Storage					
	All 2311	All 2314	All 2301	9 trk tape	7 trk tape	Mixed 7&9 trk tapes
Any device supported by QSAM except 7 track tape	Yes	Yes	Yes	Yes	No	No
7 trk tape	Yes	Yes	Yes	Yes	Yes	Yes

For case B, make sure that duplicate
ddnames were not specified. Correct any

errors and execute the job step again. If
the problem recurs, do the following
before calling IBM for programming
support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER043A- DATA SET ATTRIBUTES NOT SPECIFIED

Explanation: Critical. DD statements
that define the input and output data sets
conflict with each other or lack any of
the following information:
- Input or output blocksize (BLKSIZE).
- Record format (RECFM).
- Record length (LRECL).

System Action: Stage 2 termination.

Programmer Response: Probable user error.
Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER044I- EXIT Exx INVALID OPTION

Explanation: An invalid data control
block field specification was passed to
the sort/merge program at exit E18, E19,
E28, E29, E38, or E39. The xx value in
the above message text is replaced by the
number of the exit at which the error
occurred.

System Action: The invalid option is
ignored.

Programmer Response: Probable user error.
Check the parameter list passed by the
user-written routine against the following
table before rerunning the application.
An x indicates which options are allowed
with the exit in question.

Option	E18	E19	E28	E29	E38	E39
SYNAD	x	x	x	x	x	x
EXLST	x	x	x	x	x	x
EROPT	x		x		x	
EODAD	x					

Correct any errors and execute the job
step again. If the problem recurs, do the
following before calling IBM for
programming support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER045I - END SORT PH

Explanation: The program's sort phase has
been successfully executed.

System Action: None.

Programmer Response: None.

IER046A - SORT CAPACITY EXCEEDED

Explanation: Critical. The sort has used
up all available intermediate storage and
the input data set has not been exhausted.

System Action: The program terminates.

Programmer Response: Probable user error.
If magnetic tape is used for intermediate
storage be sure that all reels contain
full-length tapes. (A bad tape may appear
short because of a large number of write
errors.) If all reels contain full length
tapes, rerun the application and specify
more intermediate devices. If a direct
access device is used for intermediate
storage, assign more tracks to the sort.
Note that reverse sequence files may
require more space. If the problem
recurs, do the following before calling
IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
- Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
- Make sure that a SYSABEND DD statement
  was included for the failing job step.
- Have the associated job stream and
  master console log available.
- Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER047A - RCD CNT OFF, IN xxxxxx, OUT xxxxxx

> Explanation: Critical. The number of
> records entering and leaving a program
> phase are not equal; these numbers do not
> include records inserted or deleted by
> user-written routines. If an actual data
> set size was specified in the SIZE
> parameter of the SORT control statement,
> it is placed in the IN field of this
> message. This message can appear in phase
> 1 or phase 2. In phase 3 the message is
> RCD CNT OFF and message IER054I contains
> the count. The numbers replace the values
> specified as xxxxxx in the text of the
> message as shown above.
>
> System Action: The program terminates.
>
> Programmer Response: Probable user error.
> Make sure that the value of the SIZE
> parameter in the SORT control statement is
> accurate. If it is correct, or if no
> value has been specified, make sure that
> the intermediate storage devices have not
> been demounted during the job. Correct
> any errors and execute the job step again.
> If the problem recurs, do the following
> before calling IBM for programming
> support:
> • Make sure that MSGLEVEL=(1,1) was
>   specified in the JOB statement.
> • Make sure that the PARM parameter of the
>   EXEC statement contained the DIAG and
>   MSG=AP subparameters. (If applicable,
>   the CORE parameter should also be
>   included in the EXEC statement.)
> • Make sure that a SYSABEND DD statement
>   was included for the failing job step.
> • Have the associated job stream and
>   master console log available.
> • Note the amount of main storage
>   specified at sort/merge generation, and
>   have this value available.

IER048I - NMAX EXCEEDED

> Explanation: Critical. The sort has
> exceeded the calculated sort capacity
> while processing the input data set, and
> exit E16 is specified.
>
> System Action: The user-written routine
> at exit E16 is entered.
>
> Programmer Response: Probable user error.
> No response necessary. (The number of
> records sorted is equal to the NMAX
> calculated by the sort. See sort message
> IER038I.) Execute the job step again. If
> the problem recurs, do the following
> before calling IBM for programming
> support:
> • Make sure that MSGLEVEL=(1,1) was
>   specified in the JOB statement.
> • Make sure that the PARM parameter of the
>   EXEC statement contained the DIAG and
>   MSG=AP subparameters. (If applicable,
>   the CORE parameter should also be
>   included in the EXEC statement.)
> • Make sure that a SYSABEND DD statement
>   was included for the failing job step.
> • Have the associated job stream and
>   master console log available.
> • Note the amount of main storage

> specified at sort/merge generation, and
> have this value available.

IER049I - SKIP MERGE PH

> Explanation: It is not necessary to
> execute the intermediate merge phase to
> complete a sorting application because the
> number of sequences created by the sort
> phase is ≤ the merge order.
>
> System Action: Control is passed directly
> from the sort phase to the final merge
> phase.
>
> Programmer Response: None.

IER050I - END MERGE PH

> Explanation: The program's intermediate
> merge phase has been successfully
> executed.
>
> System Action: None.
>
> Programmer Response: None.

IER051A - UNENDING MERGE

> Explanation: Critical. There is not
> enough intermediate storage assigned to
> successfully complete the program's
> intermediate merge phase.
>
> System Action: The program terminates.
>
> Programmer Response: Probable user error.
> Assign more intermediate storage and rerun
> the job. Note that reverse sequence files
> may require more space. If the problem
> recurs, do the following before calling
> IBM for programming support:
> • Make sure that MSGLEVEL=(1,1) was
>   specified in the JOB statement.
> • Make sure that the PARM parameter of the
>   EXEC statement contained the DIAG and
>   MSG=AP subparameters. (If applicable,
>   the CORE parameter should also be
>   included in the EXEC statement.)
> • Make sure that a SYSABEND DD statement
>   was included for the failing job step.
> • Have the associated job stream and
>   master console log available.
> • Note the amount of main storage
>   specified at sort/merge generation, and
>   have this value available.

IER052I - EOJ

> Explanation: The program's final merge
> phase has been successfully executed.
>
> System Action: Return is made to the
> operating system for a normal end of task.
>
> Programmer Response: None.

IER053A - OUT OF SEQ

> Explanation: Critical. The current
> record leaving the final merge phase is
> not in collating sequence with the last
> record blocked for output.

System Action: The program terminates.

Programmer Response: Probable user error.
If variable length format records are
being processed, make sure that the input
records are all large enough to contain
the specified control fields. If a
user/written routine was modifying the
records leaving the final merge phase at
the time this message was printed, check
the routine thoroughly. If out of
sequence records are to be inserted by the
user routine, make sure that the correct
parameter to suppress the sequence check
is returned to Sort/Merge. If no user
exit routine was used, rerun the job. If
the problem recurs, do the following
before calling IBM for programming
support:
● Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
● Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
● Make sure that a SYSABEND DD statement
  was included for the failing job step.
● Have the associated job stream and
  master console log available.
● Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER054I - RCD IN xxxxxx, OUT xxxxxx

Explanation: This message lists the
number of records accepted by the sort as
input and the number of records in the
output data set. The numbers replace the
xxxxxx in the text of the message as shown
above. Leading zeros are suppressed; if
there were no records in the input data
set, this field will be blank. In a
merging application, the RECORDS IN field
is blank unless an actual data set size
was specified in the SIZE parameter of the
MERGE control card. When an actual size
is specified, it is inserted in the IN
field of the message.

System Action: None.

Programmer Response: None.

IER055I - INSERT xxxxxx, DELETE xxxxxx

Explanation: The number of records
inserted and/or deleted during a
sort/merge program execution replaces the
values shown as xxxxxx in the above
format.

System Action: None.

Programmer Response: None.

IER056A - SORTIN/SORTOUT NOT DEFINED

Explanation: Critical. SORTIN and/or
SORTOUT do not appear as ddnames on DD
statements supplied to the sort/merge
program. This message can also appear
when DD statements are supplied for a

merge, and a SORT control statement is
given instead of a MERGE statement.

System Action: The program terminates.

Programmer Response: Probable user error.
Check DD statements for error. Correct
any error and execute the job step again.
If the problem recurs, do the following
before calling IBM for programming
support:
● Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
● Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
● Make sure that a SYSABEND DD statement
  was included for the failing job step.
● Have the associated job stream and
  master console log available.
● Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER057A - SORTIN NOT SORTWK01

Explanation: Critical. An intermediate
storage data set other than SORTWK01 was
assigned to the same tape drive as SORTIN.

System Action: The program terminates.

Programmer Response: Probable user error.
Check DD statements for error. Correct
any error and execute the job step again.
If the problem recurs, do the following
before calling IBM for programming
support:
● Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
● Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,
  the CORE parameter should also be
  included in the EXEC statement.)
● Make sure that a SYSABEND DD statement
  was included for the failing job step.
● Have the associated job stream and
  master console log available.
● Note the amount of main storage
  specified at sort/merge generation, and
  have this value available.

IER058A - SORTOUT A WORK UNIT

Explanation: Critical. SORTOUT was
specified on the same tape drive as an
intermediate storage data set.

System Action: The program terminates.

Programmer Response: Probable user error.
Check DD statements for error. Correct
any error and execute the job step again.
If the problem recurs, do the following
before calling IBM for programming
support:
● Make sure that MSGLEVEL=(1,1) was
  specified in the JOB statement.
● Make sure that the PARM parameter of the
  EXEC statement contained the DIAG and
  MSG=AP subparameters. (If applicable,

the CORE parameter should also be
included in the EXEC statement.)
* Make sure that a SYSABEND DD statement
was included for the failing job step.
* Have the associated job stream and
master console log available.
* Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER059A - RCD LNG INVALID FOR DEVICE

Explanation: Critical. The record length
in a data set to be handled by the program
is either less than 18 bytes, or is too
large for the assigned intermediate
storage devices. (For example, a record
which cannot be contained on one disk
track is too large.)

System Action: The program terminates.

Programmer Response: Probable user error.
If the record length is too small,
redefine the sort with a record length of
at least 18 bytes. If the length is too
large, assign a different type of
intermediate storage device. Maximum
lengths for various devices are:

Device	Maximum Record Length
2311	3,600 bytes
2314	7,272 bytes
2301	20,458 bytes
Tape	32,000 bytes
Tape (spanned records)	27,400 bytes

If the problem recurs, do the following
before calling IBM for programming
support:
* Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
* Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
* Make sure that a SYSABEND DD statement
was included for the failing job step.
* Have the associated job stream and
master console log available.
* Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER060A - DSCB NOT DEFINED

Explanation: Critical. A DD statement
used to define a direct access
intermediate storage data set is
incorrect.

System Action: The program terminates.

Programmer Response: Probable user error.
Make sure that no DD statements are in
error. Make sure that deferred mounting
of direct access intermediate storage data
sets is not specified. Correct any errors
and execute the job step again. If the

problem recurs, do the following before
calling IBM for programming support:
* Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
* Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
* Make sure that a SYSABEND DD statement
was included for the failing job step.
* Have the associated job stream and
master console log available.
* Note the amount of main storage
specified at sort/merge generation, and
have this value available.

IER061A - I/O ERR, jobname, stepname, unit
address, device type, ddname, operation
attempted, error description, last seek
address or block count, access method.
(SYNADAF)

Explanation: Critical.

This message is generated for one of two
reasons:

* The job control cards incorrectly
specify record length or blocking
information for the data set located on
the device indicated by the "unit
address" field in the message.

* A permanent error occurred during an I/O
operation on the indicated device.

System Action: If no user options are
specified the program terminates.

Operator Response: If the "error
description" field in the message
indicates a hardware error, execute the
job again with the indicated unit offline,
using an alternative unit and/or volume in
its place during execution.

Programmer Response: Probable user error.
Make sure that the DD statement for the
data set assigned to this device contains
the correct DCB information. (In a merge
application, if the device in error holds
an input data set, make sure that the DCB
information specified in the SORTIN01 DD
statement correctly describes the data on
this device except for BLKSIZE.) Correct
any errors and execute the job step again.
If the problem recurs, do the following
before calling IBM for programming
support:
* Make sure that MSGLEVEL=(1,1) was
specified in the JOB statement.
* Make sure that the PARM parameter of the
EXEC statement contained the DIAG and
MSG=AP subparameters. (If applicable,
the CORE parameter should also be
included in the EXEC statement.)
* Make sure that a SYSABEND DD statement
was included for the failing job step.
* Have the associated job stream and
master console log available.
* Note the amount of main storage
specified at sort/merge generation, and
have this value available.

Explanation: Critical. The linkage editor found a serious error; execution of the sort/merge program is impossible.

System Action: The program terminates.

Programmer Response: Probable user error. Make sure that the DD statements used by the linkage editor are correct and that none are missing. (If the linkage editor is used, the SYSPRINT, SYSLIN, SYSUT1, and SYSLMOD DD statements must be supplied, unless the SORT cataloged procedure is specified in the EXEC statement.) If the DD statements are correct, make sure that all user routines in libraries or in the system input stream are correctly assembled object modules or load modules, and that modules to be linkage edited together do not contain duplicate entry point names. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER063A - OPEN ERR xxxxxxxx

Explanation: Critical. An error occurred during execution of the OPEN routine for data set xxxxxxxx, where xxxxxxxx represents the ddname of the data set being opened.

System Action: The program terminates.

Programmer Response: Probable user error. Check for a missing DD statement. Correct any error and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
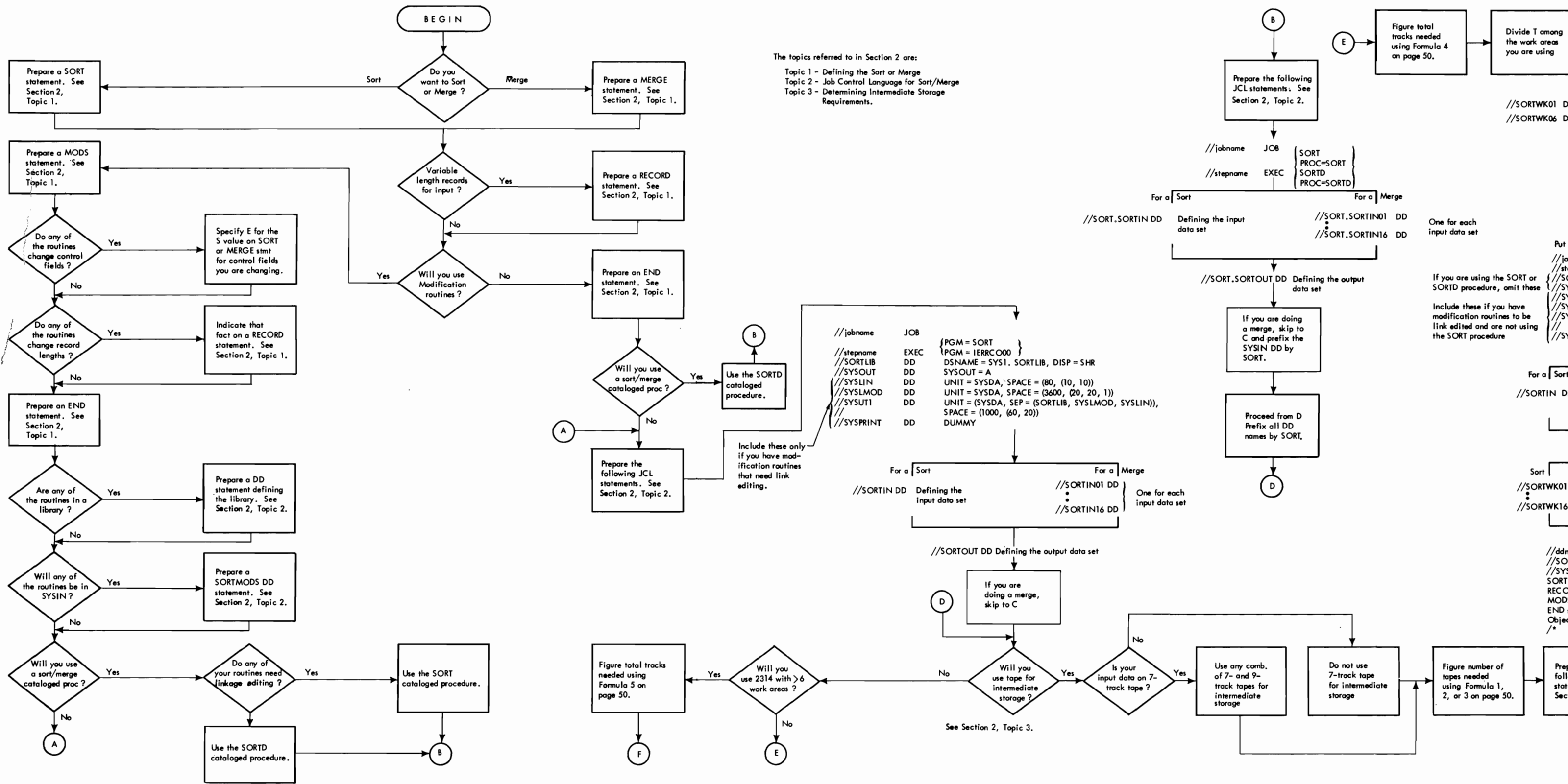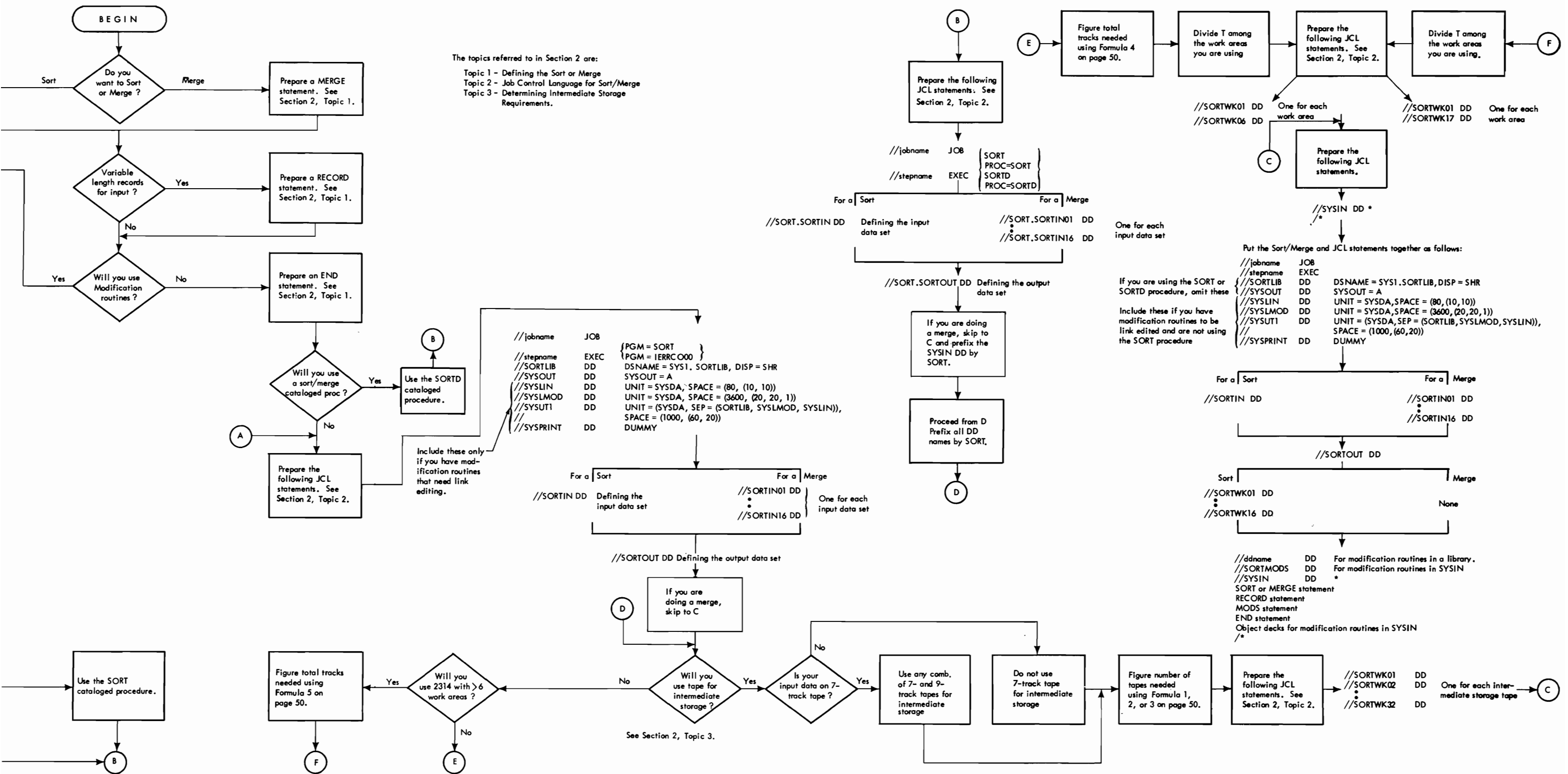- Note the amount of main storage specified at sort/merge generation, and have this value available.

Explanation: Critical. The sort/merge program was unable to delete either itself or a user-written modification routine. This message should appear only when modification routines are used.

System Action: The program terminates.

Programmer Response: Probable user error. Make sure that the user modification routines are not modifying the sort/merge program code and information areas, and rerun the job. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER065A - PROBABLE DECK STRUCTURE ERROR

Explanation: Critical. The end of the SYSIN data set was found before all needed user modification modules were read.

System Action: The program terminates.

Programmer Response: Probable user error.
1. Be sure the SYSIN data set contains all modification routines that the MODS statement specifies it will contain.
2. Check for misplaced job control language statements, especially a /* preceding a user modification module on SYSIN. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
- Make sure that a SYSABEND DD statement was included for the failing job step.
- Have the associated job stream and master console log available.
- Note the amount of main storage specified at sort/merge generation, and have this value available.

IER066A - APROX RECD CNT xxxxxx

> Explanation: Critical. Sort capacity has been reached. The count xxxxxx is an approximation of the number of records the sort/merge program can handle with the assigned intermediate storage.
>
> System Action: The program terminates.
>
> Programmer Response: Probable user error. Respond as indicated in the accompanying message, IER046A. If the problem recurs, do the following before calling IBM for programming support:
> • Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
> • Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
> • Make sure that a SYSABEND DD statement was included for the failing job step.
> • Have the associated job stream and master console log available.
> • Note the amount of main storage specified at sort/merge generation, and have this value available.

IER067I - INVALID EXEC OR ATTACH PARAMETER

> Explanation: An error was found in the PARM field parameters of the EXEC statement, or in the optional parameters of the parameter list passed to a sort initiated by ATTACH, LINK, or XCTL. Invalid parameters are ignored. If a parameter is entered more than once, the last entry is used (if valid).
>
> System Action: Processing continues. Invalid parameters are ignored.
>
> Programmer Response: Probable user error. No action is necessary. For later runs, make sure that the optional parameters are valid. Valid parameters are:
> • one of BALN, OSCL, POLY, CRCX
> • CORE=xxxxxx
> • MSG=xx where xx is either AP, AC, CP, CC, or NO.
> If the problem recurs, do the following before calling IBM for programming

support:
• Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
• Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
• Make sure that a SYSABEND DD statement was included for the failing job step.
• Have the associated job stream and master console log available.
• Note the amount of main storage specified at sort/merge generation, and have this value available.

IER068A - OUT OF SEQ SORTINxx

> Explanation: Critical. During a merge only run, a data set was found to be out of sequence. The xx is replaced by the data set identification (01 to 16).
>
> System Action: The program terminates.
>
> Programmer Response: Probable user error. If a user written routine was modifying the records, check the routine thoroughly. It should not modify control fields at exit E35. If no user written routine is being used, make sure that all input data sets have been sorted by the same method, and that they all have a similar format. Correct any errors and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
> • Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
> • Make sure that the PARM parameter of the EXEC statement contained the DIAG and MSG=AP subparameters. (If applicable, the CORE parameter should also be included in the EXEC statement.)
> • Make sure that a SYSABEND DD statement was included for the failing job step.
> • Have the associated job stream and master console log available.
> • Note the amount of main storage specified at sort/merge generation, and have this value available.

BEGIN

Do you want to Sort or Merge ?

Sort → Prepare a SORT statement. See Section 2, Topic 1.

Merge → Prepare a MERGE statement. See Section 2, Topic 1.

Variable length records for input ?

Yes → Prepare a RECORD statement. See Section 2, Topic 1.

Will you use Modification routines ?

No → Prepare an END statement. See Section 2, Topic 1.

Yes → Prepare a MODS statement. See Section 2, Topic 1.

Do any of the routines change control fields ?

Yes → Specify E for the S value on SORT or MERGE stmt for control fields you are changing.

No →

Do any of the routines change record lengths ?

Yes → Indicate that fact on a RECORD statement. See Section 2, Topic 1.

No →

Prepare an END statement. See Section 2, Topic 1.

Are any of the routines in a library ?

Yes → Prepare a DD statement defining the library. See Section 2, Topic 2.

No →

Will any of the routines be in SYSIN ?

Yes → Prepare a SORTMODS DD statement. See Section 2, Topic 2.

No →

Will you use a sort/merge cataloged proc ?

Yes → Do any of your routines need linkage editing ?

Yes → Use the SORT cataloged procedure.

No → Use the SORTD cataloged procedure. → B

No → A

The topics referred to in Section 2 are:
Topic 1 – Defining the Sort or Merge
Topic 2 – Job Control Language for Sort/Merge
Topic 3 – Determining Intermediate Storage Requirements.

Will you use a sort/merge cataloged proc ?

Yes → Use the SORTD cataloged procedure.

No → A → Prepare the following JCL statements. See Section 2, Topic 2.

```
//jobname      JOB
//stepname     EXEC      { PGM = SORT
                          { PGM = IERRCO00 }
//SORTLIB      DD        DSNAME = SYS1. SORTLIB, DISP = SHR
//SYSOUT       DD        SYSOUT = A
//SYSLIN       DD        UNIT = SYSDA, SPACE = (80, (10, 10))
//SYSLMOD      DD        UNIT = SYSDA, SPACE = (3600, (20, 20, 1))
//SYSUT1       DD        UNIT = (SYSDA, SEP = (SORTLIB, SYSLMOD, SYSLIN)),
//                       SPACE = (1000, (60, 20))
//SYSPRINT     DD        DUMMY
```

Include these only if you have modification routines that need link editing.

For a Sort   For a Merge
//SORTIN DD Defining the input data set   //SORTIN01 DD ... //SORTIN16 DD   One for each input data set

//SORTOUT DD Defining the output data set

If you are doing a merge, skip to C

D

Will you use tape for intermediate storage ?   See Section 2, Topic 3.

No →

Yes → Is your input data on 7-track tape ?

No → Do not use 7-track tape for intermediate storage

Yes → Use any comb. of 7- and 9-track tapes for intermediate storage

→ Figure number of tapes needed using Formula 1, 2, or 3 on page 50.

Will you use 2314 with > 6 work areas ?

Yes → Figure total tracks needed using Formula 5 on page 50. → F

No → E

B → Prepare the following JCL statements. See Section 2, Topic 2.

```
//jobname      JOB       { SORT
//stepname     EXEC  PROC=SORT
                          { SORTD
                       PROC=SORTD
```

For a Sort   For a Merge
//SORT.SORTIN DD Defining the input data set   //SORT.SORTIN01 DD ... //SORT.SORTIN16 DD   One for each input data set

//SORT.SORTOUT DD Defining the output data set

If you are doing a merge, skip to C and prefix the SYSIN DD by SORT.

Proceed from D. Prefix all DD names by SORT.

D

E → Figure total tracks needed using Formula 4 on page 50. → Divide T among the work areas you are using

//SORTWK01 DD
//SORTWK06 DD

If you are using the SORT or SORTD procedure, omit these

Include these if you have modification routines to be link edited and are not using the SORT procedure

//job
//ste
//SO
//SY
//SY
//SY
//SY
//
//SY

For a Sort
//SORTIN DD

Sort
//SORTWK01
//SORTWK16

//ddn
//SOR
//SYS
SORT
RECO
MODS
END s
Objec
/*

Summary c

BEGIN

Sort / Merge

Do you want to Sort or Merge ?

Merge → Prepare a MERGE statement. See Section 2, Topic 1.

Variable length records for input ? — Yes → Prepare a RECORD statement. See Section 2, Topic 1.

No

Will you use Modification routines ? — No → Prepare an END statement. See Section 2, Topic 1.

Yes

Will you use a sort/merge cataloged proc ? — Yes → Use the SORTD cataloged procedure.

No

A

Prepare the following JCL statements. See Section 2, Topic 2.

The topics referred to in Section 2 are:
Topic 1 – Defining the Sort or Merge
Topic 2 – Job Control Language for Sort/Merge
Topic 3 – Determining Intermediate Storage Requirements.

B

```
//jobname      JOB
//stepname     EXEC     { PGM = SORT
                          PGM = IERRCO00 }
//SORTLIB      DD       DSNAME = SYS1. SORTLIB, DISP = SHR
//SYSOUT       DD       SYSOUT = A
//SYSLIN       DD       UNIT = SYSDA, SPACE = (80, (10, 10))
//SYSLMOD      DD       UNIT = SYSDA, SPACE = (3600, (20, 20, 1))
//SYSUT1       DD       UNIT = (SYSDA, SEP = (SORTLIB, SYSLMOD, SYSLIN)),
//                      SPACE = (1000, (60, 20))
//SYSPRINT     DD       DUMMY
```

Include these only if you have modification routines that need link editing.

For a Sort   For a Merge
//SORTIN DD   Defining the input data set   //SORTIN01 DD
                                                          .
                                                     //SORTIN16 DD   One for each input data set

//SORTOUT DD Defining the output data set

D

If you are doing a merge, skip to C

Use the SORT cataloged procedure.

Figure total tracks needed using Formula 5 on page 50.

Will you use 2314 with > 6 work areas ? — Yes →

No

Will you use tape for intermediate storage ? — No →

Yes

Is your input data on 7-track tape ? — No →

Yes

Use any comb. of 7- and 9-track tapes for intermediate storage

Do not use 7-track tape for intermediate storage

Figure number of tapes needed using Formula 1, 2, or 3 on page 50.

Prepare the following JCL statements. See Section 2, Topic 2.

```
//SORTWK01     DD
//SORTWK02     DD
     .
//SORTWK32     DD
```
One for each intermediate storage tape → C

See Section 2, Topic 3.

B

F

E

B

Prepare the following JCL statements. See Section 2, Topic 2.

```
//jobname      JOB     { SORT
//stepname     EXEC      PROC=SORT
                         SORTD
                         PROC=SORTD }
```

For a Sort   For a Merge
//SORT.SORTIN DD   Defining the input   //SORT.SORTIN01 DD
                   data set   .
                                                     //SORT.SORTIN16 DD   One for each input data set

//SORT.SORTOUT DD Defining the output data set

If you are doing a merge, skip to C and prefix the SYSIN DD by SORT.

Proceed from D Prefix all DD names by SORT.

D

E

Figure total tracks needed using Formula 4 on page 50.

Divide T among the work areas you are using.

Prepare the following JCL statements. See Section 2, Topic 2.

Divide T among the work areas you are using.

F

```
//SORTWK01 DD                    One for each
//SORTWK06 DD      work area
```

```
//SORTWK01 DD      One for each
//SORTWK17 DD      work area
```

C

Prepare the following JCL statements.

```
//SYSIN DD *
/*
```

Put the Sort/Merge and JCL statements together as follows:

If you are using the SORT or SORTD procedure, omit these

Include these if you have modification routines to be link edited and are not using the SORT procedure

```
//jobname     JOB
//stepname    EXEC
//SORTLIB     DD    DSNAME = SYS1.SORTLIB, DISP = SHR
//SYSOUT      DD    SYSOUT = A
//SYSLIN      DD    UNIT = SYSDA, SPACE = (80, (10,10))
//SYSLMOD     DD    UNIT = SYSDA, SPACE = (3600, (20,20,1))
//SYSUT1      DD    UNIT = (SYSDA, SEP = (SORTLIB, SYSLMOD, SYSLIN)),
//                  SPACE = (1000, (60,20))
//SYSPRINT    DD    DUMMY
```

For a Sort   For a Merge
//SORTIN DD   //SORTIN01 DD
                                                          .
                                                     //SORTIN16 DD

//SORTOUT DD

Sort   Merge
//SORTWK01 DD   None
    .
//SORTWK16 DD

```
//ddname      DD    For modification routines in a library.
//SORTMODS    DD    For modification routines in SYSIN
//SYSIN       DD    *
SORT or MERGE statement
RECORD statement
MODS statement
END statement
Object decks for modification routines in SYSIN
/*
```

Indexes to systems reference library manuals are consolidated in the publication <u>OS Systems Reference Library</u>
<u>Master Index</u>, Order No. GC28-6644. For additional information about any subject listed below, refer to other
publications listed for the same subject in the Master Index.

GC28-6543-8