SC33-4035-7
File No. S370-33

Program Product

OS/VS Sort/Merge Programmer's Guide

Program Number 5740-SM1

Release 5

IBM

SC33-4035-7
File No. S370-33

Program Product

OS/VS Sort/Merge Programmer's Guide

Program Number 5740-SM1

Release 5

IBM

This publication was produced using the
IBM Document Composition Facility
(program number 5748-XX9)
and the master was printed on the
IBM 3800 Printing Subsystem.

Eighth Edition (March 1981)

This is a major revision of, and makes obsolete, SC33-4035-6, and its technical newsletter, SN20-9331.

This edition applies to Release 5, Modification 0, of IBM OS/VS Sort/Merge Program Product 5740-SM1, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Amendments" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

PREFACE

This manual is for programmers who wish to sort or merge files using OS/VS Sort/Merge Program Product No. 5740-SM1.

To use this manual, you should have a basic understanding of OS/VS and its job control language (JCL); to take advantage of all the options and facilities of the program, you will need the documents listed under "Reading List."

Using this manual, you will be able to prepare all the input necessary to perform a sort or merge. You will also be able to link your own routines to the sort/merge program to perform such services as summarizing, altering, or inserting records as they are being sorted or merged.

ORGANIZATION OF MANUAL

This manual contains the following sections:

- "Introduction to the Program" describes the program's relationship to the operating system, and explains the program's functions and facilities, its hardware and storage requirements, user inputs, and factors affecting performance.

- "Writing a Simple Program" describes how to write a sort/merge program for users who are unfamiliar with the product. It takes them, via a flow diagram, through the steps necessary to create a sort/merge application. Also included is an example of a sort application.

- "Calculating Storage Requirements" discusses the storage devices used for intermediate storage, the factors determining the amount of intermediate storage required for a sort/merge program, and the program's method of selecting a sorting technique; it also describes how to calculate main storage requirements.

- "Program Control Statements" describes how you use program control statements to describe your input data, to supply information about the control fields being used, and to describe to the system your own routines that you wish to use during program execution.

- "Job Control Statements" shows you how and what job control statements you must write in order to have your sort/merge program executed.

- "Program Exits and User Routines" describes how you can insert a routine of your own into the sort/merge program, via program exits.

- "Initiating a Program Using System Macro Instructions" describes how to initiate execution of the program from within your own program using a system macro instruction.

- "Improving Program Efficiency" gives hints on how you can get a faster sort or merge operation.

- "Appendix A. What to do if the Program Stops" describes, in the first section, how to localize a problem when sort/merge behaves in an unexpected way; the second section describes various uses of the DEBUG control statement.

- "Appendix B. Data Format Examples" gives examples of the assembled data formats, as used with IBM System 360/370.
- "Appendix C. Error and Information Messages" lists, explains, and suggests responses to all the error messages produced by this sort/merge program.
- "Appendix D. Examples of Control Statements for Sort/Merge Applications "
- "Appendix E. EBCDIC and ASCII Collating Sequences" lists the collating sequences from low to high order for EBCDIC and ASCII characters.
- "Appendix F. Timing Estimates" gives tables that contain estimated maximum total execution times for some sorting applications using this program.

READING LIST

The reading list that follows is divided according to the options and facilities of the program and how you intend to use them.

For All Applications

The following manuals supplement the JCL information given in this guide; you may need them for reference:

OS/VS1 JCL Reference, GC24-5099

OS/VS2 JCL Reference, GC28-0692

For an explanation of SMF record type 16, which provides a way for an installation to collect statistics from which to audit its sort activities, generate utilization reports, develop tuning information, etc., see:

OS/VS1 System Management Facilities (SMF), GC24-5115

OS/VS2 MVS System Programming Library: System Management Facilities (SMF), GC28-0706 (for users of OS/VS2 MVS Release 3.8)

OS/VS2 MVS System Programming Library: System Management Facilities (SMF), GC28-1030 (for users of OS/VS2 MVS/System Product)

For an explanation of the options available at generation time and estimates of storage required by the program, consult:

OS/VS Sort/Merge Installation Guide, SC33-4034

For overall discussion of sort/merge features, see:

OS/VS Sort/Merge General Information, GC33-4033

For quick reference, see:

OS/VS Sort/Merge Reference Summary, SX33-8001

For compatibility of message options from 5734-SM1, see:

OS Sort/Merge Programmer's Guide, SC33-4007

Planning Checkpoint/Restart

Complete information on the advanced checkpoint/restart facility is contained in the publications

OS/VS1 Checkpoint/Restart, GC26-3876

OS/VS2 MVS Checkpoint/Restart, GC26-3877

COBOL and PL/I Users

See the Programmer's Guide describing the compiler version available at your installation.

Assembler Language Users

OS/VS-DOS/VS-VM/370 Assembler Language Manual, GC33-4010

Program Initiation with System Macro Instructions

OS/VS1 Supervisor Services and Macro Instructions, GC24-5103

OS/VS2 MVS Supervisor Services and Macro Instructions, GC28-0683

Data Management

OS/VS1 Data Management Macro Instructions, GC26-3872

OS/VS2 MVS Data Management Macro Instructions, GC26-3793

OS/VS1 Data Management Services Guide, GC26-3874

OS/VS2 MVS Data Management Services Guide, GC26-3875

OS/VS1 Data Management for System Programmers, GC26-3837

OS/VS2 MVS System Programming Library: Data Management, GC26-3830

Dynamic Allocation

OS/VS2 MVS System Programming Library: Job Management, GC28-0627

ASCII

OS/VS1 Data Management Macro Instructions, GC26-3872

OS/VS2 MVS Data Management Macro Instructions, GC26-3793

USASI Tape Labels

OS/VS Tape Labels, GC26-3795

VSAM Users

OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838

OS/VS1 Access Method Services, GC26-3840

OS/VS2 Access Method Services, GC26-3841

For storage requirements, see

Planning for Enhanced VSAM, GC26-3842

For debugging aids, see

OS/VS1 Debugging Guide, GC24-5093

OS/VS2 MVS System Programming Library: Debugging Handbook, Vol. 1, GC28-0708

OS/VS2 MVS System Programming Library: Debugging Handbook, Vol. 2, GC28-0709

SUMMARY OF AMENDMENTS

FOR SC33-4035-7

RELEASE 5, MODIFICATION 0

- Another standard disk sorting technique (VLR-Blockset) has been added to improve performance when sorting Variable Length Records.
- Ability to add to or change installed or passed user options, using the new OPTION control statement.
- Support of 3375 DASD, a new auxiliary storage device for initial input, final output, and intermediate work data sets.
- Ability to produce statistical data about sort applications executed.
- Ability to specify that format CH be translated the same as format AQ.
- Ability to specify whether or not record counters should be checked at the end of execution of sorting applications that use the E35 exit routine without a SORTOUT data set.
- The design point is changed from 48K to 54K bytes.

FOR SC33-4035-6

RELEASE 4, MODIFICATION 0 FROM PTF 43

- Support of 3380 DASD, a new auxiliary storage device for initial input, final output, and intermediate work data sets.

RELEASE 4, MODIFICATION 0

- A further standard disk sorting technique (FLR-Blockset) has been added to improve performance when sorting Fixed Length Records.
- SORTIN/SORTOUT I/O handling is enhanced to improve performance.
- The default printing of the sort/merge specially formatted dump is removed.
- The design point is changed from 32K to 48K bytes.

FOR SC33-4035-5

RELEASE 3, MODIFICATION 0 FROM PTF 32

The optimum disk technique has been made standard by removal of the remaining restrictions on its use. The other disk techniques (BALN and CRCX) are retained for compatibility reasons and can be forced if required.

RELEASE 3, MODIFICATION 0

- Unless one of the nonstandard disk techniques is forced:
  - The sort program's work data sets can be on a mixture of any of the supported disk types.
  - If necessary, a secondary storage allocation is automatically made; this need not be specified in JCL.
  - Unused space is automatically released; this need not be specified in JCL.
- Program control information passed from an invoking program in the parameter list can now be overridden by using a new DD statement, SORTCNTL DD, to identify a data set containing different program control information.

CONTENTS

FIGURES

SECTION 1. INTRODUCTION TO THE PROGRAM

This section describes the relationship of the IBM OS/VS Sort/Merge Program Product 5740-SM1 (hereafter referred to as the sort/merge program—or simply sort/merge) to the operating system; its functions and facilities; its requirements in terms of hardware, main storage, and user input; and factors affecting performance.

RELATIONSHIP TO THE OPERATING SYSTEM

Sort/merge operates under the operating system control program. Therefore, it must be initiated according to operating system conventions: You must define any data sets used by the program according to operating system standards. You can use the label checking facilities of the operating system during program execution. (Operating system label checking facilities are described in OS/VS1 Supervisor Services and Macro Instructions and OS/VS2 MVS Supervisor Services and Macro Instructions.)

Because sort/merge uses the operating system data management facilities, you must describe all data sets (except those allocated via the DYNALLOC parameter) necessary for program operation in job control language data definition (DD) statements. These statements must be placed in the operating system input stream with the job step that initiates program execution.

WHAT THE PROGRAM WILL DO

Sort/merge has two basic functions:

- To sort records, that is, to arrange them in a given sequence.
- To merge from 2 to 16 previously sorted record sequences into one sequence. When you merge records, the sequences to be merged must have been previously sorted into the same order (ascending or descending) as that required for final output.

USING THE PROGRAM EFFICIENTLY

The objective of the sort/merge program is to give as fast a sort or merge as possible. Many factors (such as the size of the work data sets specified, record lengths, default values in operation) are involved in determining the efficiency of the program. These factors are evaluated at the beginning of the program (in phase 0), and optimization takes place in two ways:

- Optimal values are calculated for many variables, such as buffer sizes.
- For a sort, a "sequence distribution" technique is selected automatically.

The program has the following components:

- Four standard disk sorting techniques named VLR-Blockset (the new sorting technique for variable-length records), FLR-Blockset and Peerage (for fixed-length records), and Vale (for both fixed- and variable-length records). (Message ICE092I or ICE093I indicates which of these is being used.)

- Three standard tape sorting techniques named Balanced (BALN), Polyphase (POLY), and Oscillating (OSCL).
- Merge only.
- Two conventional disk sorting techniques, normally not used.

Generally, a disk sort is quicker than a tape sort. If you use tape for sorting you may find it useful to be aware of the factors that influence the program's choice of technique. This topic is discussed in Section 3, "Calculating Storage Requirements" and in Section 8, "Improving Program Efficiency."

LIMITATIONS ON INPUT

Sort Application

Sort input may be a blocked or unblocked sequential data set containing fixed- or variable-length records on any device that can be used with QSAM or VSAM. DSN=NULLFILE cannot be specified when EXCP access method (see OS/VS1 Data Management for System Programmers and OS/VS2 MVS System Programming Library: Data Management) is used (this is a system restriction). QSAM input data sets may be empty, but VSAM data sets may not. Input data sets may be concatenated even if they are on unlike devices, as long as the conditions described in Section 5 under "SORTIN DD Statement" are met.

If a VSAM input data set is password protected, passwords can be entered at the console or (with some restrictions) through routines at exits E18, E38, and E39.

If any of the data sets are on tape without standard labels, you must specify DCB parameters on their DD cards.

The length of the records that the program can handle depends on the amount of main storage available. In no case may the length of any record exceed the length specified by the user as the maximum record length.

Figure 1 shows the maximum record length the program will accept for a given amount of main storage when fixed- or variable-length records are used.

For spanned records, maximum lengths are slightly smaller. Conditions such as control fields of different formats, large numbers of control fields, or large numbers of work data sets reduce the length of the records that may be sorted using a given amount of storage. The minimum block length for tape work units is 18 bytes; the minimum record length is 14 bytes.

Main Storage Available (in bytes)	Intermediate Storage Device	
	Tape	DA Devices
<64K	3,200	1,200
64K	8,500	7,000
128K	19,000	13,000
256K	32,000	32,760

Figure 1. Maximum Input and Output Record Lengths

Merge Application

Input to the merge may be up to 16 blocked or unblocked sequential data sets containing fixed- or variable-length records on any device that can be used with QSAM or VSAM. The input data sets may be either QSAM or VSAM, but not both. The records in the input data sets must be already sorted into the same order as that required for final output. For a given application all records must be of the same format, but the blocking factors may differ if the data set with the largest block size is specified in the SORTIN01 DD statement.

LIMITATIONS ON OUTPUT

Output may be to either QSAM or VSAM data sets, regardless of whether input was QSAM or VSAM. However, a VSAM data set used for output must have been previously defined using the Access Methods Service utility.

If output is a keyed-sequential VSAM data set (KSDS), then the key must be the major control field (or the key fields must be in the same order as the major control field). Note that most versions of VSAM do not allow the storing of records with duplicate keys.

The output record type (fixed or variable) must be the same as the input record type.

CONTROL FIELDS AND COLLATING SEQUENCE

The program orders your records on the basis of one or more control fields you specify. The first field you specify is called the major field. The program compares the major fields of the records and sorts them in ascending or descending order (according to which order you have specified).

All other fields you specify are called minor fields. If the major fields in two records are equal, the program sorts the records according to the minor fields you have specified. If the first minor fields in two records are equal, the program compares the second minor fields, and so on, until it finds a difference, or the end of the control field is reached.

The input order of records will be preserved on the output data set if all their control fields are identical, and the EQUALS option is specified (see "SORT Control Statement").

Control fields may overlap, or be contained within other control fields. They need not be contiguous, but must be located in the first 4092 bytes of the record.

The collected control fields of each record, arranged in order of priority, are regarded by the program as a single control word which can be up to 4092 bytes long.

A control word made up of four control fields is shown in Figure 2.

Records are sorted using either the standard IBM collating sequence (EBCDIC) or the ASCII collating sequence.

The EBCDIC sequence can be modified, for example to allow the alphabetic collation of national characters. The modification can be generated as a default when the program is installed; or you can specify it at execution time through the ALTSEQ control statement.

You can also specify at installation time or by means of a parameter of the OPTION control statement that both format CH and format AQ fields should be translated using the ALTSEQ table, or only format AQ.

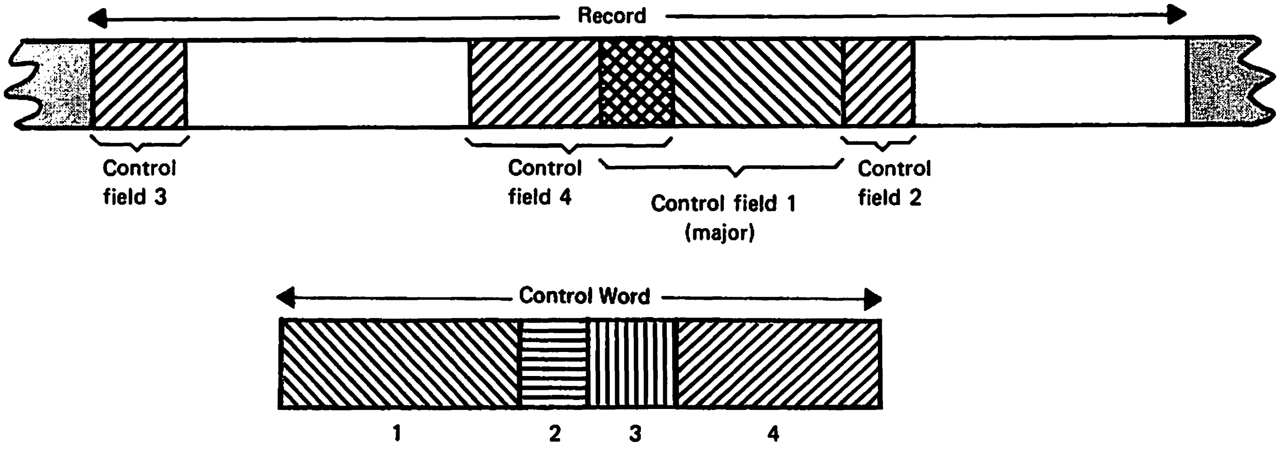


Figure 2. Control Fields

The collating sequence for character data and binary data is absolute; that is, character and binary fields are not interpreted as having signs. For packed decimal, zoned decimal, fixed-point, normalized floating-point, and the signed numeric data formats, collating is algebraic; that is, each quantity is interpreted as having an algebraic sign.

PROGRAM FACILITIES AND OPTIONS

Some of the program default values depend on the specifications made by your system programmer when the sort/merge program was installed. Sort/merge installation is described in the *OS/VS Sort/Merge Installation Guide*.

The following list is a summary of the sort/merge installation default keywords and functions that may be set when the program is generated.

Keywords	Functions
ALTSEQ	Alters the normal EBCDIC collating sequence.
BLKSET	Bypasses or selects FLR-Blockset.
CHALT	Translates format CH the same as format AQ, or translates format AQ only.
CHECK	Suppresses record count checking for sorting applications that use the E35 user exit routine without a SORTOUT data set.
EQUALS	Preserves the input order of equally collating records.
ERETINV	Terminates sort/merge with a return code of 16 or an ABEND for a dynamically invoked program.

ERETJCL Terminates sort/merge with a return code of 16 or an ABEND for an EXEC-initiated program.

EXCPVR Uses EXCPVR for SORTWK I/O.

LIST Lists program control statements.

MAXLIM Sets an upper limit to amount of address space available for sorting.

MINLIM Sets a minimum limit to amount of address space available for sorting.

MSGS Controls printing of program messages.

PRINT Specifies an alternate name for print data sets; otherwise, SYSOUT is used.

RELEASE Releases unused work space.

RESALL Reserves storage for system and application use.

RESDNT Indicates whether sort/merge modules reside in link pack area.

RESINV Reserves space for programs invoking sort/merge.

SECALL Allows automatic secondary allocation of temporary work space.

SIZE Sets maximum amount of main storage.

SMF Produces SMF records.

SORTLIB Generates a SORTLIB.

SVC Specifies a user SVC number for sort/merge.

SYSTEM Generates an OS, VS1, SVS, or MVS version of sort/merge.

VBLKSET Bypasses or selects VLR-Blockset.

VERIFY Verifies sequence of output records.

VIO Indicates whether virtual allocation of work data sets is accepted.

The PARM field options of the EXEC job control statement allow you to override some of the specifications made at sort generation time, such as the amount of main storage allocated for program execution and the handling of error messages.

The OPTION statement also provides you with the ability to override SORT statement parameters that are either in a parameter list of a dynamically invoked sort or in the SORTIN data set. See the OPTION control statement in Section 4 and the SORTCNTL DD statement in Section 5 for details.

You can also obtain certain diagnostic information for use as a debugging aid: in the case of a disk sort, by using the DEBUG control statement (see Appendix A); for a tape sort or merge, through use of the PARM field of the EXEC statement (see Section 5), or through the passed parameter list.

MACHINE REQUIREMENTS

The program requires the following machine equipment for execution:

- Any System/370, 303x, 3801, or 4341 processor supported by an OS/VS or OS operating system.
- Any units that are required for input and output in addition to the above. These units must be supported by QSAM or VSAM.
- The 3880 Model 2 or 3 with the Speed Matching Buffer Feature to permit attachment of the 3380 to systems with block multiplexor channels with data rates less than 3 megabytes per second.
- Any additional units required as intermediate storage for a sort. Intermediate storage requirements are given in Section 3.

MAIN STORAGE REQUIREMENTS

In general, the more main storage you can make available to the program, the better the performance. However, problems can arise under OS/VS if an unduly large virtual region or partition is assigned, if no maximum limit to sort storage was set at installation time. See "Main Storage" in Section 8. The minimum is 54K bytes.

Sort main storage is defined when the sort/merge program is generated. If this is not suitable, calculate the requirements for your particular application and override the amount specified using the SIZE parameter on the EXEC card (see Section 5) or in the passed parameter list. To work out your requirements, see Section 3 under "Main Storage."

PROGRAM EXECUTION

To execute the sort/merge program, you must prepare two types of statements: program control statements and job control language (JCL) statements. Program control statements are processed by the sort/merge program; they describe your records and how you want them sorted. JCL statements are processed by the operating system control program; among other things, they describe your input and output data sets and your intermediate storage requirements.

A summary of which statements are needed under what circumstances is given in Section 2, which provides a step-by-step guide to control statement preparation.

PROGRAM CONTROL STATEMENTS

Eight program control statements are used by the program: the SORT, MERGE, OPTION, MODS, RECORD, ALTSEQ, DEBUG, and END statements. These control statements are your way of giving the program necessary information. You will find a full discussion of the program control statements in Section 4.

JCL STATEMENTS

JCL statements are used to initiate execution of the sort/merge program and describe to the operating system the data sets required by the program.

A complete description of the format and of the specifications for the JCL statements required by the program is contained in Section 5 of this publication.

A sort usually requires intermediate storage as working space during program execution; you must specify intermediate storage device(s) and the work space required in certain data definition statements—unless you use the DYNALLOC facility under MVS. The formulas for determining space requirements are described in Section 3. A merge does not require intermediate storage.

PROGRAM INITIATION

You can initiate execution of the program in the following ways:

- In the input stream with an EXEC job control statement using the name of the program or the name of a cataloged procedure, as described in Section 4 of this publication.
- In a program written in Basic Assembler Language with a system macro instruction, as described in Section 7 of this publication.
- In programs written in either COBOL or PL/I with a special facility of the language. For more information, see the programmer's guide describing the compiler version available at your installation.

PROGRAM MODIFICATION

During execution, the program can pass control at various points, known as program exits, to routines you have designed and written to perform specific functions. For example, you can write such routines to summarize, insert, delete, shorten, or otherwise alter records as they are being sorted or merged. You can also write your own routines to correct I/O errors that the control program cannot handle or to perform any necessary abnormal end-of-task operation before the program is terminated.

You can include your routines as an object deck in the input stream at execution time, or they can reside in a private library.

The program exits and their uses are explained in Section 6.

RETURN CODES

Sort/merge returns a return code of 0 to the operating system (or other invoking program) upon successful completion. If completion is unsuccessful, a return code of 16 is returned or an ABEND is issued, depending on what was specified at installation time. See Section 7.

CHECKPOINT/RESTART

Checkpoint/restart is a facility of the operating system which permits an automatic or deferred restart if the sort/merge program abnormally terminates. You must specify certain parameters in the program control statements and prepare a JCL DD statement if you wish to include this facility in a sort/merge execution. See "CKPT" in Section 4.

Note: If checkpoint/restart is specified, the Blockset techniques will be bypassed by the sort/merge program.

For more information on the checkpoint/restart facility, see OS/VS1 Checkpoint/Restart or OS/VS2 MVS Checkpoint/Restart.

STATISTICAL DATA COLLECTION

If you want to collect statistical data concerning execution time, record distribution, and so on, you can use the SMF installation option. SMF is a keyword operand of the ICEMAC installation macro. Users who have properly installed and initialized the sort/merge program under an MVS or VS1 programming system (SMF is not supported under OS or SVS) have this option available to them.

If SMF is specified, sort/merge causes an SMF record to be written for each sort which completes successfully (return code 0). If an SMF record is desired, either a short or full SMF record can be produced by means of the SMF keyword on the ICEMAC installation option. A full SMF record will only be produced by sort/merge if requested (SMF=FULL), and only if the sorting operation is for variable-length records.

Notes:

1. If you want sort/merge to produce SMF records under the MVS programming system, a new SVC routine for sort/merge must be installed. If SMF records under the VS1 programming system are desired, a modified SVC routine for sort/merge must be installed.

2. Meaningful SMF records are produced only when sort/merge selects Peerage, Vale, or one of the Blockset techniques. If one of the conventional sorting techniques, such as BALN, is selected, an SMF record will be produced without any statistical data.

For more information concerning statistical data collection, see OS/VS1 System Management Facilities (SMF) or OS/VS2 MVS System Programming Library: System Management Facilities (SMF).

MAXIMUM EFFICIENCY

The specifications you make in your program control and JCL statements affect program execution, efficiency, and speed. Suggestions for improving the performance of a sort/merge application are given in Section 8.

When you are designing your sort application, remember that the program can use many I/O devices for input, output, and intermediate storage. You should assign the program a relatively high priority to be sure that it gets control of the processor frequently and does not tie up the I/O devices while it waits for processor time.

SECTION 2. WRITING A SIMPLE PROGRAM

Figure 3 is a simple, step-by-step guide, including an example, to preparing your control statements for a program application. However, all the options and features of the program are not covered in Figure 3 on page 10. Some of those not covered are:

- The PARM option of the EXEC statement, which permits you to override some of the specifications made at sort generation time, select a specific distribution technique for tape, and obtain special diagnostic information. The PARM option is described in detail in Section 5.
- The program exits, whose purpose and uses are described in Section 6.
- The checkpoint/restart facility, which permits an automatic or deferred restart if the program terminates abnormally. See checkpoint/restart in the index for more information.
- Achieving maximum program efficiency, which is explained in Section 8.
- Initiating the program with a system macro instruction from within one of your own assembler language programs, which is described in Section 7.
- Use of the DEBUG control statement, which is described in Appendix A.

When you have prepared your control statements, collate them as described in Section 5, "Job Control Statements" (Figure 10).

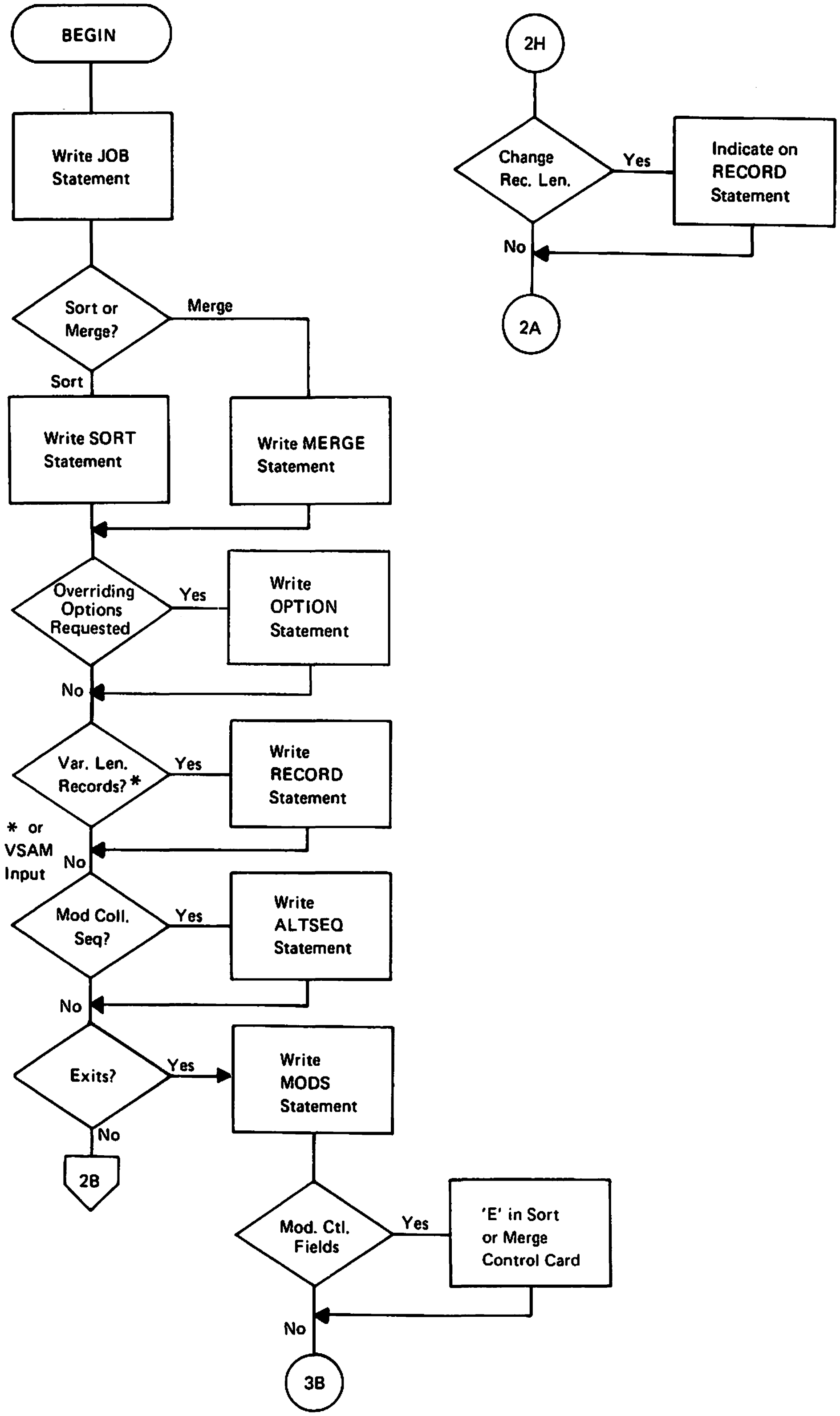


Figure 3 (Part 1 of 3). Step-by-Step Guide to Preparing Control Statements

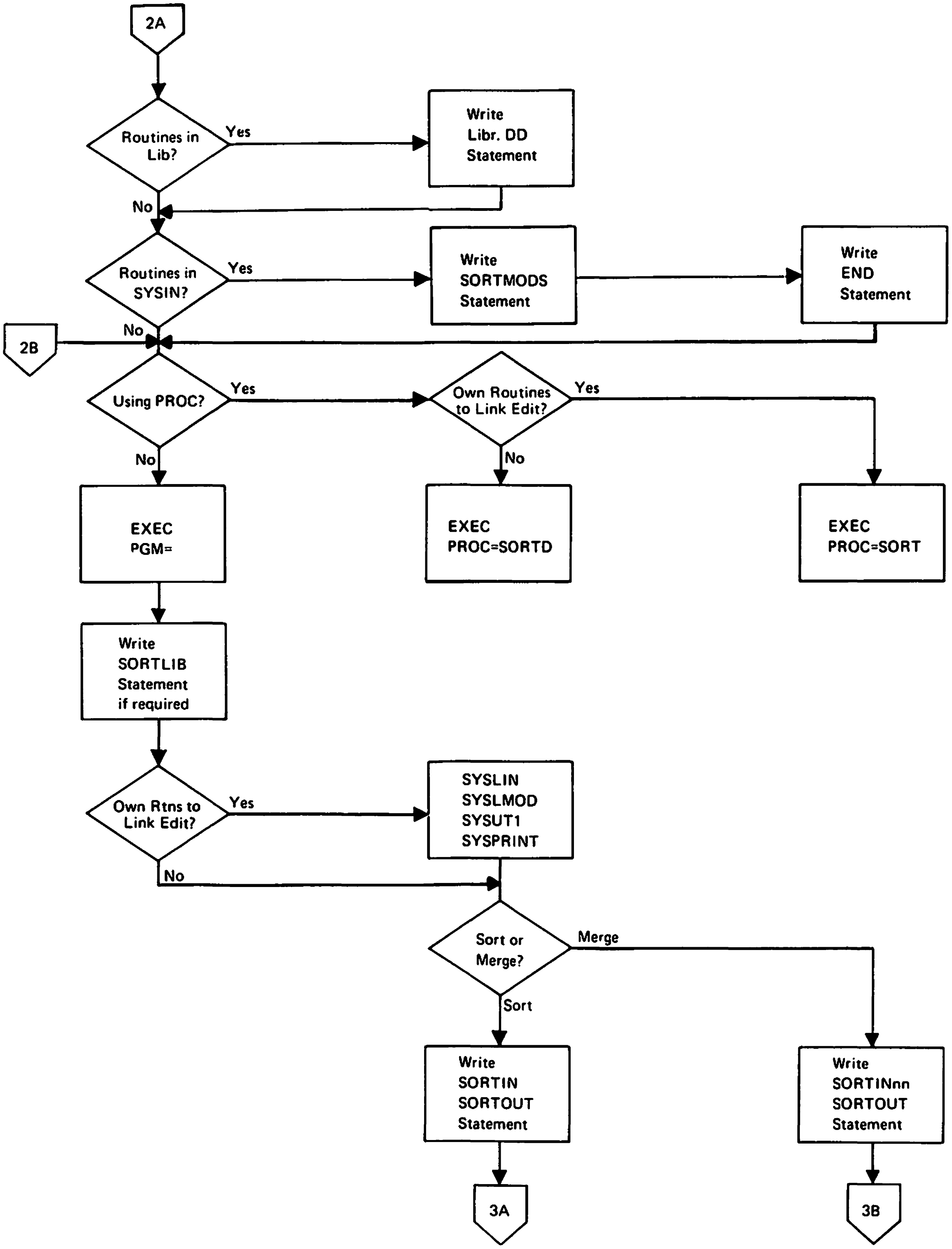


Figure 3 (Part 2 of 3). Step-by-Step Guide to Preparing Control Statements

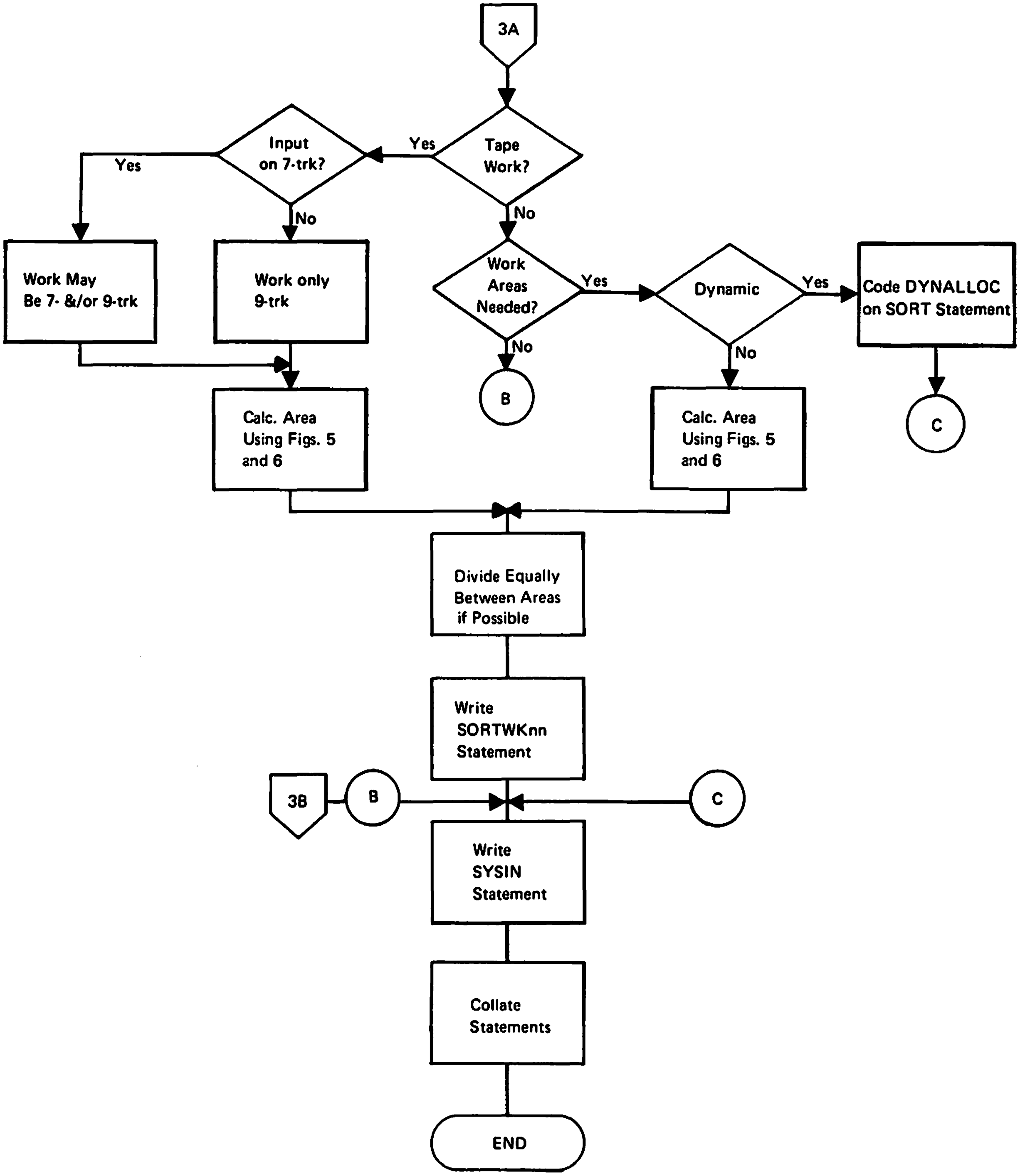


Figure 3 (Part 3 of 3). Step-by-Step Guide to Preparing Control Statements

CONTROL STATEMENT EXAMPLE

The following example shows the JCL and sort/merge statements required for a simple sort application. Other examples are described in Appendix D.

```
//EXAMP    JOB  A402,PROGRAMMER,REGION=256K                         01
//SRT     EXEC  PGM=SORT,PARM='SIZE(MAX)'                           02
//SYSOUT    DD  SYSOUT=A                                            03
//SORTIN    DD  UNIT=3380,VOL=SER=000101,DISP=SHR,DSN=INPUT         04
//SORTOUT   DD  UNIT=3400-3,DSN=OUTPUT,VOL=SER=222222,              05
//              DISP=(,KEEP)                                        06
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(10))                          07
//SORTWK02 DD  UNIT=SYSDA,SPACE=(CYL,(10))                          08
//SYSIN     DD  *                                                   09
   SORT  FIELDS=(5,12,CH,A),FILSZ=E2000                             10
/*
```

01 The JOB statement introduces this job to the operating system, and specifies a region of 256K bytes.

02 The EXEC statement calls the program by its alias SORT and specifies that the program should use all the main storage available to it.

03 The SYSOUT DD statement directs the sort messages to system output class A.

04 The SORTIN DD statement describes an input data set named INPUT. The data set is on a 3380 disk with the serial number 000101. The DISP parameter indicates that the data set is known to the operating system.

05-06 The SORTOUT DD statement describes the output data set. Output will be recorded on a 9-track tape and will be kept. The data set will be placed on a standard label tape with tape volume number 222222. By default, format, record length and block size are the same as for SORTIN.

07-08 These DD statements define temporary work data sets. The two data sets are on SYSDA direct access devices. Ten cylinders are specified for each data set.

09 A data set follows in the input stream.

10 SORT statement. The FIELDS operand describes one field. It begins on byte 5 of each record, is 12 bytes long, contains character (EBCDIC) data, and is to be sorted into ascending order. The file size is estimated to be 2000 records.

SECTION 3. CALCULATING STORAGE REQUIREMENTS

This section describes how to calculate the amount of main storage needed to run a sort or merge. It then describes how to calculate the amount of space which a sort may need as intermediate storage on tape or disk.

MAIN STORAGE

In general, the more (virtual) main storage you make available to the program (up to a certain limit), the better the performance. For the program to be efficient, at least 72K bytes of main storage should normally be used, but to obtain best performance always try to allocate between 128K bytes and 512K bytes of main storage, depending on file size. However, the amount of virtual storage should be related to the amount of real storage available to the program. As a guideline, use the total real storage available for page frames divided by the usual number of initiators in the system.

The amount of main storage to be made available to sort/merge is defined when the program is installed. If for any reason this default value is unsuitable, you can override it with the SIZE parameter of the EXEC statement, as described in Section 5.

You can calculate the minimum main storage requirement (in bytes) for sort/merge by using the formula:

(1.2 x MIN) + 8K + m (EXEC-initiated sort)

or

(1.2 x MIN) + 8K + m + reserved space (dynamically invoked sort)

where

MIN
: is the space needed for sort itself, and is calculated using the formula given in Figure 4. The constant 1.2 provides for space lost through fragmentation, and the additional 8K bytes is used by the system.

m
: is the number of bytes of main storage that your exit routine(s) uses. It is the maximum "m" value you specified on your MODS control statement.

reserved space
: is that space required by the invoking program for data handling. The number and size of buffers you need depends upon what routines you have, how the data is stored, and which access method you use.

For example, a COBOL-invoked sort requires a number of bytes to be reserved for COBOL's use in its default or user-written input/output routines, which are normally needed at execution time for OPEN/CLOSE modules and for buffers.

Formula	MIN = A + BLK + (C x LEN)			
	A	BLK	C	LEN
SORTIN Standard disk sort technique	50000	(Maximum) input block size	5	Input LRECL
BALN (disk)	13000		5	
CRCX	20000		IS	
BALN (tape), POLY	13000		5	
OSCL	20000		max(5,IS)	
SORTOUT Standard disk sort technique	50000	Output block size	4	Output LRECL
BALN (disk)	13000		IS	
CRCX	20000		IS	
BALN (tape)	13000		(IS + 1)/2	
POLY	13000		IS	
OSCL	20000		IS	
MERGE	12000	Output block size	No. of input data sets	(Max) input block size
IS: Number of intermediate storage areas				
For a Sort:	Apply formula to both SORTIN and SORTOUT, and take the greater.			
For a Merge-only:	Apply formula to MERGE.			
Spanned records:	Add space for assembling the records (=LRECL) for each data set containing spanned records.			

Figure 4. Calculating Main Storage Requirements

Notes:

1. At least 54K bytes should be allocated to the program.

2. If you are using VSAM data sets, you must allow space for VSAM's buffer pools (maximum of input and output for a sort, total of input and output for a merge), and for VSAM control blocks. Refer to Planning for Enhanced VSAM for details of how to calculate the amounts required.

3. For a disk sort, if the MINLIM value specified at installation time is larger than a given SIZE value for a certain application, the MINLIM value will be used.

4. Dependent upon main storage fragmentations and system usage in a region or partition, the System Measurement Facility (SMF) may log more storage than was actually used.

5. For calculating the amount of storage necessary to execute VLR-Blockset, see Appendix E.

INTERMEDIATE STORAGE

Most sorting applications need work space on disk or tape. Merge applications need none. The amount of space required depends on the type of device on which you assign storage, the number of records in your input data set, and the amount of main storage assigned to the program.

STORAGE DEVICES

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

IBM 2400 and 3400 series magnetic tape units can be used for intermediate storage. 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 direct-access devices. 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 tapes for intermediate storage, you cannot use the data converter, nor can you use the translation feature for anything but character data.

Unless you force one of the nonstandard techniques, you can specify a mixture of direct access devices for a given sort application. The types of device available for intermediate storage are:

IBM 2314/2319 disk
IBM 3330/3333 series disks (Model 1 and/or Model 11)
IBM 3340/3344 disk
IBM 3350 disk
IBM 3375 disk
IBM 3380 disk
IBM 3850 MSS

Note: The 3880 Model 2 or 3 with the Speed Matching Buffer Feature permits attachment of the 3380 to systems with block multiplexor channels with data rates less than 3 megabytes per second.

SPACE REQUIREMENTS

Space requirements are summarized in Figures 5 and 6.

Tape Techniques	Maximum Input	Work Storage Areas Required	Max.No.of Work Areas	Comments
Balanced tape BALN	15 reels	Min=2(V+1) tape units	32 reels	Used if >3 work storage tapes provided and file size not given
Polyphase tape POLY	1 reel	Min=3 tape units	17 reels	Used if 3 work storage tapes provided
Oscillating tape OSCL	15 reels	Min=V+2 or 4 tape units, whichever is greater	17 reels	File size must be given. The tape drive containing SORTIN cannot be used as a work unit

Key

V No. of input volumes if blocking equals work storage blocking

Figure 5. External Work Storage Requirements of the Various Tape Techniques

TAPE

Three different techniques are available to the program: the BALN, POLY, and OSCL techniques. To calculate their requirements, see Figure 5.

Note: The value you obtain for "min." is literally a minimum value; if, for example, your input uses a more efficient blocking factor than the sort program or is spanned, you will need more intermediate work space.

DIRECT ACCESS

Formulas for calculating requirements are given in Figure 6.

Divide the number of tracks or cylinders evenly among the areas you select. The formulas are based on areas of equal size, and more tracks will be needed if you do not divide them equally.

System performance is improved if storage is specified in cylinders rather than tracks. The number of tracks per cylinder is 19 for the 3330 series, 20 for the 2314, 12 for the 3340, 30 for the 3350, 12 for the 3375, and 15 for the 3380. FLR-Blockset will be bypassed if space is not allocated in cylinders (MVS only).

The program will allocate secondary extents as required on up to 12 work areas, even if not requested in the JCL, if sort/merge has been installed with the option SECALL=YES, unless the data set is virtual I/O.

Tracks not required when merging begins are automatically released if the RELEASE=YES installation option is selected (unless work data sets have been defined as permanent rather than temporary).

Release is not done for in-main-storage sorts or skip merge. The sort/merge program may do an in-main-storage sort if enough main storage is available to hold all the records.

Disk Techniques	Maximum Input	Work Storage Areas Required	Max.No. of Work Areas	Comments
Standard (default) disk techniques	No fixed maximum -depends on available main storage and work storage	No areas needed if enough main storage available If areas needed, minimum no. of tracks = ((FxS)/K)+N Allocate extents in cylinders to get best performance.	100 areas	Secondary extents will be automatically allocated when needed, if allowed at the installation.*
Balanced direct access BALN		3 areas Minimum number of tracks= ((SxN)/Kx(N-1))+2N	6 areas	Can be forced when 3-6 work areas provided
Crisscross direct access CRCX		6 areas Minimum number of tracks = (1.25xS)/K	17 areas	Can be forced when 6-17 work areas provided

Key

B   Work storage track utilization: 7000 for 2314/2319, 12000 for 3330 series, 8000 for 3340, 18000 for 3350, 45000 for 3380

F   Multiplication factor as follows:

Blockset
1.8 if >100K bytes main storage available
1.9 if <100K bytes main storage available

Peerage and Vale
1.05 if >100K bytes main storage available**
1.10 if 50-100K bytes main storage available**
1.50 if <50K bytes main storage available

*For Blockset, always allow for secondary extents. Blockset work space requirements can exceed the amount calculated in the formula, depending on the randomness of the input data and the length of the control fields.

**If work device types are mixed and/or input records are fixed-length and long (more than a quarter of work track length but less than a full track), then F should be increased towards 1.50.

K   B/L (≥ 1; only integer part used)

L   (Max.) input record length which should be increased by the length of each control field with any of the following formats:

ZD	zoned decimal
AC	character ASCII
AQ	alternative collating sequence

or if a control field is to be modified, that is,

E   is specified as the sequencing order

N   No. of work areas

S   No. of records to be sorted (FILSZ)

Figure 6. External Work Storage Requirements of the Various Disk Techniques

More space than indicated may be needed:

- If you have a long control word. As a rule of thumb, add 5% for every 150 bytes of control word after the first 100 bytes.
- If you have a mix of work devices. In most cases, if intermediate storage disks are mixed, additional work space should be allocated.
- If your application modifies control fields, requires alternative sequencing (ALTSEQ), or uses zoned decimal control fields, then L in the formulas in Figure 6 should be increased by the length of such control fields.
- If you specify the CKPT operand on the SORT control statement, 20-30% of the primary allocation of SORTWK tracks is set aside for checkpoint processing.

Example

Determine minimum requirements when sorting 10,700 eighty-byte records using three areas on 3330, with 120K bytes of main storage available to the program. Normally, the Blockset technique will be used for fixed-length records.

K = 12,000/80 = 150

F = 1.80

Min. = 1.8 x 10,700/150 + 3 = 132

Divided among three areas: 44,44,44. For greater efficiency, allocate in cylinders, for example, three areas of two cylinders each.

EXCEEDING INTERMEDIATE STORAGE CAPACITY

At the beginning of a sorting operation, the sort/merge program estimates a maximum sorting capacity (Nmax) and generates an informative message: ICE092I or ICE093I for a standard disk sort, ICE038I for a tape or nonstandard disk sort.

The message gives the approximate capacity in number of records. With disk work space, the value is based on use of only the first extent of work data sets. For variable-length records the value is based on the maximum record length.

The value printed in message ICE038I is an average value rounded down to the nearest thousand. This value assumes random input. If you have a reversed sequenced file and tape work storage, sort capacity may be exceeded at a lower value, because of the higher number of partly empty end-of-string blocks.

If, during the course of sorting, the allocation of secondary space on one of the sort work data sets fails, the system will issue a B37 informational message. Sort/merge can recover from this by allocating space on one of the other work data sets, if one is available.

Work Storage on Disk

Since the program uses secondary extents for up to 12 work areas even if not requested in the JCL (unless you force one of the nonstandard techniques), the probability of exceeding intermediate storage capacity is very low. However, if this happens for a nonstandard disk sort, the program gives control to your routine at exit E16, if available. This routine can direct the program to take one of the following actions:

- Continue sorting with only part of the input data set; the remainder could be sorted later and the two results merged to complete the application.
- Terminate the program without any further processing.

Work Storage on Tape

Note that for magnetic tape, a tape length of 2400 feet is assumed in calculating Nmax, so for tapes of other lengths the figure will not be correct. When tapes with mixed density are used, the smallest density is used in the calculation.

If you specify an actual data set size, and that size is larger than the maximum capacity estimated by the program (Nmax), the program terminates before beginning to sort. If you specify an estimated data set size, or none at all, and the number of records reaches the maximum (Nmax), the program gives control to your routine at exit E16, if you have written and included one. This routine can direct the program to take one of the following actions:

- Continue sorting the entire input data set with available intermediate storage. If the estimate of the input data set size was high, enough intermediate storage may remain to complete the application.
- Continue sorting with only part of the input data set; the remainder could be sorted later and the two results merged to complete the application.
- Terminate the program without any further processing.

Program Action

If you do not include an E16 routine, the program continues to process records for as long as possible. 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 program generates a separate message for each of the three possible error conditions. They are:

ICE041A - N GT NMAX: Generated before sorting begins (for a tape sort) when the exact data size supplied on a SORT control statement is greater than Nmax.

ICE046A - SORT CAPACITY EXCEEDED: Generated when the sort has used all available intermediate storage while processing.

ICE048I - NMAX EXCEEDED: Generated when a tape sort has exceeded Nmax and has transferred control to a user-written E16 routine for further action.

The test for message ICE041A is made with the maximum possible calculated value, that is, sort/merge is sure it will fail. In case of doubt, the message will not be issued.

SECTION 4. PROGRAM CONTROL STATEMENTS

This section tells you how to write the sort/merge program control statements. In these statements, you describe the input data, provide information about the control fields to be used, and describe any of your own routines you wish to be used during program execution. For a full explanation of program exits, and how you can use your own routines during a sort/merge application, see Section 6.

There are eight control statements:

SORT statement   Provides information about control fields and data set size. Use this statement if your application is a sort.

MERGE statement   Provides the same information as a SORT statement. Use this statement if your application is a merge.

OPTION statement   Provides an alternate way to specify or modify certain program options available at installation time (such as EQUALS, CHALT, CHECK, and VERIFY) or on the SORT control statement (such as CKPT or DYNALLOC).

RECORD statement   Provides record length and type information. This statement is required when you include user routines that change record lengths during sort/merge execution, when there is no SORTIN DD statement, or when input is a VSAM data set. It can be supplied at other times to improve efficiency.

MODS statement   Links your routines with the related sort/merge program exits. This statement is required only when you include user routines in a sort/merge application. A description of how to write such routines and how they may be used in a sort/merge application is contained in Section 6.

ALTSEQ statement   Specifies modifications to the IBM EBCDIC collating sequence. The modified sequence will be used for any control field whose format is specified as AQ.

DEBUG statement   For use with a disk sort when detailed information on program execution is required for optimization, debugging or bypassing purposes.

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

The program checks the validity of each statement before processing. If the program finds an error, it issues a diagnostic message. (See Appendix C for descriptions of messages.)

An overview of the format and parameters of all the program control statements is given in Figure 7.

NOTATIONAL CONVENTIONS

A uniform system of notation describes the format of the job control language and sort/merge control statements. This notation is not part of the language; it simply provides a basis for describing the structure of the commands.

The command-format illustrations in the following figure use these conventions:

- Brackets, [], indicate an optional parameter.
- Braces, {}, indicate a choice of entry; unless a default is indicated, you must choose one of the entries.
- Items separated by a vertical bar, |, represent alternative items. No more than one of the items may be selected.
- An ellipsis, ..., indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (parentheses, commas, apostrophes, etc.) must be entered as shown.

Operations	Operands
SORT|MERGE	{FIELDS=(p,m,f,s...,p,m,f,s)| FIELDS=(p,m,s...,p,m,s),FORMAT=f} [,FILSZ=x|,SIZE=y] [,SKIPREC=z] [,CKPT] [,EQUALS|,NOEQUALS] [,DYNALLOC=d|,DYNALLOC=(d[,n])]

Parameter	Explanation	Notes
FIELDS=	Description of control fields	Fields must be described in descending order of significance.
p	Position within record	All fields except binary must start on a byte boundary. No field may extend past byte 4092.
m	Length	Acceptable control field lengths (in bytes), and available formats are as follows:
f	Format	Length — Format, Description 1-4092 — CH (character EBCDIC, unsigned) 1-256 — If CHALT=YES is specified, CH is treated the same as AQ. 1-32 — ZD (zoned decimal, signed) 1-32 — PD (packed decimal, signed) 1-256 — FI (fixed-point, signed) 1 bit-4092 bytes — BI (binary, unsigned) 1-256 — FL (floating-point, signed) 1-256 — AC (character ASCII, unsigned) 2-256 — CSL (signed numeric, leading separate sign) 2-256 — CST (signed numeric, trailing separate sign) 1-256 — CLO (signed numeric, leading overpunch sign) 1-256 — CTO (signed numeric, trailing overpunch sign) 2-256 — ASL (signed numeric, ASCII, leading separate sign) 2-256 — AST (signed numeric, ASCII, trailing separate sign) 1-256 — AQ (character EBCDIC, alternative collating sequence) The sum of lengths must not exceed 4092 bytes. For format details, see Appendix B.
s	Desired sequencing	Must be one of the following: A - ascending D - descending E - user-modified control field that can be sorted in ascending order

Figure 7 (Part 1 of 7). Program Control Statements

Parameter	Explanation	Notes
FORMAT=f	Optional; may be used when all control field data formats are the same.	f must be one of the available formats listed above under FIELDS=f.

FILSZ=x
SIZE=y
	Optional; the number of records to be sorted.	If x is an estimate, the value must be preceded by the character E (FILSZ=Ex). If SIZE is used instead of FILSZ, the value should represent the number of records in the input file.

SKIPREC=z	Optional; the program will skip z records before sorting.	Ignored if specified for a merge.
CKPT	Optional; checkpoints are taken.	The spelling CHKPT is also accepted. Checkpoints cannot be taken during: • A merge-only operation with VSAM output • An invoked merge handling output through E35.
EQUALS NOEQUALS	Optional; order of equals.	Specifies that the order of equally collating records need not be preserved from input to output. Ignored if specified for a merge.
DYNALLOC=	Optional; dynamic allocation of intermediate work storage.	Valid only for MVS. Ignored if specified for a merge.
d	Device type.	D can be any of 2314, 3330, 3330-1, 3340, 3375, 3380, 3350, 3400-3, 3400-4, 3850, 2400, 2400-3, 2400-4, or their user-assigned group name, such as SYSDA.
n	Number of devices (work areas).	Number of work data sets (up to 100).

Figure 7 (Part 2 of 7). Program Control Statements

Operation	Operands
OPTION	[FILSZ=x|SIZE=y] [,SKIPREC=z] [,CKPT] [,EQUALS|,NOEQUALS] [,DYNALLOC=d|,DYNALLOC=(d[,n])] [,CHALT|,NOCHALT] [,VERIFY|,NOVERIFY] [,CHECK|,NOCHECK] [,BLKSET|,NOBLKSET]

Parameter	Explanation	Notes

FILSZ=x
SIZE=y
	Optional. The number of records to be sorted.	If x is an estimate, the value must be preceded by the character E (FILSZ=Ex). If SIZE is used instead of FILSZ, the value should represent the number of records in the input file. Overrides number in SORT statement.

SKIPREC=z	Optional. The program will skip z records at the beginning of the input data set.	Ignored if specified for a merge. Overrides number in SORT statement.
CKPT	Optional. Checkpoints are taken.	The spelling CHKPT is also accepted. Checkpoints cannot be taken during a merge-only operation with VSAM output or during an invoked merge handling output through E35.
EQUALS NOEQUALS	Optional. Order of equals.	Specifies that the order of equally collating records need not be preserved from input to output. Ignored if specified for a merge.
DYNALLOC	Optional. Dynamic allocation of intermediate work storage.	Valid only for MVS. Ignored if specified for a merge.
d	Device type.	D can be any of 2314, 3330, 3330-1, 3340, 3350, 3375, 3380, 3400-3, 3400-4, 3850, 2400, 2400-3, 2400-4, or their user-assigned group name, such as SYSDA.
n	Number of devices (work areas).	Number of work data sets (up to 100). Overrides number in SORT statement.
CHALT NOCHALT	Optional. Specifies both formats AQ and CH, or AQ only.	Specifies that both formats AQ and CH control fields be translated through the alternate collating sequence (ALTSEQ) translate table (CHALT), or only format AQ control fields (NOCHALT). Overrides installation values.
VERIFY NOVERIFY	Optional. Sequence checking.	Specifies that sequence checking on final output record sequence should or should not be done. Overrides installation values.

Figure 7 (Part 3 of 7). Program Control Statements

Parameters	Explanation	Notes
CHECK NOCHECK	Optional. Check record counters.	Specifies that record counters should or should not be checked at the end of program execution. The CHECK/NOCHECK specification is only valid for applications with output record processing in an E35 exit routine. Overrides installation values.
BLKSET NOBLKSET	Optional. Attempt to use or bypass Blockset techniques.	Specifies that sort/merge is to attempt to execute either the FLR-Blockset technique (for fixed-length records) or the VLR-Blockset technique (for variable-length records) or to bypass them.

Figure 7 (Part 4 of 7). Program Control Statements

Operation	Operands
RECORD	TYPE=x,[LENGTH=(L1,L2,L3,L4,L5)]

Parameter	When needed	Value	Default
TYPE=x	When all records are supplied via exit E15	x must be: F-(fixed length), V-(variable-length EBCDIC), or D-(variable-length ASCII)	SORTIN RECFM
LENGTH=	(For fixed-length records)		
L1	When no SORTIN DD statement supplied.	SORTIN LRECL*; otherwise, overridden to that value.	SORTIN LRECL*.
L2	When length changed at E15.	Length after E15.	Length specified for L1 (or default if not specified).
L3	When SORTOUT LRECL* different from SORTIN and no SORTOUT LRECL* available.	SORTOUT LRECL*; otherwise, overridden to that value.	SORTOUT LRECL*; if none exists, L1.
LENGTH=	(For variable-length records)		
L1	When no SORTIN DD statement supplied.	Maximum record length (plus 4 bytes if input is VSAM); otherwise, overridden to default.	SORTIN LRECL* (plus 4 bytes if input is VSAM).
L2	When maximum length changed at E15.	Maximum record length after E15 (plus 4 bytes if input is VSAM).	Length specified for L1 (or default if not specified).
L3	When SORTOUT LRECL* different from SORTIN, and no SORTOUT LRECL* available.	SORTOUT LRECL* (plus 4 bytes if input is VSAM); otherwise overridden to default	SORTOUT LRECL* (plus 4 bytes if input is VSAM).
L4	Aids optimization for a sort; not needed for a merge.	Minimum length (after E15), plus 4 bytes if input is VSAM.	Length to end of rightmost control field (≥ 18 bytes).
L5	Aids optimization for a sort; not needed for a merge.	Average length (after E15), plus 4 bytes if input is VSAM.	L5 = (L2 + L4)/2
*For a VSAM data set, the equivalent of LRECL is maximum record size (RECSZ).			

Figure 7 (Part 5 of 7). Program Control Statements

Operation	Operands
MODS	exit=(n,m,s[,e])...,exit=(n,m,s[,e])

Parameter	Explanation	Notes
exit=	The name of an exit to be activated.	Must be a valid exit name (for example, E28, E61). Up to 17 exit routines can be specified.
n	Name of the routine; member name if routine in a library.	
m	Size, in bytes, of the routine.	
s	Location of the routine.	Either the ddname of the data set containing the routine, or SYSIN.
e	Link-editing requirements.	e must be S, T, or N: S - routine to be link-edited separately. T - to be link-edited with other routines for same phase. T is the default. N - no additional link-editing needed.

Operation	Operands
ALTSEQ	CODE=(fftt,...fftt)

Parameter	Explanation	Notes
CODE=	Indicates that the collating sequence is to be modified.	Modifications are based on the EBCDIC sequence.
ff	The character whose collating position is to be changed.	Two hexadecimal digits in EBCDIC code (for example, Z is "E9").
tt	The position to be occupied by the characters ff.	Two hexadecimal digits (for example, "to collate after Z" would be "EA").

Figure 7 (Part 6 of 7). Program Control Statements

Operation	Operands
DEBUG	ABEND|NOABEND (Only valid for disk sort)

Parameter	Explanation	Notes
ABEND NOABEND	An unsuccessful run is to: -terminate with ABEND. -terminate with return code of 16.	Is used only for standard disk sort. Overrides the ERETJCL and ERETINV options specified at program installation time.
DUMP NODUMP	Recognized but ignored.	
Other parameters can be used, but are primarily intended for debugging purposes. They are described in Appendix A.		

Operation	Operands
END	none

The END statement must be used when user routines or data is in the input stream. It must come after all other program control statements.

Figure 7 (Part 7 of 7). Program Control Statements

CONTROL STATEMENT COMPATIBILITY

Six other control statements (INPFIL, OUTFIL, INCLUDE, OMIT, OUTREC, and SUM) that are used by other IBM sort/merge programs are accepted, but not processed. Since the OPTION control statement is now used by OS/VS sort/merge, any job streams from other IBM sort/merge programs that still contain an OPTION control statement will cause sort/merge to terminate unless the parameters conform to the new OPTION control statement.

The information contained in the INPFIL and OUTFIL statements is supplied to the program in DD statements. The functions of INCLUDE, OMIT, OUTREC, and SUM statements must be provided by routines at program exits.

The program will accept SORT, MERGE, RECORD, ALTSEQ, and END statements prepared for other IBM System/360 or System/370 sort/merge programs; any obsolete parameters will be ignored. However, because of the difference in parameter specifications, the program will not accept other programs' MODS control statements, with the exception of those used by the IBM Sort/Merge Program 360S-SM-023, and Program Product Sort/Merge 5734-SM1.

Note that, although applications using the 360S-SM-023 and 5734-SM1 programs can be successfully run using the OS/VS program, the reverse is not necessarily true, as this program provides facilities which the others do not.

CONTROL STATEMENT FORMAT

FULL CODING RULES FOR CONTROL STATEMENTS

All sort/merge control statements have the same general format, shown in Figure 8.

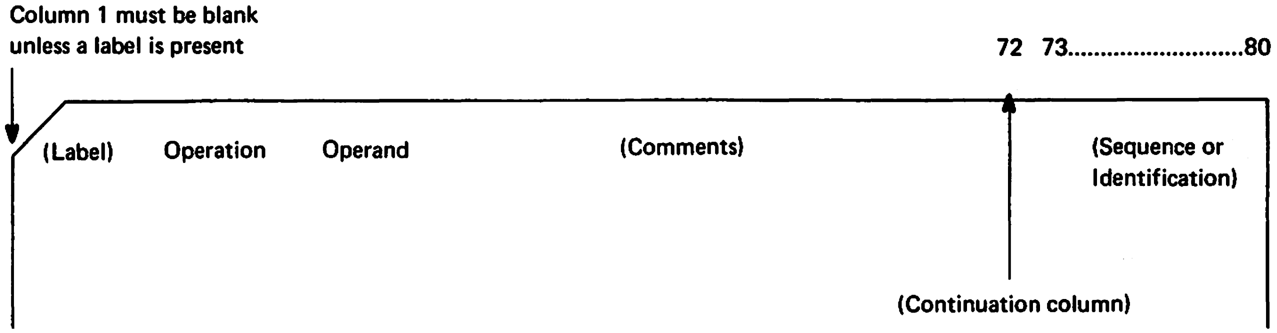


Figure 8. Control Statement Format

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, unless the first field is a label, in which case it must begin in column 1.

Label Field: If present, the label must appear first on the card. It must begin in column 1, and must conform to the operating system requirements for statement labels.

Operation Field: This field must not extend beyond column 71 of the first card. It contains a word (SORT, MERGE, OPTION, RECORD, MODS, ALTSEQ, DEBUG, or END) that identifies the statement type to the program. It must not begin in column 1. In the example below, 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 follow the operation field, and be separated from it by at least one blank. If the statement occupies more than one card, this field must begin on the first card. Each operand has 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
- keyword=value
- keyword=(value1,value2...,valuen)

The following example illustrates each of these formats.

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

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.

Continuation Column (72): Any character other than a blank in this column indicates that the present statement is continued on the next card. However, as long as the last character of the operand field on a card is a comma, the program will assume that the next card is a continuation card. The nonblank character in column 72 is required only when a comments field is to be continued or when a parameter is broken at column 71.

Columns 73 through 80: This field may be used for any purpose you desire.

Continuation Cards

The format of the sort/merge continuation card is shown in Figure 9.

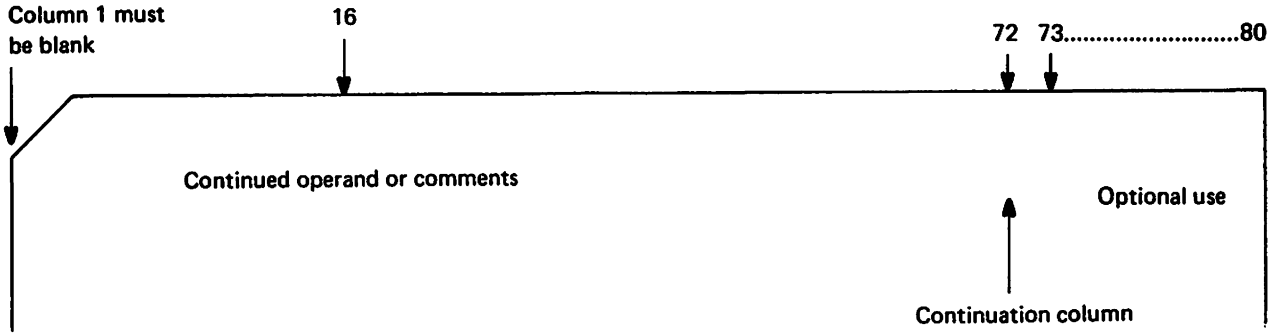


Figure 9. Continuation Statement Format

The continuation column and columns 73 through 80 of a continuation card fulfill the same purpose as they do on the first card of a control statement. Column 1 must be blank.

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:

- If a comments field is broken, column 72 must contain a nonblank character. The continuation must begin in one of columns 2 through 16.
- If an operand field is broken after a comma, the continuation column (72) can be left blank. The continuation must begin in one of columns 2 through 16.
- If an operand is broken at column 71, column 72 must contain a nonblank character. The continuation must then begin in column 16.

SUMMARY OF RESTRICTIONS

The following rules apply to control statement preparation:

- Unless a label is present, column 1 of each control statement must be blank.
- Labels must begin in column 1, and conform to operating system requirements for statement labels.
- The whole operation definer must be contained on the first card of a control statement.

- The first operand 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 (except for estimated data set size, which may contain nine characters).

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

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

Note: Control statement error conditions detected during scan will cause sort to transfer to Peerage/Vale to rescan the control statements.

SORT CONTROL STATEMENT

```
SORT    {FIELDS=(p,m,f,s...,p,m,f,s)|
        FIELDS=(p,m,s...,p,m,s),FORMAT=f}
        [,FILSZ=x|,SIZE=y]
        [,SKIPREC=z]
        [,CKPT]
        [,EQUALS|,NOEQUALS]
        [,DYNALLOC=d|
        ,DYNALLOC=(d[,n])]
```

The SORT control statement must be used when a sorting application is to be performed; this statement describes the control fields in the input records on which the program will sort.

SORT operands override options specified or generated by default at installation time; in turn, they can be overridden by parameters of the OPTION control statement. See also Figure 7 for a description of the format of the SORT control statement and a summary of the parameters it can contain.

FIELDS

The 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.

All control fields must be located within the first 4092 bytes of a record, and must not extend beyond the shortest record to be sorted. The collected control fields (comprising the control word) can be up to 4092 bytes long. As shown in Figure 7, the FIELDS operand can be written in two ways.

Use the first FIELDS operand format to describe control fields that contain different data formats; use the second format to describe SORT fields that contain data of the same format. The second format is optional; you can always use the first format if you prefer.

The program examines the major control field first, and it must be specified first. The minor control fields are specified following the major control field. In Figure 7, p, m, f, and s describe the control fields. The specifications for the parameters in the SORT control statement are summarized in Figure 7. The text that follows gives these specifications in detail.

p

specifies the beginning (high-order location) of a control field relative to the beginning of the record which contains the control field.

Note that the beginning of a variable-length record must include a 4-byte record descriptor word (RDW) which precedes the actual record. This is true even for VSAM input records, for which the sort/merge program will supply the necessary RDW on input to the program and remove it again at output (if output is to a VSAM data set). You should therefore always add four to the byte position in variable-length records.

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

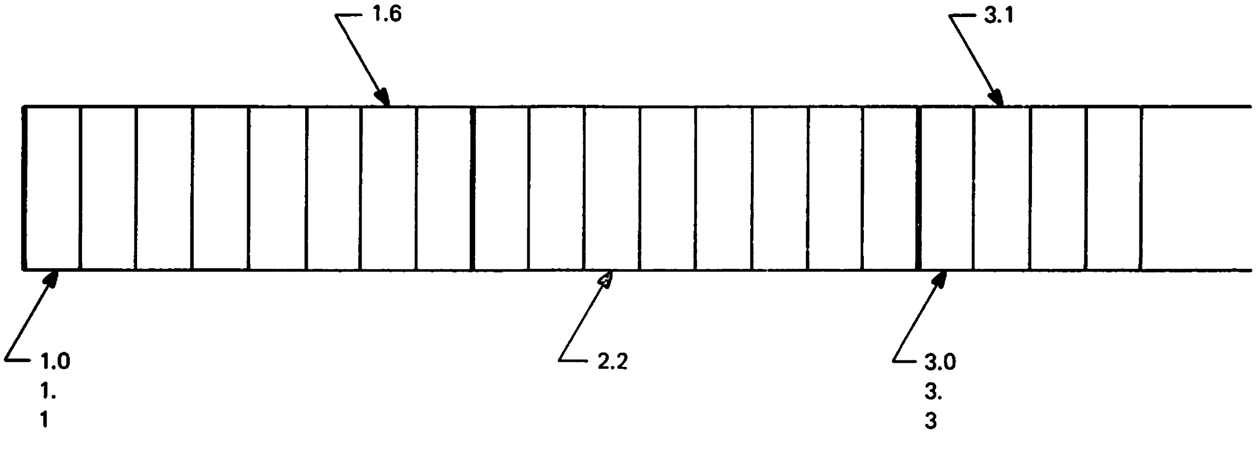
Fields containing binary values are described in a "bytes.bits" notation as follows:

- First, specify the byte location relative to the beginning of the record and follow it with a period.
- Then, specify the bit location relative to the beginning of that byte. Remember that the first (high-order) bit of a byte is bit 0 (not bit 1); 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 binary field falls on a byte boundary (say, for example, on the fourth byte), you can write it in one of three ways:

```
4.0
4.
4
```

Other examples of this notation are:



m

specifies the length of the control field. All control fields except binary must be a whole number of bytes long. Binary fields are expressed in the notation "bytes.bits". The length of a binary control field that is a whole number (d) of bytes long can be expressed in one of three ways:

d.0
d.
d

The number of bits specified must not exceed 7. A control field 2 bits long would be represented as 0.2.

The total number of bytes occupied by all control fields must not exceed 4092 (or, when the EQUALS option is in operation, 4088 bytes). When you determine the total, count a binary field as occupying 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. All fields must be completely contained within the first 4092 bytes of the record.

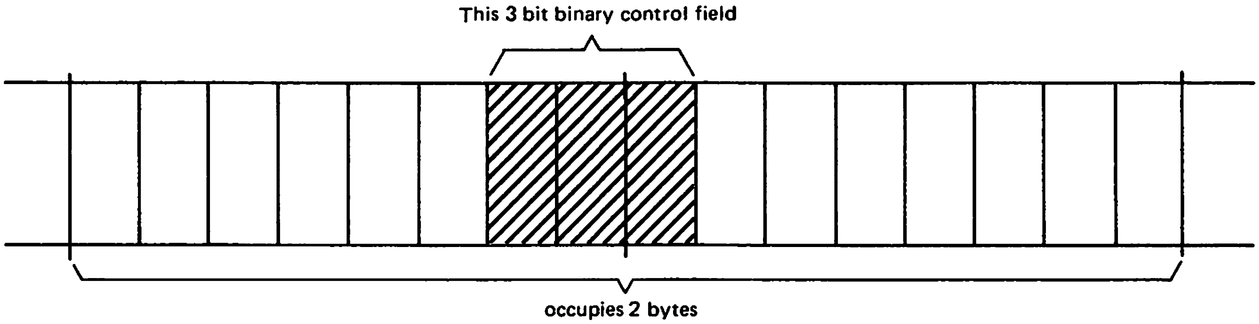


Figure 7 shows the maximum control field length for each format and indicates whether the format may be signed or unsigned.

f

specifies the format of the data in the control field. f can be any one of the two or three character abbreviations shown in the notes column in Figure 7.

If you specify more than one control field and all the control fields contain the same type of data, you can omit the f parameters and use the optional FORMAT operand, described below.

All floating-point data must be normalized before the program can collate it properly. You can use a routine of your own to do this at execution time, by associating it with one of the program exits. Specify the E option for the value of s in the FIELDS operand for each control field you are going to modify.

| See Appendix B for data format examples.

s

specifies how the control field is to be ordered. The valid codes are:

A—ascending order
D—descending order
E—control fields to be modified

Specify E if you include user routines to modify control fields before the program sorts them. After a user routine modifies the control fields, the program compares them logically and then sorts them into ascending order.

For information on how to add a user routine to modify a control field, see Section 6 of this publication.

Default: None; parameter must be specified.

FORMAT

FORMAT=f

f can be used to specify the format of the data described in the FIELDS parameter, if you specify more than one control field and the data in all the control fields is of the same format. The possible values of f are listed in Figure 7.

If you specify more than one control field, and the data in the several fields has different formats, you must specify an f parameter for each field instead of using FORMAT.

Default: None; must be specified if not included in FIELDS parameter.

FILSZ|SIZE

This parameter should always be specified. It is especially important if DYNALLOC is to be used.

FILSZ=x

x is the exact number of records to be sorted; it must take into account records to be inserted or deleted at exit E15, or skipped by SKIPREC.

SIZE=y

y is the exact number of records to be used as input, excluding any changes to be made at exit E15, or by SKIPREC (that is, the number of records in the SORTIN data set).

If the actual number of records is not the same as the value specified, the program will terminate with the value x or y placed in the IN field of the message ICE047A or ICE054I. This applies to both FILSZ and SIZE.

FILSZ|SIZE=En

n is the estimated number of records to be sorted and it must be immediately preceded by the letter E; it should in either case be large enough to include both the SORTIN data set and any records you may add at exit E15.

For example, if you estimate your total data set size to be 5000 records, specify FILSZ=E5000. The program will accept either FILSZ or SIZE, but FILSZ is preferable when its use is possible, as it allows better optimization for tape techniques and for disk techniques, when variable-length records are used. It should also be specified when using dynamic allocation.

If you omit the FILSZ or SIZE operand, the 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 (only for nonstandard disk techniques).

- If input is a VSAM data set (or sets), data set size is equal to that given in the VSAM catalog. Always specify FILSZ, therefore, if you want to add or delete records at E15.

Default: None; optional but recommended. Can be overridden by FILSZ|SIZE specified on the OPTION statement.

SKIPREC

SKIPREC=z

z is the number of records you want to skip before starting to process the input data set, and will usually be used if, on a preceding sort run, you have sorted only part of the input data set.

A program with an input data set which exceeds intermediate storage capacity will normally terminate unsuccessfully. However, for a tape or nonstandard disk sort, you can use a routine at E16 (as described in Section 6) to instruct the program to sort only those records already read in. It will then print a message giving the number of records sorted. You can use SKIPREC in a subsequent sort run to sort the remaining records, and then merge the output from different runs to complete the application.

Note: If SKIPREC is specified, the Blockset techniques are bypassed by the sort/merge program.

Default: None; optional. Can be overridden if SKIPREC is specified on OPTION statement.

CKPT

CKPT (the spelling CHKPT is also accepted) causes the program to activate the checkpoint/restart facility of the operating system. No checkpoints can be taken:

- If an invoked merge is handling output through exit E35
- If output from a merge-only operation is to be a VSAM data set
- In any user routine at a program exit

If this parameter is specified, the program takes the following checkpoints:

1. Start of sort phase (all tape techniques)
2. Start of each intermediate merge phase pass (balanced and polyphase tape technique); or at intervals during the intermediate merge phase (oscillating tape and all disk techniques)
3. Start of final merge phase

When you use the checkpoint/restart facility, you must write a JCL statement to define a data set for the checkpoint records. How to write this JCL statement (//SORTCKPT) is described in Section 5. In addition, you may need to specify more intermediate storage. See Section 3.

Note: If checkpoint/restart is specified, the Blockset techniques are bypassed by the sort/merge program.

Default: None; optional.

EQUALS|NOEQUALS

The program has a facility whereby the order of identically collating records can be preserved from input to output. Whether or not this facility is available by default depends on the specification made when the program was installed.

You can override the default setting by use of this parameter.

EQUALS
: means the order must be preserved.

Notes:

1. When the EQUALS option is used, 4 bytes containing a sequence counter are added internally to the beginning of each record. (For variable-length records the 4 bytes are located between the RDW (Record Descriptor Word) and the record itself.) Because of these, SM1 internally updates the starting point of each control field by 4 bytes. Do not specify EQUALS when variable-length records are sorted and the RDW is part of the control field, and a tape technique or a nonstandard disk technique is used.
2. The total number of bytes occupied by all control fields must not exceed 4088 when the EQUALS option is in operation.
3. Use of EQUALS can slow down the sort.

NOEQUALS
: means the order need not be preserved.

Default: Can be overridden by specification of EQUALS or NOEQUALS on the OPTION statement, or defaults to the option specified at installation time.

DYNALLOC (MVS ONLY)

The user can assign the task of dynamically allocating needed work space to sort/merge. This will relieve the user from the necessity of calculating and specifying, through JCL, the amount of intermediate work space needed by the program. The program will, by use of the dynamic allocation facility of the MVS operating system, allocate work space to get the best possible performance for the current application.

DYNALLOC=d|
DYNALLOC=(d[,n])
: d can be any of the following devices: 2314, 3330, 3330-1, 3340, 3350, 3375, 3380, 2400, 2400-3, 2400-4, 3400-3, 3400-4, 3850, or their user-assigned group name, such as SYSDA. n is the number of requested work data sets.

For disk work data sets, the total size is calculated using the information in the FILSZ keyword or, if the FILSZ keyword is omitted, the sort default value for dynamic allocation, 6000 blocks, is used. The block size in either case is the internal record length or 1000 bytes, whichever is the larger. One fifth of each work data set's primary space is specified as secondary allocation for that work data set. The size of each work data set is the total work area divided by n.

Dynamically allocated work data sets will not be unallocated until the job step is finished because SMF does not log the use of data sets that are dynamically unallocated. This means that recursive sorts reuse the work space allocated to the first sort. To prevent lack of space, give the first sort work space enough to satisfy the sort with the highest space requirement.

For tape work data sets, the number of volumes specified (explicitly or by default) will be allocated to the program. The program will request standard label tapes.

If DYNALLOC is specified under any system other than MVS, it is ignored. It is also ignored if SORTWK DD statements are provided.

With NOVIO: If your sort/merge program was installed with the NOVIO option ("no virtual I/O"):

- Work space will be allocated on nontemporary data sets (DSNAME parameter specified).
- The device (d) that you specify cannot be a virtual device unless a corresponding real disk is available in your system.

Default: If DYNALLOC is specified without the n parameter, n defaults to 3 (n defaults to 3 even if 0 is specified). The first parameter, d, must be specified. Can be overridden by DYNALLOC specified on the OPTION statement.

SORT STATEMENT EXAMPLES

```
SORT   FIELDS=(2,5,CH,A),FILSZ=29483
```

SORT Example 1. One Control Field and File Size Option

FIELDS
The control field begins on the second byte of each record in the input data set, is five bytes long, contains character data, and is to be sorted into ascending sequence.

FILSZ
The data set to be sorted contains exactly 29,483 records.

```
SORT   FIELDS=(7,3,CH,D,1,5,FI,A,398.4,7.6,BI,D,99.0,230.2,
          BI,A,452,8,FL,A),FILSZ=10693,CKPT,DYNALLOC=(3330,4)
```

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

FIELDS
The first four values describe the major control field. It begins on byte 7 of each record, is 3 bytes long, contains character (EBCDIC) 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 the fifth bit (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 the program examines it.

FILSZ
The data set to be sorted contains exactly 10,693 records.

CKPT
Instructs the program to take checkpoints during this run.

Note: When CKPT is specified, Blockset is bypassed by the sort/merge program.

DYNALLOC (MVS only)
Four work data sets will be allocated on 3330. The space on each data set will be calculated using the FILSZ value.

```
SORT   FIELDS=(3,8,ZD,E,40,6,CH,D),FILSZ=E30000
```

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

FIELDS
The first four values describe the major control field. It begins on byte 3 of each record, is 8 bytes long, and 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 (EBCDIC) data, and will be sorted into descending sequence.

FILSZ
The input data set contains approximately 30,000 records.

```
SORT   FIELDS=(25,4,A,48,8,A),FORMAT=ZD,EQUALS
```

SORT Example 4. Two Control Fields, Format and Equals Options

FIELDS
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.

FORMAT
The FORMAT=f 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:

```
SORT   FIELDS=(25,4,ZD,A,48,8,ZD,A),EQUALS
```

EQUALS
specifies that the order of equally collating records is to be preserved from input to output.

MERGE CONTROL STATEMENT

```
MERGE  {FIELDS=(p,m,f,s...,p,m,f,s)|
       FIELDS=(p,m,s...,p,m,s),FORMAT=f}
       [,FILSZ=x|,SIZE=y]
       [,CKPT]
```

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. Like SORT, MERGE parameters can be overridden by similar parameters specified on the OPTION control statement. The format, defaults, and specifications for the MERGE statement are similar to the SORT statement with the following differences:

- The operation definer is MERGE instead of SORT.
- The SKIPREC option is not used (ignored if specified).
- The EQUALS|NOEQUALS option is not used (ignored if specified).
- The DYNALLOC option is not used (ignored if specified).
- The value of the FILSZ operand is the total number of records in all the input data sets.

See Figure 7 for a description of the format of the MERGE control statement and a summary of the parameters it can contain.

FIELDS

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 are described in the discussion of the SORT statement. The defaults for this and the following parameters are also given there. See also Figure 7.

FORMAT

The FORMAT operand is used in the same way for a merge as for a sort.

FILSZ|SIZE

The FILSZ or 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. Either FILSZ or SIZE is acceptable. See the SORT control statement (FILSZ|SIZE).

CKPT

The CKPT (or CHKPT) operand is also optional. It causes the program to use the checkpoint facility of the operating system. The program takes checkpoints at end of volume on the output file, unless you supply the address of your own exit list for the SORTOUT DCB at exit E39. If this parameter is supplied, or if the output file takes up less than one volume, no checkpoints are taken.

When you use the checkpoint/restart facility, you must write a JCL statement to define a data set for the checkpoint records. How to write this JCL statement (//SORTCKPT) is described in Section 5.

MERGE STATEMENT EXAMPLES

```
 MERGE   FIELDS=(2,5,CH,A),FILSZ=29483
```

MERGE Example 1. One Control Field, Size Option

FIELDS

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

FILSZ

The input data sets contain exactly 29,483 records.

```
 MERGE   FIELDS=(3,8,ZD,E,40,6,CH,D),FILSZ=E30000
```

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

FIELDS

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.

FILSZ

The input data sets contain approximately 30,000 records.

```
 MERGE   FIELDS=(25,4,A,48,8,A),FORMAT=ZD,CKPT
```

MERGE Example 3. Two Control Fields, Format Option

FIELDS

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 parameter can be used because both control fields have the same data format.

CKPT

Instructs the program to take checkpoints during this run.

OPTION CONTROL STATEMENT

```
OPTION  [,FILSZ=x|SIZE=y]
        [,SKIPREC=z]
        [,CKPT]
        [,DYNALLOC=d|
        ,DYNALLOC=(d[,n])]
        [,CHALT|,NOCHALT]
        [,VERIFY|,NOVERIFY]
        [,CHECK|,NOCHECK]
        [,BLKSET|,NOBLKSET]
```

The OPTION control statement allows you to specify or override some of the options available with the SORT or MERGE control statements (such as FILSZ|SIZE, SKIPREC, CKPT, EQUALS|NOEQUALS, and DYNALLOC).

The OPTION control statement also allows you to override some of the options available at installation time (such as EQUALS, VERIFY, CHALT, CHECK, BLKSET, and VBLKSET).

If a parameter is not specified on the OPTION control statement, the resulting parameter is determined by specifications made on the SORT or MERGE control statement, or those made at installation time. OPTION parameters used by other IBM sort/merge programs will cause sort/merge to terminate unless they conform to the following parameters. See also Figure 7 for a description of the OPTION control statement and its parameters.

The OPTION control statement can be included in the SYSIN data set or it can be included in the SORTCNTL data set when sort/merge is dynamically invoked by another program. If the latter is done, the invoking program does not have to be recompiled. See Section 5 for information on how to specify a SORTCNTL DD statement in the JCL of the job that dynamically invokes sort/merge.

FILSZ|SIZE

It is recommended that this parameter always be specified. It is especially important if DYNALLOC is to be used.

FILSZ=x
: x is the exact number of records to be sorted; it must take into account records to be inserted or deleted at exit E15, or skipped by SKIPREC.

SIZE=y
: y is the exact number of records to be used as input, excluding any changes to be made at exit E15, or by SKIPREC (that is, the number of records in the SORTIN data set).

If the actual number of records is not the same as the value specified, the program will terminate with the value x or y placed in the IN field of the message ICE047A or ICE054I. This applies to both FILSZ and SIZE.

FILSZ|SIZE=En
: n is the estimated number of records to be sorted and it must be immediately preceded by the letter E; it should in either case be large enough to include both the SORTIN data set and any records you may add at exit E15.

For example, if you estimate your total data set size to be 5000 records, specify FILSZ=E5000. The program will accept either FILSZ or SIZE, but FILSZ is always preferable when its use is possible, as it allows better optimization for tape techniques and for disk techniques, when variable-length records are used. It should also be specified when DYNALLOC under MVS is requested.

If you omit the FILSZ or SIZE operand, the 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 (only for nonstandard disk techniques).
- If input is a VSAM data set (or sets), data set size is equal to that given in the VSAM catalog. Always specify FILSZ, therefore, if you want to add or delete records at E15.

Default: None; optional.

SKIPREC

SKIPREC=z
z is the number of records you want to skip before starting to process the input data set, and will usually be used if, on a preceding sort run, you have sorted only part of the input data set.

A program with an input data set which exceeds intermediate storage capacity will normally terminate unsuccessfully. However, for a tape or nonstandard disk sort, you can use a routine at E16 (as described in Section 6) to instruct the program to sort only those records already read in. It will then print a message giving the number of records sorted. You can use SKIPREC in a subsequent sort run to sort the remaining records, and then merge the output from different runs to complete the application.

Note: If SKIPREC is specified, the Blockset techniques are bypassed by the sort/merge program.

Default: None; optional.

CKPT

CKPT (the spelling CHKPT is also accepted) causes the program to activate the checkpoint/restart facility of the operating system. No checkpoints can be taken:

- If an invoked merge is handling output through exit E35
- If output from a merge-only operation is to be a VSAM data set
- In any user routine at a program exit

If this parameter is specified, the program takes the following checkpoints:

1. Start of sort phase (all tape techniques)
2. Start of each intermediate merge phase pass (balanced and polyphase tape technique); or at intervals during the intermediate merge phase (oscillating tape and all disk techniques)

3. Start of final merge phase

When you use the checkpoint/restart facility you must write a JCL statement to define a data set for the checkpoint records. How to write this JCL statement (//SORTCKPT) is described in Section 5.

Note: If checkpoint/restart is specified, the Blockset techniques are bypassed by the sort/merge program.

Default: None; optional.

EQUALS|NOEQUALS

The program has a facility whereby the order of identically collating records can be preserved from input to output. Whether or not this facility is available by default depends on the specification made when the program was installed. You can override the default setting by use of this parameter.

EQUALS
 means the order must be preserved.

Notes:

1. When the EQUALS option is used, 4 bytes containing a sequence counter are added internally to the beginning of each record. (For variable-length records, the 4 bytes are located between the RDW (Record Descriptor Word) and the record itself.) Because of these, SM1 internally updates the starting point of each control field by 4 bytes. Do not specify EQUALS when variable-length records are sorted and the RDW is part of the control field, and a tape technique or nonstandard disk technique is used.

2. The total number of bytes occupied by all control fields must not exceed 4088 when the EQUALS option is in operation.

3. Use of EQUALS can degrade performance.

NOEQUALS
 means the order need not be preserved.

Default: If this parameter is not specified, sort/merge defaults to the specification made on the SORT control statement or at installation time.

DYNALLOC (MVS ONLY)

The user can assign the task of dynamically allocating needed work space to sort/merge. This will relieve the user from the necessity of calculating and specifying, through JCL, the amount of intermediate work space needed by the program. The program will, by use of the dynamic allocation facility of the MVS operating system, allocate work space to get the best possible performance for the current application.

DYNALLOC=d|
DYNALLOC=(d[,n])
 d can be any of the following devices: 2314, 3330, 3330-1, 3340, 3350, 3375, 3380, 2400, 2400-3, 2400-4, 3400-3, 3400-4, 3850, or their user-assigned group name, such as SYSDA. n is the number of requested work data sets.

For disk work data sets, the total size is calculated using the information in the FILSZ keyword or, if the FILSZ keyword is omitted, the sort default value for dynamic allocation, which is 6000 blocks, is used. The block size in either case is the internal record length or 1000 bytes, whichever is the larger. One fifth of each work data set's primary space is specified as secondary allocation for that work data set. The size of each work data set is the total work area divided by n.

Dynamically allocated work data sets will not be unallocated until the job is finished. This is because SMF does not log the use of data sets that are dynamically unallocated. This means that recursive sorts reuse the work space allocated to the first sort. To prevent lack of space give the first sort work space enough to satisfy the sort which has the highest space requirement.

For tape work data sets, the number of volumes specified (explicitly or by default) will be allocated to the program. The program will request standard label tapes.

If DYNALLOC is specified under any system other than MVS, it is ignored. It is also ignored if SORTWK DD statements are provided.

With NOVIO: If your sort/merge program was installed with the NOVIO option ("no virtual I/O"):

- Work space will be allocated on nontemporary data sets (DSNAME parameter specified).

- The device (d) that you specify cannot be a virtual device unless a corresponding real disk is available in your system.

Default: If DYNALLOC is specified without the n parameter, n defaults to 3 (n defaults to 3 even if 0 is specified). The first parameter, d, must be specified. If this parameter is not specified, sort/merge defaults to the specification made on the SORT control statement.

CHALT|NOCHALT

You can specify that you want format CH fields translated by the ALTSEQ table as well as format AQ, or just the latter. Whether or not this facility is available by default depends on the specification made when the program was installed. You can override the default setting by use of this parameter.

CHALT
    means that sort/merge will translate character control fields with formats CH and AQ using the ALTSEQ table.

NOCHALT
    means that format CH fields will not be translated.

Default: If this parameter is not specified, sort/merge defaults to the specification made at installation time.

VERIFY|NOVERIFY

This parameter enables sort/merge to perform sequence checking on the final output record sequence. Whether or not this facility is available by default depends on the specification made when the program was installed. You can override the default setting by use of this parameter.

VERIFY
    means that sequence checking will be performed.

NOVERIFY
    means that sequence checking will not be performed.

Note: Use of VERIFY can degrade performance.

Default: If this parameter is not specified, sort/merge defaults to the specification made at installation time.

CHECK|NOCHECK

This parameter enables sort/merge to check the record counters at the end of program execution. Whether or not this facility is available by default depends on the specification made when the program was installed. You can override the default setting by use of this parameter.

CHECK
    means that record counter checking will be done at the end of program execution.

NOCHECK
    means that record counter checking will not be done.

Default: If this parameter is not specified, sort/merge defaults to the specification made at installation time.

BLKSET|NOBLKSET

This parameter allows sort/merge to attempt to execute one of the Blockset techniques. Whether or not this facility is available by default depends on the specification made when the program was installed. You can override the default setting by use of this parameter.

BLKSET
    means that sort/merge will try to execute one of the Blockset techniques. However, certain conditions must be met before sort/merge will select Blockset (see "Conditions for Use of Blockset Sorting Techniques" in Section 8).

NOBLKSET
    means that sort/merge will bypass the Blockset techniques.

Default: If this parameter is not specified, sort/merge defaults to the specification made at installation time (BLKSET for fixed-length records and VBLKSET for variable-length records).

OPTION STATEMENT EXAMPLES

```
SORT   FIELDS=(1,20,CH,A)
OPTION SIZE=50000,SKIPREC=5,CKPT,EQUALS,DYNALLOC=3350
```

OPTION Statement Example 1. One Control Field and Related Options

FIELDS
    The control field begins on the first byte of each record in the input data set, is 20 bytes long, contains character data, and is to be sorted into ascending order.

SIZE
    The data set to be sorted contains 50,000 records.

SKIPREC
    Five records will be skipped before starting to process the input data set.

CKPT
    Sort/merge takes checkpoints during this run.

Notes:

1. When CKPT or SKIPREC is specified, Blockset will be bypassed by the sort/merge program.

2. If nonconflicting parameters, such as CKPT, happen to be coded on both the SORT and OPTION control statements, it's a "don't care" situation, with no advantage gained from doing so.

EQUALS
    The order of equally collating records is preserved from input to output.

DYNALLOC=3350
    Three data sets (by default) are allocated on 3350 (MVS only). The space on each data set is calculated using the SIZE value.

The parameters coded on the OPTION control statement can still be specified on the SORT or MERGE control statement, as they were under Release 4.

```
SORT   FIELDS=(1,2,CH,A),CKPT
OPTION EQUALS,NOCHALT,NOVERIFY,CHECK
```

OPTION Example 2. Illustrating the Relationships Between
the OPTION and SORT Control Statements
and the ICEMAC Installation Option

FIELDS
    The control field begins on the first byte of each record in the input data set, is 2 bytes long, contains character data, and is to be sorted into ascending order.

CKPT
    Sort/merge takes checkpoints during this run (see also Notes under CKPT for Example 1).

EQUALS
    The order of equally collating records is preserved from input to output.

NOCHALT
    Only AQ fields will be translated through the ALTSEQ translate table. (This would override CHALT=YES had that been specified at installation time.)

NOVERIFY
    No sequence check is performed on the final output records.

CHECK
    Record counters are checked at the end of program execution.

```
OPTION FILSZ=50,SKIPREC=5,DYNALLOC=3330
SORT  FIELDS=(1,2,CH,A),SKIPREC=1,SIZE=200,DYNALLOC=(3350,5)
```

OPTION Example 3. Using OPTION to Override SORT

This example shows how parameters specified on the OPTION control statement take precedence over those specified on the SORT control statement, regardless of the order of the 2 statements.

FILSZ
    Sort/merge expects 50 records on the input data set. (Note that there is a difference in meaning between FILSZ and SIZE, and that the OPTION specification of FILSZ will be used in place of SIZE.)

SKIPREC

Sort/merge causes five records from the beginning of the input file to be skipped. (SKIPREC=1 on the SORT statement is ignored.)

DYNALLOC

Sort/merge allocates three work data sets (by default) on a 3330 (MVS only).

FIELDS

The control field begins on the first byte of each record in the input data set, is 2 bytes long, contains character data, and is to be sorted in ascending order.

```
OPTION NOBLKSET
```

OPTION Example 4. Bypassing Blockset Techniques

NOBLKSET

Sort/merge bypasses FLR-Blockset or VLR-Blockset regardless of whether the Blockset techniques were specified at installation time. Sort/merge uses Peerage, Vale, or some other conventional sorting technique instead.

```
OPTION BLKSET
```

OPTION Example 5. Using OPTION to Override Specification Made at Installation Time

BLKSET

Even if 'BLKSET=NO' (for fixed-length records) or 'VBLKSET=NO' (for variable-length records) were specified at installation time, 'OPTION BLKSET' would override both and cause sort/merge to try to execute using one of the Blockset techniques before any other technique.

RECORD CONTROL STATEMENT

```
RECORD  TYPE=x,[LENGTH=(L1,L2,L3,L4,L5)]
```

The RECORD control statement describes the format and lengths of the records being sorted or merged. It is required when you change record lengths during a sort/merge program run; for a sort invoked from a program written in assembler or PL/I; and when input is from a VSAM data set. However, to optimize performance when sorting variable-length records, you can use the RECORD statement to supply the minimum and average record lengths to the program.

See also Figure 7 for a description of the RECORD control statement and its parameters.

TYPE

TYPE=F
indicates that the records to be sorted or merged are fixed-length records.

TYPE=V
indicates that the records are EBCDIC variable-length.

TYPE=D
indicates that the records are ASCII variable-length.

For QSAM records, the 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 in Section 5), or that given on the data set label. If the formats are not the same or TYPE is not specified, the program uses the one given in the data set label/DD statement.

Default: Required for E15 input if no SORTIN RECFM; otherwise, defaults to SORTIN RECFM.

LENGTH

This parameter is required when you change record lengths at one or more exits, or when no SORTIN DD statement is supplied. You can aid optimization by always supplying it when sorting variable-length records.

Details of how to write the parameter are given in Figure 7.

Input record length, L1, is required and only used when no SORTIN DD statement is supplied. L1 must be at least as large as the maximum input record size; if it is larger than needed, performance can suffer.

It is extremely important to specify an accurate value for L2 if you change record lengths at E15. Note that if you have specified a value for L1 but not for L2, the value you specified will act as default for L2 even if the L1 value has subsequently been overridden.

If work units are tape, the minimum length for records to be sorted (L2) is 18 bytes.

Output record length, L3, can usually be supplied by default: only if no LRECL (or maximum RECSZ, for VSAM) is available for SORTOUT, either in the DD statement or in the label, and the L1 value is unsuitable, do you need to specify L3.

Specifying the minimum record length (L4) helps performance. However, if you specify too large a value, the program will fail and will issue message ICE015A. The default for L4 is the minimum length needed to contain all control fields; if this length is less than 18 bytes, then 18 bytes is used instead—unless the records are shorter than 18 bytes, in which case record length is used.

L5 is the average record length for variable-length records. If the average record length is more than 350 bytes, you should specify L5. This will enable sort/merge to select the best technique, whether Vale or VLR-Blockset, to handle sorting. If you don't specify L5, sort/merge will try to execute using VLR-Blockset.

Default: For defaults, see RECORD in Figure 7.

Omitting Values

Normal syntax rules apply:

- You can drop values from the right, that is, LENGTH=(80,70,70,70).
- You can omit values from the middle or left as long as you indicate their omission by a comma, that is, LENGTH=(,,,30,80).
- At least one value must be given.

RECORD STATEMENT EXAMPLES

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

RECORD Example 1. Fixed-Length, Three Length Values

TYPE
The input records are fixed-length.

LENGTH
The records in the input data set are each 60 bytes long. Exit E15 is used to change the records to 40 bytes in the sort phase and exit E35 is used to change the records to 50 bytes in the final merge phase.

```
RECORD   TYPE=V,LENGTH=(200,175,180,50,80)
```

RECORD Example 2. Variable-Length, Five Length Values

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

LENGTH
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 handled by the sort phase is 50 bytes and the average record length is 80 bytes.

```
RECORD   TYPE=V,LENGTH=(200,,,,80)
```

RECORD Example 3. Variable Length, Two Length Values

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

LENGTH
The maximum length of the records in the input data set is 200 bytes. You do not change record lengths, so you omit L2 and L3; L4 is also omitted. The average record length is 80 bytes.

MODS CONTROL STATEMENT

```
MODS   exit=(n,m,s[,e])...,exit=(n,m,s[,e])
```

The MODS statement is needed only if you want the program to pass control to your routines at program exits. The MODS statement associates the user routine(s) with specific exits in the program and provides the program with descriptions of these routines. For details about exits from the program and how user routines can be used, see Section 6.

See also Figure 7 for a general description of the format and specifications of the MODS control statement and its parameters.

The program has exits from which control can be transferred to your own routines. These exits have three-character names, in the form Exy where x is the number of the program phase in which the exit occurs, and y is the number of the exit within that phase. (For example, E31 is the first exit in Phase 3.)

To use one of the exits, you substitute its three-character name for the word exit in the MODS statement format example (Figure 7). The values that follow 'exit' describe the user routine. These values are:

n

the name of your routine (member name if your routine is in a library). You may use any valid operating system name for your routine. This allows you to keep several alternative routines with different names in the same library.

m

the number of bytes of main storage that your routine uses. Include storage obtained (GETMAIN) by your routine, or on its behalf, for example by OPEN.

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.

e

indicates the linkage editor requirements of your routine. It must have one of the values T, S, or N.

T

means that your routine must be link-edited together with other routines to be used in the same phase of the program.

S

means that your routine requires link-editing but that it can be link-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 link-editing.

N

means that your routine has already been link-edited and can be used in the sort/merge run without further link-editing. All routines for which you specify N must be in the same library, or in libraries defined as a concatenated data set.

If no parameter is specified, T is assumed.

Refer to "Spare the Linkage Editor" in Section 8 for details on how to design your routines.

When you are preparing your MODS statement, bear in mind that:

- 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 (including requirements for system services), 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: you do not precede the value by E for an estimate.

- If the routines you are using for a particular sort/merge run are in several libraries, you need a DD statement for each library. DD statements required for the program are described in Section 5.

- If your routines are in the system input stream (SYSIN), you must arrange them in numeric order (the E11 routine before the E15 routine, etc.). You must supply a SORTMODS DD statement, as described in Section 5. If you use the same routine in several sort/merge program phases, you must provide a separate copy of the routine for each exit.

Default: All parameters must be specified except for e. If e is not specified, the default is T.

MODS STATEMENT EXAMPLES

```
MODS    E15=(ADDREC,552,MODLIB,N),E35=(ALTREC,11032,MODLIB,N)
```

MODS Example 1. Two Routines in a Library, No Link-Editing

E15 At exit E15, the program will transfer control to your own routine. Your routine is in the library defined by a job control statement with the ddname MODLIB. Its member name is ADDREC; it is 552 bytes long has been link-edited previously, and does not require further link-editing.

E35 At exit E35, the program will transfer control to your routine. Your routine is in the library defined by the job control statement with the ddname MODLIB. Its member name is ALTREC; it is 11032 bytes long and has been link-edited previously.

```
MODS    E17=(CLSE,344,SYSIN)
```

MODS Example 2. One Routine in SYSIN, Link-Editing is Needed

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

```
MODS    E16=(NMAXERR,1000,MYLIB),E21=(E21OWN,552,MODLIB),
        E31=(E31,456,SYSIN),E35=(SUMUP,5000,SYSIN)
```

MODS Example 3. Four Routines

E16 The program will transfer control 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. It needs link-editing (together with other routines for the same phase).

E21 At exit E21, the program will transfer control to your routine which resides under the member name E21OWN in the library defined by the job control statement with the ddname MODLIB. Your routine is 552 bytes long and requires link-editing.

E31 Another of your routines is in SYSIN, and will gain control at exit E31. It is 456 bytes long and must be link-edited together with other routines in the same phase (the default linkage editor specification).

E35 You have also placed a routine named SUMUP as an object deck in the input stream. It is approximately 5000 bytes long, must be link-edited together with other routines in its phase (that is, the E31 routine), and will receive control at exit E35.

```
MODS    E11=(E11,504,MYLIB,S)
```

MODS Example 4. One Routine, Separate Link-Editing

E11 At exit E11 in the sort phase, the program will transfer control to your routine E11. It is located in a library defined by a job control statement with the DDname MYLIB, is 504 bytes long, and can be link-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 link-editing, your routine can have no external references.

ALTSEQ CONTROL STATEMENT

```
ALTSEQ    CODE=(fftt...,fftt)
```

The ALTSEQ statement is used if you wish to change the collating sequence of EBCDIC character data. If a modified version of the collating sequence is available by default at your installation, the ALTSEQ statement will override it.

When you supply an ALTSEQ statement, the modified collating sequence will be used for any control field whose format you specify on the SORT statement as AQ. If you specify AQ without supplying an ALTSEQ statement, the program will use the default available at your installation, if there is one. Otherwise, it will use the standard EBCDIC collating sequence.

CODE

The modifications are described in the form CODE=(fftt,fftt...), where:

ff
: represents in hexadecimal the character whose position is to be changed, in the EBCDIC collating sequence.

tt
: is the EBCDIC hexadecimal representation of the position to which the character is to be moved.

The order in which the parameters are specified is immaterial.

Note: If CHALT is specified on the OPTION control statement or CHALT=YES is specified at installation time, control characters with format CH will be translated by the ALTSEQ table in addition to those with format AQ.

Default: If this parameter is not specified, sort/merge defaults to the specification made at installation time.

ALTSEQ STATEMENT EXAMPLES

```
ALTSEQ   CODE=5BEA
```

ALTSEQ Example 1

The character represented by X'5B'($ or national character) is to collate after 'Z' (at position X'EA').

```
ALTSEQ   CODE=(F0B0,F1B1,F2B2,F3B3,F4B4,F5B5,F6B6,
         F7B7,F8B8,F9B9)
```

ALTSEQ Example 2

The numerals 0-9 are to collate before uppercase letters (but after lowercase letters).

```
ALTSEQ   CODE=(C180,C282,8283,C384,8385,C486,8487,C588,8589,
         C68A,868B,C78C,878D,C88E,888F,C990,8991,D192,9193,
         D294,9295,D396,9397,D498,9499,D59A,959B,D69C,969D,
         D79E,979F,D8A0,98A1,D9A2,99A3,E2A4,A2A5,E3A6,A3A7,
         E4A8,A4A9,E5AA,A5AB,E6AC,A6AD,E7AE,A7AF,E8B0,A8B1,
         E9B2,A9B3)
```

ALTSEQ Example 3

Uppercase A is to collate before lowercase a, B before b, and so on through to Zz. The parameters may be specified in any order.

DEBUG CONTROL STATEMENT (STANDARD DISK TECHNIQUES ONLY)

```
DEBUG   ABEND|NOABEND
```

The DEBUG control statement cannot be used if work data sets are on tape; if specified, it is ignored.

In normal use, only the ABEND and NOABEND parameters will be of interest. They override the default error return settings (ERETINV or ERETJCL options) made when the program was installed.

The DEBUG control statement can also be used to force a nonstandard disk sorting technique if a problem has occurred and a bypass is wanted. Other parameters and details of dumps obtained are described in Appendix A.

See also Figure 7 for a general description of the format and specifications of the DEBUG control statement.

ABEND

If you specify this parameter and your sort or merge is unsuccessful, it will ABEND with a user completion code equal to the appropriate message number. It will also cause an ABEND if the unsuccessful sort or merge was invoked from another program.

NOABEND

An unsuccessful sort or merge will terminate with a return code of 16.

Default: This parameter is used only for standard disk sorts. It overrides the ERETJCL and ERETINV options specified at program installation time.

DUMP|NODUMP

These options are recognized but ignored.

END CONTROL STATEMENT

```
END
```

The END statement marks the end of all program control 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 /* or a job control statement. For example, if you include your own routines in the input stream, they are placed between the program control statements and the next job control statement, so you must use an END statement.

If the END statement is used in the SORTCNTL data set and a listing of control statements is requested, END will not appear.

The format of the END statement is also shown in Figure 7. The statement has no operands.

SECTION 5. JOB CONTROL STATEMENTS

This section describes the job control language (JCL) statements you must write for the program. You must include JCL statements with each program application you submit for execution, to describe your application to the operating system.

The job control statements required for a program application include a JOB statement, an EXEC statement, and several DD statements; these statements, their functions, and the order in which they are arranged in the system input stream are shown in Figure 10.

The inclusion of certain JCL statements depends on whether you initiate the program with an EXEC statement in the input job stream, or with a system macro instruction within your own program. The JCL statements you include can also depend on whether or not you wish to use program exits for routines of your own. These differences are noted in Figure 10. If you intend to use system macro instructions or program exits, or both, you should be familiar with the material in Sections 6 and 7 of this publication.

While reading this section, you may need OS/VS1 JCL Reference or OS/VS2 JCL Reference for supplementary information; you should have it available for ready reference.

JOB STATEMENT

The JOB statement is the first JCL statement of your job. It must contain a valid jobname in its name field and the word JOB in its operation field. All parameters in its operand field are optional, although your installation may make such information as the account number and the programmer's name mandatory.

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

EXEC STATEMENT

The EXEC statement is the first JCL statement of each step in your job. It is also the first statement of each procedure step in a cataloged procedure. It identifies to the operating system the sort/merge program or the sort cataloged procedure that is to be used. The EXEC statement is followed in the input stream by DD statements.

This subsection describes the required and optional parameters of the EXEC statement. These parameters include either the program name or the name of a cataloged procedure, followed by optional parameters. To initiate sort execution with a system macro instruction within your own program, see Section 7.

A cataloged procedure is a set of JCL statements, including DD statements, that has been assigned a name and placed in a partitioned data set known as the procedure library. Two cataloged procedures are supplied with the program: SORT and SORTD. They are specified in the first parameter of the EXEC statement by PROC=SORT,PROC=SORTD, or simply SORT or SORTD.

```
//jobname     JOB           Always needed

Preceding job steps, if any

//stepname    EXEC          Always needed.

                            The following DD statements can be in any
                            order:

//STEPLIB     DD            Omit when using a cataloged procedure.
//SORTLIB     DD            SORTLIB only needed for tape sorts or
                            any merge-only application, or if any of
                            the old disk sort techniques are forced.
//SYSOUT      DD
//SYSLIN      DD[2]
//SYSLMOD     DD[2]
//SYSUT1      DD[2]
//SYSPRINT    DD[2]

//DDname      DD            Library definition if you use routines
                            from a library.

//SORTIN      DD            Usually needed. For a merge-only, the
                            SORTINnn cards should come here in
                            consecutive order.

//SORTOUT     DD            Usually needed.

//SORTWKnn    DD            Not needed for a merge-only or for sorts
                            in main storage. Must not be included if
                            you want dynamic allocation (MVS only).
                            (The DDname SORTDKnn is used by the program
                            instead of SORTWKnn if it carries out
                            dynamic reallocation.)

//SORTMODS    DD            Only needed if you have routines in SYSIN.

//SORTCKPT    DD            Only needed if checkpoints are to be taken.

//SYSUDUMP    DD            (or SYSABEND or SYSMDUMP) Not always needed.

//SORTCNTL    DD[3]         Include if you want to define a data set
                            from which additional or changed sort
                            control statements can be read, when the
                            sort is invoked from another program.


//SYSIN       DD   *
    SORT statement[1]       (or MERGE statement) Always needed.
    OPTION statement[1],[3]
    RECORD statement[1],[3]
    MODS statement[1],[3]

    ALTSEQ statement[1],[3] Used to modify the EBCDIC collating
                            sequence (see Section 4).

    DEBUG statement[1],[3]  Mainly for debugging (see Appendix A).

    END statement[3]        Must be last statement.

  Object decks for your own routines (if any).
  /*

[1]Can be in any order.
[2]Include if you have routines of your own to be link-edited, and are
 not using the cataloged procedure (SORT).
[3]Not always needed (see Section 4).
```

Figure 10. Input Job Stream

The format of the EXEC statement is:

```
//stepname EXEC {[PGM=SORT|ICEMAN]|[PROC=SORT|SORTD]|
                 [SORT|SORTD]}[,PARM='options[1]']
                 [,any other parameters]
```

[1] See "'PARM' Field Options" below.

If you use the PROC= notation it has the same effect as simply using the name of the procedure, but serves as a reminder that a cataloged procedure is being used.

If you are not using a cataloged procedure, you should use PGM= either with the actual name of the sort module (ICEMAN) or with its alias, SORT. Check that the alias has not been changed at your particular installation.

'SORT' CATALOGED PROCEDURE

Use the SORT cataloged procedure when you include user routines that require link-editing. Because this procedure allocates linkage editor data sets, whether or not you include user routines, it is inefficient if you do not include such routines.

When you specify EXEC PROC=SORT or EXEC SORT, the following JCL statements are generated:

```
//SORT    EXEC  PGM=ICEMAN                                       00
//STEPLIB  DD   DSNAME=yyy,DISP=SHR                              10
//SORTLIB  DD   DSNAME=xxx,DISP=SHR                              20
//SYSOUT   DD   SYSOUT=A                                         30
//SYSPRINT DD   DUMMY                                            40
//SYSLMOD  DD   DSNAME=&GOSET,UNIT=SYSDA,SPACE=(3600,(20,20,1))  50
//SYSLIN   DD   DSNAME=&LOADSET,UNIT=SYSDA,SPACE=(80,(10,10))    60
//SYSUT1   DD   DSNAME=&SYSUT1,SPACE=(1024,(60,20)),             70
//              UNIT=(SYSDA,SEP=(SORTLIB,SYSLMOD,SYSLIN))        80
```

00 The stepname of the procedure is SORT. This EXEC statement initiates the program, which is named ICEMAN. A region parameter will probably have been added when the program was installed.

10 The STEPLIB DD statement defines the data set containing the sort/merge program modules that reside in a link library. The data set is cataloged, and the data set name represented by yyy is specified at generation time; it can be SYS1.LINKLIB.

20 The SORTLIB DD statement defines the data set that contains the sort/merge program modules. The data set is cataloged, and the data set name represented by xxx was specified at generation time; it can be SYS1.SORTLIB.

30 Defines an output data set for system use (messages). It is directed to system output class A.

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

50 Defines a data set for linkage editor output. Any system direct access device is acceptable for the output. Space for 20 records with 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.

60 The SYSLIN data set is used by the program for linkage editor control statements. 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.

70/80 The SYSUT1 DD statement defines a work data set for the linkage editor.

'SORTD' CATALOGED PROCEDURE

Use the SORTD cataloged procedure either (a) when you do not include user routines or (b) when you include user routines that do not require link-editing.

When you specify EXEC PROC=SORTD or EXEC SORTD, the following JCL statements are generated:

```
//SORT   EXEC  PGM=ICEMAN                                     00
//STEPLIB DD   DSNAME=yyy,DISP=SHR                            10
//SORTLIB DD   DSNAME=xxx,DISP=SHR                            20
//SYSOUT  DD   SYSOUT=A                                       30
```

00 The stepname of the SORTD procedure is SORT. A region parameter will probably have been added when the program was installed.

10 The STEPLIB DD statement defines the data set containing the sort/merge program modules that reside in a link library. The data set is cataloged, and the data set name represented by yyy is specified at generation time; it can be SYS1.LINKLIB.

20 Defines the data set containing sort/merge program modules. The data set name of the program subroutine library, represented by xxx, is specified at generation time; it can be SYS1.SORTLIB.

30 Directs messages to system output class A.

'PARM' FIELD OPTIONS

The options described below are keyword parameters, and can therefore be specified in any order.

```
  PARM='[BALN|OSCL|POLY] [,SIZE(value)|,SIZE(MAX)]
       [,FLAG(I)|,FLAG(U)|,NOFLAG] [,LIST|,NOLIST] [,DIAG]'
```

BALN|OSCL|POLY: When using tape work areas, you can force the program to use a specific sorting technique. The techniques available are:

- BALN—the balanced tape technique
- OSCL—the oscillating tape technique
- POLY—the polyphase tape technique

If you omit this option for a tape sort, the program tries to select the most efficient technique for your particular application. You should therefore be extremely cautious of forcing a specific technique, since this can result in reduced efficiency.

If you use disk work areas and specify a technique parameter in the PARM field (BALN, PEER, or CRCX), it will be recognized but ignored. You can then instead force a technique (for example, for bypassing purposes) using the DEBUG statement described in Appendix A.

For more information on choice of techniques, see Figure 6 in Section 3, "Summary of Intermediate Storage Requirements."

SIZE(VALUE)|SIZE(MAX): You can temporarily override the main storage allocated to sort/merge by specifying:

- SIZE(value), where value is a decimal value representing the number of bytes of main storage to be allocated. See Section 3 for a description of how to calculate the required amount.
- SIZE(MAX), which instructs the program to calculate the amount of main storage available and allocate this maximum amount, up to the MAXLIM value set when the program was installed. The program will allow space (within MAX) if needed for VSAM and its buffer pools.

  Do not use SIZE(MAX) with password-protected data sets if passwords are to be entered through a routine at an exit, since the program cannot then open the data sets in Phase 0 to make the necessary calculations.

If the value of SIZE is less than the MINLIM value set at installation time, the MINLIM value will be used.

The program also accepts the parameter CORE for this option. SIZE and CORE may not both be specified at the same time. For compatibility reasons, it will also accept the format SIZE=value|SIZE=MAX.

FLAG(I)|FLAG(U)|NOFLAG: You can temporarily override the message option specified at sort generation time, as follows:

- FLAG(I)—All messages, informational and critical, are written. Critical messages also appear on the operator console.
- FLAG(U)—Only critical (unrecoverable) messages are written. They also appear on the operator console.
- NOFLAG—No messages are printed; critical messages appear on the operator console.

For compatibility reasons, the form MSG=NO|CC|CP|AC|AP|PC is also accepted. The meanings are described in the OS Sort/Merge Programmer's Guide relating to Sort/Merge Program Product 5734-SM1.

LIST|NOLIST: You can temporarily override the list option specified at sort generation time.

- LIST means that all sort/merge control statements will be printed on SYSOUT, preceded by a heading.
- NOLIST specifies that neither heading nor control statements are to be printed.

DIAG: DIAG is intended as a diagnostic tool on nonstandard disk, tape, or merge applications at execution time. You should take care to specify it only when you actually need it, because it can impair program performance.

This option provides a listing of the program control statements, a module map, and a list of diagnostic messages containing addresses of areas critical for program execution. A complete list of the diagnostic messages is given in Appendix A.

If the program terminates unsuccessfully, which is indicated by a critical message, the DIAG option causes an 0C1 abend. If you include a SYSABEND, SYSMDUMP, or SYSUDUMP statement, you will also receive a dump of main storage. For information on abnormal termination dumps, refer to OS/VS1 Debugging Guide, or OS/VS2 Debugging Handbook.

In systems with multiple console support, diagnostic messages are printed on the system master console, unless they have been suppressed.

Diagnostic information for standard disk techniques can be obtained by using the DEBUG control statement, described in Appendix A.

DD STATEMENTS

A number of DD statements must be provided. Some are system DD statements, and will usually be supplied by the cataloged procedure, if you use one; others, you must always supply yourself if they are required. They are described below under "System DD Statements" and "Program DD Statements," respectively.

Required DD statement parameters are summarized in Figure 11, and DCB subparameters in Figure 12.

If you are running under MVS and are using conventional techniques (that is, those that have tape work storage, or are forcing a nonstandard disk technique), you are advised not to use FREE=CLOSE on your DD statements.

Shared Tape Units

A single tape unit may be assigned to two sort/merge data sets when the data sets are one of the following pairs:

- Unless OSCL is being used, the input data set and the first intermediate storage data set (SORTWK01)
- The input data set and the output data set

If you wish to associate the SORTIN data set with SORTWK01, you could include in the DD statement for SORTWK01 the parameter: UNIT=AFF=SORTIN. The AFF subparameter causes the system to place the data set on the unit occupied by the data set associated with the DDname following the subparameter (SORTIN, in this case).

In the same way, you could associate SORTIN with SORTOUT by including UNIT=AFF=SORTIN in the SORTOUT DD statement.

Parameter	Condition Under Which Required	Summary of Parameter Values	Default Value
DSNAME or DSN	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 or temporary name of the data set.	The system assigns a unique name.
DCB	Always required when 7-track tape is used; for input on tape without standard labels; and when the default values are not applicable.	Specifies information used to fill the data control block (DCB) associated with the data set.	(See separate subparameters in Figure 12)
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 data set on direct access.	Specifies the amount of space needed to contain the data set.	
VOLUME or VOL	When the input data set is neither cataloged nor passed, for multireel 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).
{AMP|BUFSP}	When password-protected VSAM data sets are used and the password is supplied through E18, E38 or E39.	Minimum buffer pool value given when creating the data set.	None.

Figure 11. DD Statement Parameters Used by Sort/Merge

Subparameter	Condition Under Which Required	Summary of Subparameter Values	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	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 and the default value is not suitable, except on SORTWK statements.	Specifies the format of the records in the data set.	• For OLD data sets, the value in the data set label. • For NEW SORT-OUT data sets, the same as for the first SORTIN or SORTINnn data set.[3] • No default if input on unlabeled tape, or BLP or NSL specified.
LRECL[1]		Specifies the maximum length (in bytes) of the logical records in the data set.	
BLKSIZE[2]		Specifies the maximum length (in bytes) of the physical records in the data set.	
OPTCD	When processing data in ASCII format.	Specifies that the tape processed is in ASCII format.	
BUFOFF	When processing data in ASCII format.	Specifies the length of the buffer offset or specifies that the buffer offset is the block length indicator.	

[1] With fixed-length records, LRECL can be used in the SORTOUT DD statement to shorten output records, if care is taken that the shortened records still include all of the control fields. With variable-length records, LRECL cannot be used in the SORTOUT DD statement to shorten output records.

[2] This is the only subparameter allowed for DD * data sets.

[3] If you are executing SM1 in several different steps within the same job you are advised not to rely on the defaults for SORTOUT but to give explicit values, as the system may not be able to keep track of the desired values.

Figure 12. DCB Subparameters Used by Sort/Merge

SYSTEM DD STATEMENTS

If you do not use a cataloged procedure to invoke the program, you may need to include system DD statements in the input stream. (See also the following section for DD statements dedicated to sort/merge, such as SORTLIB.) The DD statements contained in the cataloged procedure (or provided by you) are:

//JOBLIB DD or

//STEPLIB DD statement will be needed to identify your program link library if it is not already known to the system.

//SYSIN DD contains the sort/merge control statements when sort/merge is not invoked by another program. It can also contain user exit routines. The control data set normally resides in the input stream; however, it can be defined as a sequential data set or as a member of a partitioned data set. The data set must not be defined as RECFM=U.

//SYSOUT DD used as the system output data set for messages. Always use this statement if a cataloged procedure is not used. If you are invoking the program from another program, check whether a DDname other than SYSOUT was specified at generation time. Before printing sort messages, a skip to a new page is performed. (If you are invoking sort from a COBOL program and using no other DDname than SYSOUT, the use of EXHIBIT or DISPLAY in your COBOL program can give uncertain printing results.)

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

//SYSUT1 DD used as a work area by the linkage editor. Use this statement when user routines that need link-editing are included.

//SYSLIN DD defines a data set on which the sort program will place control information for the linkage editor. Use this statement when user routines that require link-editing are included.

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

//SYSUDUMP DD (or SYSABEND) defines output from a system ABEND dump routine. Needed if an unsuccessful run is to terminate with an ABEND dump (instead of a return code of 16).

PROGRAM DD STATEMENTS

In addition to the standard JCL statements required for normal program execution, the sort/merge program may use other dedicated JCL DD statements, as follows:

//SORTLIB DD defines the data set that contains load modules for the program. Only needed if a cataloged procedure is not used or if you are using any sort application with tape work areas or any merge application, or if any of the nonstandard disk techniques are forced.

//SORTIN DD defines the input data set for a sorting application.

//SORTINnn DD define the input data sets for a merging application.

//SORTWKnn DD define intermediate storage data sets. Usually needed for a sorting application unless dynamic allocation is requested.

//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 your exit 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 in 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.

//SORTCNTL DD defines the data set from which additional or changed sort control statements can be read, when the sort is invoked from another program.

//SORTDKnn DD is the DDname given to a VIO SORTWKnn allocation by sort/merge if it is dynamically reallocated (MVS only) and should never be specified in the job stream.

SORTLIB DD Statement

The SORTLIB DD statement defines the data set that contains sort/merge load modules to execute the conventional sorting and merging techniques. You need a SORTLIB if you are (1) using any sort application with tape work areas, (2) not using a cataloged procedure, (3) using any merge only application, or (4) forcing any of the nonstandard techniques.

```
//SORTLIB DD DSNAME=USORTLIB,DISP=(OLD,KEEP)
```

DD Example 1. SORTLIB DD Statement

This example shows DD statement parameters that define a previously cataloged input data set:

DSNAME
: causes the system to search the catalog for a data set with the name USORTLIB. When the data set is found, it is associated with the DDname SORTLIB. The control program obtains the unit assignment and volume serial number from the catalog and writes 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 kept after the current job step.

For information on the parameters used in the SORTLIB DD statement, the conditions under which they are required, and the default values assumed if a parameter is not included, see Figure 11 on page 63. The subparameters of the DCB parameter are described similarly in Figure 12. See your OS/VS1 JCL Reference or OS/VS2 JCL Reference for more detailed information.

SORTIN DD Statement

The SORTIN DD statement describes the characteristics of the data set in which the records to be sorted reside, and indicates its location.

If you provide the address of an E15 exit that supplies all input to sort/merge:

- No SORTIN statement is needed if you are invoking sort/merge from another program.
- A SORTIN DD DUMMY can be used if you are initiating sort/merge with an EXEC statement, but remember to give DCB parameters (see Figure 12); you can omit the SORTIN statement if you supply a LENGTH parameter on the RECORD control statement.

Sort/merge will accept an empty (null) QSAM data set for sorting, but an empty VSAM data set will cause a VSAM input error (code 160), and sort/merge will terminate.

For information on the parameters used, the conditions under which they are required, a summary of the information in the parameters, and the default values, see Figure 11. The subparameters of the DCB parameter are described similarly in Figure 12. Performance is enhanced if the LRECL subparameter of the DCB is accurately specified for variable-length records. The maximum input record length that you can specify for your particular configuration is given in the "Introduction."

See your OS/VS1 JCL Reference or OS/VS2 JCL Reference for more detailed information.

When input to the program is a concatenated data set, the following rules apply:

- RECFM must be the same for all data sets in the concatenation, except that FB and FBS can be mixed.

- BLKSIZE may vary, but the data set with the largest block size must be specified on the first DD statement of the concatenation.

- With fixed-length records, LRECL must be the same for all data sets. With variable-length records LRECL can vary, but the largest size must be specified for the data set described on the first DD statement.

- If the data sets are on unlike devices you cannot use the EXLST parameter at exit E18.

```
//SORTIN  DD  DSNAME=INPUT,DISP=(OLD,KEEP)
```

DD Example 2. SORTIN DD Statement

This example shows DD statement parameters that define a previously cataloged input data set:

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 writes 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 kept after the current job step.

```
//SORTIN   DD  DSN=SORTIN,DISP=(OLD,KEEP),UNIT=3400-3,
//             VOL=SER=(75836,79661,72945)
```

DD Example 3. Volume Parameter on SORTIN DD

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 this example, the input data set is on three reels that have serial numbers 75836, 79661, and 72945.

If a data set is not on standard-labeled tape (or disk), you must specify DCB parameters in its DD statement.

SORTINnn DD Statement

The SORTINnn DD statements describe the characteristics of the data sets in which records to be merged reside, and indicate the locations of these data sets; nn is any number from 01 through 16. The statements must be numbered in ascending order: SORTIN01 is the name of the first, SORTIN02 the name of the second, and so on. No numbers can be skipped and concatenated data sets are not supported.

SORTINnn DD statements are always needed for a merge unless the merge is invoked from another program, and all input is supplied through a routine at exit E32.

The data set with the largest block size must be defined in the SORTIN01 DD statement. The record format must be the same for all input data sets. Logical record length must also be the same unless the records are variable-length, in which case the largest size must belong to the data set described in SORTIN01.

The maximum input logical record length that you can use for your particular configuration is given in the Introduction under "Limitations on Input" (Figure 1).

The program will accept empty (null) QSAM data sets for merging, but an empty VSAM data set will cause a VSAM input error (code 160), and the program will terminate.

For further information on the parameters used in the SORTINnn DD statements, the conditions under which they are required, and the default value assumed if a parameter is not included, see Figure 11. The subparameters of the DCB parameter are described similarly in Figure 12. See your OS/VS1 JCL Reference or OS/VS2 JCL Reference for more detailed information.

Note: For MVS, FREE=CLOSE cannot be specified.

```
//SORTIN01 DD  DSNAME=MERGE1,VOLUME=SER=000111,DISP=OLD,
//             LABEL=(,NL),UNIT=3400-3,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=240)
//SORTIN02 DD  DSNAME=MERGE2,VOLUME=SER=000121,DISP=OLD,
//             LABEL=(,NL),UNIT=3400-3,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=240)
//SORTIN03 DD  DSNAME=MERGE3,VOLUME=SER=000131,DISP=OLD,
//             LABEL=(,NL),UNIT=3400-3,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=240)
```

DD Example 4. SORTIN01-03 DD Statements (Merge)

```
//SORTIN01 DD  DSNAME=INPUT1,VOLUME=SER=000101, *
//             UNIT=3330,DISP=OLD               *DCB PARAMETERS
//SORTIN02 DD  DSNAME=INPUT2,VOLUME=SER=000201, *SUPPLIED FROM
//             UNIT=3330,DISP=OLD               *LABELS
```

DD Example 5. SORTIN01-02 DD Statements (Merge)

SORTWKnn DD Statement

The SORTWKnn DD statements describe the characteristics of the data sets used as intermediate storage areas for records to be sorted; they also indicate the location of these data sets.

WHEN REQUIRED: One or more SORTWKnn statements are required for each sort application (but not a merge), unless:

- Input can be contained in main storage, or

- DYNALLOC has been specified in the SORT or OPTION statement under MVS. No SORTWK data sets should be provided if dynamic allocation is specified.

  Note: VLR-Blockset will be bypassed if no SORTWK data sets are provided.

For information on how to calculate the amount of storage needed, see Section 3.

DEVICES: SORTWK data sets can be on disk or on tape, but not both, as described in Section 3. Disk types can be mixed.

Tape must be 9-track unless input is on 7-track tape, in which case work tapes can (but need not) be 7-track.

GENERAL CODING NOTES

- In the DDname (SORTWKnn):

  - Cylinder allocation is required for FLR-Blockset and is recommended to improve performance for VLR-Blockset.

  - With disk work areas, nn can be any decimal number from 00 through 99 and numbers can be in any order (unless a nonstandard technique is forced, as described in Appendix A).

  - Unless the input file is very large, one or two SORTWK data sets are usually sufficient. One or two large SORTWK data sets are preferable to several small ones.

  - With tape work areas, nn can be 01 through 32; the first must be 01, and the rest must follow consecutively. No numbers can be skipped.

- DD DUMMY must not be used.

- Different SORTWK DD statements must not reference the same physical data set.

- No parameters relating to ASCII data should be included, since ASCII input is automatically translated into EBCDIC before being moved into an intermediate storage area.

DISK CODING NOTES

- Data sets must be sequential, not partitioned.

- The SPLIT cylinder parameter must not be specified.

- If no secondary allocation is requested, a default of one-fifth of primary space or one cylinder will be used, whichever is larger, for work data sets. (Secondary allocation is limited to 12 work data sets in the Peerage or Vale sorting techniques only.) An information message ICE085I is printed whenever secondary allocation has been used.

- If the data set is allocated to VIO, there will be no automatic secondary allocation.

- Secondary allocation can be requested for work data sets. If more work data sets are defined they are used with only the primary allocation. (Secondary allocation is limited to 12 work data sets in the Peerage and Vale sorting techniques only.)

- Primary and secondary space _must_ be on the same volume.

- If primary space is fragmented, then all but the first fragment are handled as secondary space.
- Release of disk work space not required may take place automatically.

VIRTUAL I/O: If SORTWKnn data sets are specified using virtual I/O under MVS, sort normally carries out dynamic reallocation, using the DDname SORTDKnn. However, if when sort/merge was installed the VIO option was specified, then virtual I/O will be used and performance will be degraded.

EXAMPLES: The following is an example of a SORTWKnn DD statement using a disk device:

```
//SORTWK01   DD   SPACE=(CYL,(15,5)),UNIT=3380
```

If you use the checkpoint/restart facility and need to make a deferred restart, you must make the following additions to the above statement so that the sort work data set will not be lost:

```
DSNAME=name1,DISP=(NEW,DELETE,KEEP)
```

Thus the same SORTWK DD statement for a deferred restart would be:

```
//SORTWK01   DD   DSNAME=name1,UNIT=3380,SPACE=(CYL,(15,5)),
//                DISP=(NEW,DELETE,KEEP)
```

DD Example 6. SORTWK01 DD Statement, Disk Intermediate Storage

If the sort/merge program terminates unsuccessfully and the above DD statement has been specified, the intermediate storage data set will remain in the system until the step has been successfully rerun or until the data set has been deleted by some other means.

The following is an example of a SORTWKnn DD statement using a tape device:

```
//SORTWK01   DD   UNIT=3400-3,LABEL=(,NL)
```

DD Example 7. SORTWK01 DD Statement, Tape Intermediate Storage

These parameters specify an unlabeled data set on a 3400 series tape unit. Because the DSNAME parameter is omitted, the system assigns a unique name.

SORTOUT DD Statement

The SORTOUT DD statement describes the characteristics of the data set in which the sorted or merged records are to be placed, and indicates its location. The maximum output record length (LRECL) that you can use for your particular configuration is given in the Introduction in Figure 1.

If you provide the address of an E35 exit that disposes of all output:

- A SORTOUT DD statement need not be supplied if you have invoked sort/merge from another program.

- A SORTOUT DD statement need not be supplied as long as you have a RECORD control statement if you have initiated sort/merge with an EXEC statement. Alternatively, you can use SORTOUT DD DUMMY; you can then specify unblocked format to minimize the size of the buffers reserved by the program.

For information on the parameters used in the SORTOUT DD statement, the conditions under which they are required, and the default values assumed if a parameter is not included, see Figure 11. The subparameters of the DCB parameter are similarly described in Figure 12.

Note: If LABEL=RETPD is specified in the SORTOUT DD statement for a standard labeled tape, the DCB parameters must also be specified. If the DCB parameters are not specified, the tape may be opened twice.

```
//SORTOUT DD  DSNAME=OUTPT,UNIT=3400-3, *DCB PARAMETERS DEFAULT
//            DISP=(NEW,CATLG)          *TO THOSE OF SORTIN
```

DD Example 8. SORTOUT DD Statement

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 3400-series tape unit.

DCB The DCB parameters default to those of SORTIN.

SORTMODS DD Statement

The SORTMODS DD statement describes the characteristics of a partitioned data set large enough to include all the user exit routines you include in the job input stream; it also describes the location of this data set.

The program temporarily transfers the user exit routines to the data set defined by this DD statement before they are link-edited for execution.

For information on the parameters used in the SORTMODS DD statement, the conditions under which they are required, and the default values assumed if a parameter is not included, see Figure 11.

```
//SORTMODS DD  UNIT=3340,SPACE=(TRK,(10,,3))
```

DD Example 9. SORTMODS DD Statement

These parameters allocate ten tracks of a 3340 disk to the SORTMODS data set. Space for three directory blocks is also requested.

SORTCKPT DD Statement

The SORTCKPT data set may be allocated on any device that operates with the Basic Sequential Access Method (BSAM). Processing must only be restarted from the last checkpoint taken.

```
//SORTCKPT DD  DSNAME=CHECK,VOLUME=SER=000123,
//             DISP=(NEW,KEEP),UNIT=3400-3
```

DD Example 10. SORTCKPT DD Statement

For information on the parameters used in the SORTCKPT DD statement, the conditions under which they are required, and the default values assumed if a parameter is not included, see Figure 10.

If the CKPT operand is specified on the SORT control statement, more intermediate storage may be required. See Section 3.

If you wish to use the checkpoint/restart facility, refer to OS/VS1 Checkpoint/Restart or OS/VS1 MVS Checkpoint/Restart.

SORTCNTL DD Statement

The SORTCNTL data set may be used to read changed and/or additional sort/merge control statements, when the sort is invoked from another program (written, for example, in COBOL or PL/I). When sort/merge is invoked, it will read and use all the statements present (see Note 2 below), which will then completely override corresponding statements which have been passed in the parameter list.

```
//SORTCNTL DD  *
```

DD Example 11. SORTCNTL DD Statement

Notes:

1. When sort/merge is invoked from a PL/I program, the SORTCNTL data set must not be used to supply a new RECORD control statement.

2. If you want sort/merge to try to execute one of the Blockset techniques, include only the OPTION control statement in the SORTCNTL data set. Inclusion of any other control statements (except END) will cause sort/merge to bypass Blockset and attempt to select Peerage or Vale, where appropriate.

SORTDKnn DD Statement

In an MVS system, sort work data sets can be assigned to VIO. If the ICEMAC parameter VIO is specified or defaults to NO, VIO sort work data sets are deallocated and reallocated by sort with the DD name SORTDKnn. The DD name SORTDKnn is reserved for use by the sort/merge program.

SECTION 6. USER EXIT ROUTINES

At certain places in the executable code of the sort/merge program, control can be passed to your own routines. These places are called user exits. Because each exit is located in a particular phase of sort/merge, a general understanding of how the sort/merge program operates is necessary to understand them fully.

The purpose of this section is to describe how you can use one or more user exits to achieve a specific result; it also describes the linkage conventions, register usage, and other conventions you must follow when writing your routines. User exit routines can be used during an execution of sort/merge to perform a variety of functions, such as deleting, inserting, altering, and summarizing records.

This section has two subsections. To help you use them as efficiently as possible we give here a brief description of their contents.

The first subsection contains the following topics:

Sort/Merge Program Description
explains the different phases of the sort/merge program and their connection with user exits.

Function of Routines at User Exits
describes the uses of routines at user exits, for instance, opening data sets, handling special I/O, etc.

Your Routines and Sort/Merge Performance
describes how your routines can affect the performance of the sort/merge program.

Preparing Your User Exit Routines
gives a few points to bear in mind when preparing your routines.

How to Load Your User Exit Routines
explains how the sort/merge program enters your routines and describes register conventions.

How to Link to User Exit Routines
describes return codes, linkage conventions, and restrictions associated with each of the exits.

You are strongly advised to familiarize yourself with the above background information before continuing to the second subsection which gives return codes, linkage conventions, and restrictions associated with each of the exits.

The second subsection discusses user exits. (Bear in mind that if exits other than E15 and/or E35 are specified, the Blockset techniques will not be used.) The phases that use exits are shown below with relevant exits:

SORT (INPUT) PHASE 1
Opening Data Sets/Initializing Routines—E11 Exit
Passing or Changing Input Records—E15 Exit
Handling Miscalculation of Intermediate Storage—E16 Exit
Closing Data Sets—E17 Exit
Handling Input Data Sets—E18 Exit
Handling Output to Work Data Sets—E19 Exit

| **INTERMEDIATE MERGE PHASE 2** (not used by Blockset)
Opening Data Sets/Initializing Routines—E21 Exit
Changing Records—E25 Exit
Closing Data Sets—E27 Exit
Handling Input—E28 Exit
Handling Output—E29 Exit

| **MERGE (OUTPUT) PHASE 3**
Opening Data Sets—E31 Exit
Passing or Changing Input Records to a Merge—E32 Exit
Adding, Deleting, or Changing Output Records—E35 Exit
Closing Data Sets—E37 Exit
Handling Input Data Sets to a Merge—E38 Exit
Handling Output Data Sets—E39 Exot

| **ALL PHASES**
Modifying Control Fields—E61 Exit

| Exit Naming Convention

The naming convention for exits is as follows:

Exy, where:

x is number of phase
y is number of exit within phase

The exception is E61, which can be taken in any of Phases 1-3.

SORT/MERGE PROGRAM DESCRIPTION

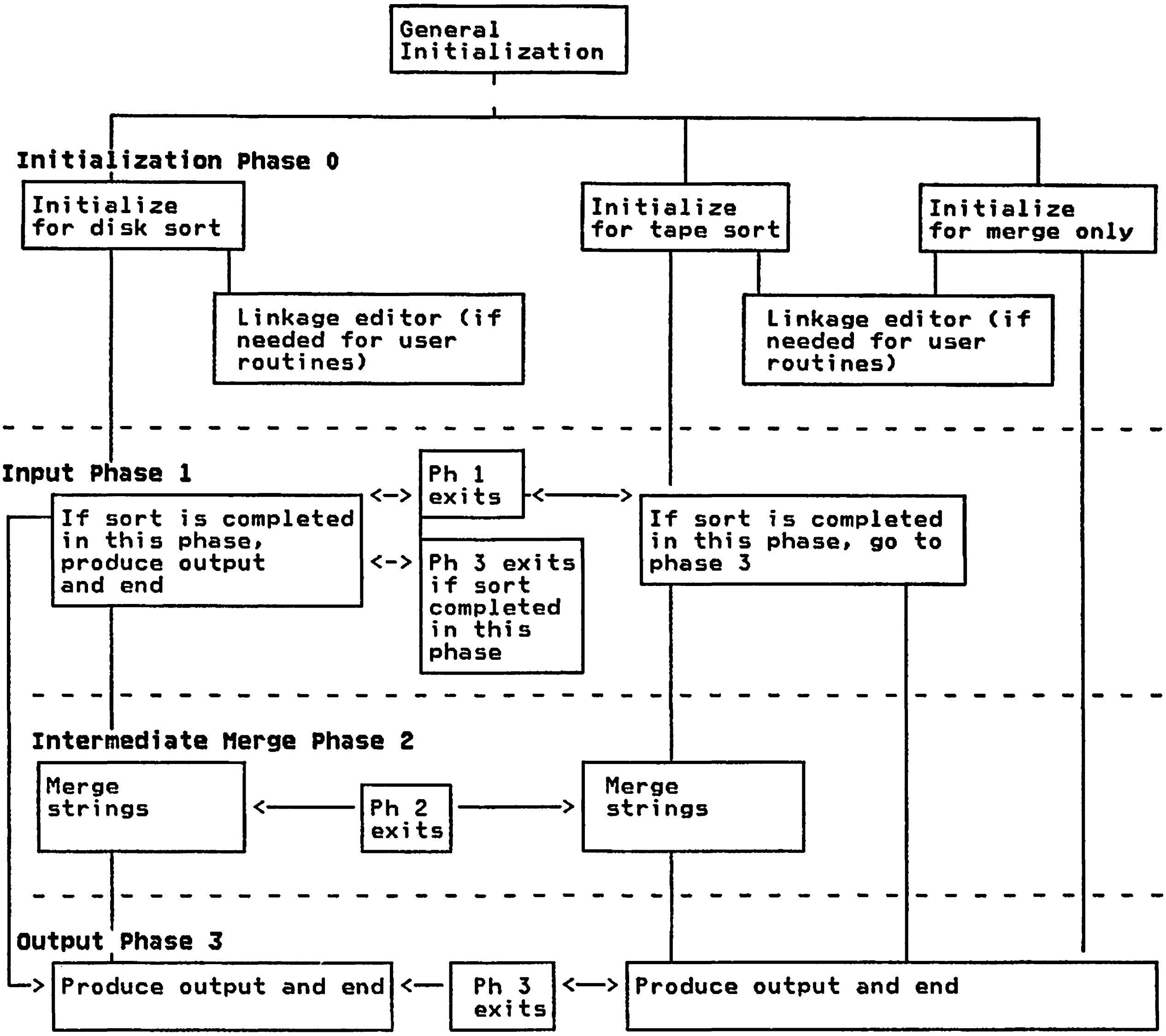
The sort/merge program is segmented into parts that can operate independently. Generally, there are two levels of segmentation:

- A phase is a large program component designed to perform one specific task (for example, final merging).

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

The total sort/merge program consists of two separate parts: one for disk sorts, and one for tape sorts; both parts have a common initialization routine. As illustrated in Figure 13, both parts operate in at least four major phases, depending on the sorting technique selected by sort/merge. All of the phases are used for sorting applications, but only two for merging operations.

Figure 13 is a phase-level flowchart of sort/merge; Figure 14 shows the various user exits, and the functions of the routines that you can write for these exits.


General Initialization
Initialization Phase 0
Initialize for disk sort
Initialize for tape sort
Initialize for merge only
Linkage editor (if needed for user routines)
Linkage editor (if needed for user routines)
Input Phase 1
Ph 1 exits
If sort is completed in this phase, produce output and end
Ph 3 exits if sort completed in this phase
If sort is completed in this phase, go to phase 3
Intermediate Merge Phase 2
Merge strings
Ph 2 exits
Merge strings
Output Phase 3
Produce output and end
Ph 3 exits
Produce output and end

Note: In addition to Initialization Phase 0, FLR-Blockset has three phases: Input Phase 1, where it reads the SORTIN data set; Key Phase 2, where it sorts the index records; and Output Phase 3, where it writes the SORTOUT data set. In addition, VLR-Blockset has a Generation phase after the Input phase where it builds code to move variable-length records.

Figure 13. Flow of Control in the OS/VS Sort/Merge Program

Exit Functions	Input Phase 1	Inter-med iate Merge Phase 2**	Output Phase 3	All
Open user data sets/initialize	E11	E21	E31	
Insert records	E15		E32,E35	
Delete/Alter records	E15	E25	E35	
Terminate the program	E15	E25	E35	
Summarize records		E25	E35	
Determine action when intermediate storage insufficient	E16*			
Close user data sets/housekeeping	E17	E27	E37	
Handle special I/O conditions: Input (incl. handling labels, read errors, EOF)	E18	E28*	E38*	
VSAM password insertion, journaling, and other VSAM exits	E18		E38*	
Output (incl. handling labels, write errors)	E19*	E29*	E39	E39
VSAM password insertion, journaling, and other VSAM exits			E39	
Modify control fields				E61
*Not valid for a standard disk sort (ignored if specified) **Phase 2 may not always be entered.				

Figure 14. Functions of Routines at Program Exits

INITIALIZATION PHASE 0

The initialization phase, which has no exits, reads and interprets program control information and decides which sorting technique will handle the application. All of the sorting techniques use this phase.

Using information obtained from the operating system and from JCL statements, it determines the optimum method of using the processor and I/O configuration available and passes control to the linkage editor, if you have routines that need link-editing.

SORT (INPUT) PHASE 1

The sort (input) phase orders the input data set into sequences and distributes them onto work data sets. There are several methods of distribution, known as string distribution techniques, and, unless a particular technique has been forced, sort/merge attempts to choose the most efficient. All sorting techniques use this phase. In the Peerage and Blockset sorting techniques, indexes are created for these distributed records.

If tape is being used for work storage, the strings can be distributed in both ascending and descending order. This enables the intermediate merge phase (using the read-backward feature) to merge the strings without rewinding tapes.

| A disk sort (except one using VLR-Blockset) can operate with no intermediate storage if the input data set can be contained in the main storage available.

The exits for this phase are shown in Figure 14.

| GENERATION PHASE (VLR-BLOCKSET ONLY)

| This phase is used by VLR-Blockset to build code to move variable-length records to output buffers.

| KEY PHASE (BLOCKSET ONLY)

| This phase is used by the Blockset techniques to sort index records.

INTERMEDIATE MERGE PHASE 2 (PEERAGE AND VALE ONLY)

This phase is loaded and executed following completion of the sort phase. It performs successive merges of the strings produced by the sort phase.

The merges are carried out from work data set to work data set, each successive merge pass decreasing the number of strings and increasing the average string length. When only one more merge is required to create a single long string (the output data set), control is given to the output phase. The user exits for this phase are shown in Figure 14.

If sufficiently few strings are produced by the sort phase, this phase (and its associated user exit routines) may be skipped. Also, with a disk sort, even if this phase is entered, not all records may be handled.

OUTPUT PHASE 3

The final merge (output) phase, used by all sorting techniques, 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 or VSAM. After execution of this phase, the sort/merge program returns control to the operating system (or invoking program). The exits for this phase are shown in Figure 14.

When the intermediate merge phase is skipped, this phase can sometimes also be skipped by a disk sort; if it is, the output phase exits will be taken (if specified) when the output data set is created in the sort (input) phase.

FUNCTIONS OF ROUTINES AT USER EXITS

Figure 14 summarizes the functions of user exit routines. Refer to it before reading the text that follows.

Note: For the Blockset techniques, use only the E15 and E35 exits. If any other exits are specified, Blockset will not be used.

LINKAGE CONVENTIONS AND PROGRAMMING LANGUAGES

User-written routines are expected to follow standard linkage conventions. They can be written in any language that provides the ability to pass the location/address of a record or parameter list in Register 1. (COBOL and PL/I users, however, are restricted by the facilities of the language.)

OPENING DATA SETS AND INITIALIZATION

You can write your own routines to open data sets and perform other forms of initialization; you must associate these routines with the E11, E21, and/or E31 exits. See Figure 14. You must also link-edit each of them together with the other routines in the same phase; otherwise, they risk being overlaid in main storage.

To check labels on input files, use the E18, E28, and E38 exits.

INSERTING, DELETING, AND ALTERING RECORDS; TERMINATING SORT

You can write your own routines to delete, insert, or alter records, or to terminate the sort/merge program. You must associate these routines with the E15, E25, E32, and/or E35 exits.

HANDLING SPECIAL I/O; VSAM EXIT FUNCTIONS

Read/Write Error Routines

Sort/merge contains six exits to handle special I/O conditions: E18, E28 and E38 for input, and E19, E29 and E39 for output. They are particularly useful for a tape sort. With a standard disk sort, all except E18 and E39 are ignored.

Note: The Blockset techniques are bypassed if any exits other than E15 and E35 are specified.

You can use them to incorporate your own or your installation's I/O error recovery routines into the sort/merge program. When sort/merge encounters an uncorrectable I/O error, it passes the same parameters as those passed by QSAM/BSAM or VSAM.

Your read and write error routines can reside in a library, or can be placed in SYSIN. Your library or SYSIN routines are brought into main storage with their associated phases.

If no user routines are supplied, and an uncorrectable read or write error is encountered, sort/merge issues message ICE061A and then terminates.

With QSAM/BSAM the following information is passed to your synchronous error routine:

- General Registers 0 and 1 are unchanged; they contain the information passed by QSAM/BSAM, as documented in the data management publications.

- General Register 14 contains the return address of sort/merge.

- General Register 15 contains the address of your error routine.

VSAM will go direct to any routine specified in the EXLST macro you passed to the sort program via the E18, E38 or E39 exits, as appropriate. Your routine must return to VSAM via Register 14. See the OS/VS VSAM Programmer's Guide for details.

Read Errors

You can write your own routines to handle I/O read errors that the operating system cannot correct; you must associate these routines with the E18, E28 and/or E38 exits. They must pass certain control block information back to the sort program to tell it whether to accept the record as it is, skip the block, or request termination. They may also attempt to correct the error.

Write Errors

You can write your own routines to handle I/O write errors that the operating system cannot correct; you must associate these routines with the E19, E29, and/or E39 exit. These routines can perform any necessary abnormal end-of-task operations before the sort/merge program is terminated.

VSAM Exit Functions

If you have VSAM input, E18 (for a sort) or E38 (for a merge) can be used to insert VSAM passwords, journal a VSAM data set, and carry out other VSAM exit functions (except EODAD), as described in more detail below. E39 can handle these functions for VSAM output.

INTERMEDIATE STORAGE CAPACITY ERRORS

You can write a routine to direct sort/merge program action if sort/merge determines that insufficient intermediate storage is available to handle the input data set; you must associate this routine with the E16 exit for tape or nonstandard disk sorts. For a tape sort, you can choose between sorting current records only, trying to complete the sort, or terminating the sort/merge program.

For more details, see "Exceeding Intermediate Storage Capacity" in Section 3.

MODIFYING CONTROL FIELDS

You can write a routine to alter control fields before sort/merge compares them. This allows you, for example, to normalize floating-point control fields. It also allows you to modify the order in which the records are finally sorted or merged, a function for which you would usually use the ALTSEQ program control statement instead. You must associate these routines with the E61 exit.

Your routine will modify the extracted image of the control fields, which is used for comparison. It does not change the original control fields. Thus your original records are not altered.

If this exit is used, the subsequent comparisons always arrange the modified control fields in ascending order.

CLOSING DATA SETS

You can write your own routines to close data sets and perform any necessary housekeeping; you must associate these routines with the E17, E27, and/or E37 exit.

To write output labels, use the E19, E29, and E39 exits.

If you have an end-of-file routine which you want to use for SORTIN, include it at the E18 exit.

USER EXIT ROUTINES AND SORT/MERGE PERFORMANCE

When you consider using user exits, you should weigh the following factors:

- Your routines occupy main storage that would otherwise be available to the sort/merge program. Because its main storage is restricted, sort/merge may need to execute extra intermediate merge phase passes. This, of course, increases sorting time.
- The execution of user exit routines adds time to the overall execution time. Later, in the descriptions of the exits, you will note that several of the exits give your routine control once for each record until you pass a 'do not return' return code to sort/merge. You should design your routines with this in mind.

PREPARING USER EXIT ROUTINES

When preparing your routines, bear the following points in mind:

- To use the user exits (other than E15/E32 and E35 in dynamically invoked applications), you must associate your routine with the appropriate exits using the MODS control statement. See "MODS Control Statement" in Section 4.
- When the disk technique is used, the entire sort/merge program is reenterable, provided your routines are reenterable and do not require link-editing from sort.
- The intermediate merge phase (and, therefore, its associated exits) may be skipped entirely if sufficiently few strings are produced in the sort (input) phase for the sort/merge program to proceed directly to the output phase—see Figure 13.
- If you are using ASCII input, remember that data presented to your routines at user exits will be in EBCDIC format (all data is represented internally in EBCDIC). If the E61 exit is used to resolve ASCII collating for special alphabetic characters, substituted characters must be in EBCDIC, but the sequencing result depends on the byte value of the ASCII translation for the substituted character.

HOW TO LOAD USER EXIT ROUTINES

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 SYSIN input stream. The sort/merge program then includes the names and locations of your routines in the list of modules to be executed during each program phase. Your routines are thus loaded and executed with their associated program phase.

No user routine will be loaded more than once in a program phase, but the same routine can appear in several different phases. For example, you can use the same Read Error routine in all three phases, but not twice in any one phase.

Only one load module will be loaded at each user exit. If you need more than one routine at an exit, and you do not load it yourself, the routines must be assembled, compiled, or link-edited as one load module. In fact, all your routines in one phase can be placed in one partitioned data set member. The member must have an entry point for each of the routines you use. When the routines are arranged in this way, 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. For example, all but one length may be specified as zero, and the total member length specified for the remaining routine.

Routines in SYSIN

The routines that you place in the SYSIN input stream are copied by the program into the SORTMODS data set; they then become input to the linkage editor.

If a routine in SYSIN is used at more than one exit you must supply one copy of the routine for each exit.

HOW TO LINK TO USER EXIT ROUTINES

The program uses a CALL macro instruction expansion to enter a user exit routine. Each routine must, therefore, contain an entry point whose name must be that of the associated program exit.

The general registers used by the sort/merge program for linkage and communication of parameters follow operating system conventions; see Figure 15.

You can return control to sort/merge with a RETURN macro instruction. You can also use this instruction to set return codes when multiple actions are available at an exit.

Your routine must save all the general registers it uses. You can use the SAVE macro to do this. If you save registers, you must also restore them; you can do this with the RETURN macro instruction.

Linkage Examples

The CALL macro instruction used by sort/merge 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. Sort/merge has already placed the register

Register	Use
1	Sort/merge places the address of a parameter list in this register.
13	Sort/merge places address of a standard save area in this register. The area may be used to save contents of registers used by your routine. The first word of the area contains the characters SM1 in its three low-order bytes.
14	Contains address of sort/merge return point.
15	Contains address of your routine. May be used as base register for your routine. This register is also used by your routine to pass return codes to sort/merge.

Figure 15. Register Conventions

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

```
  RETURN  RC=12
```

A full explanation of linkage conventions and the macro instructions discussed in this section can be found in OS/VS1 Supervisor Services and Macro Instructions or OS/VS2 MVS Supervisor Services and Macro Instructions.

E11 EXIT, OPENING DATA SETS/INITIALIZING ROUTINES

You might use routines at this exit to open data sets needed by your other routines in the associated phases, or to initialize your other routines. This routine can, if you wish, be designed for separate link-editing. Return codes are not used.

E15 EXIT, PASSING OR CHANGING RECORDS

The E15 exit is taken in the sort (input) phase. The E15 exit routine receives control once for each input record, before the record is handled by the sort. Some uses are:

- Add records to an input data set
- Pass an entire input data set to sort/merge
- Delete records from an input data set
- Change records in an input data set (but not control fields—use E61 exit for that)

If your E15 routine is inserting records from your VSAM data sets, you must build an extra 4-byte record descriptor word (RDW) at the beginning of each record before the routine passes it to sort/merge. The format of an RDW is described in the OS/VS1 Data Management Services Guide or OS/VS2 MVS System Programming Library: Data Management. (Alternatively, you could declare the records as fixed-length, and pad them to the maximum length.)

Information Supplied by Sort/Merge

The routine at E15 is entered each time a new record is brought into the sort phase. Sort/merge places the address of a parameter list in register 1. The parameter list contains the address of the new record; it 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 diagram below, which shows the format of the parameter list.

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, the program passes a zero address the first time it uses the E15 exit.

Return Codes

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

0	No Action/Record Altered
4	Delete Record
8	Do not Return
12	Insert Record
16	Terminate Sort/Merge

0—No Action
If you want the program to retain the record unchanged, place the address of the record in general register 1 and return with a zero return code.

0—Record Altered
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 with a zero return code. If your routine changes record size, you must communicate that fact to sort/merge on a RECORD statement. (See Section 4 and OS/VS1 Supervisor Services and Macro Instructions or OS/VS2 MVS Supervisor Services and Macro Instructions for further information about the length indicator and the record descriptor word.)

4—Delete Record
If you want the program to delete the record from the input data set, return with a code of 4. You need not place the address of the record in register 1.

8—Do Not Return
The program continues to return control to the user routine until it receives 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 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 the program indicates the end of the data set, which it does by passing your routine a zero address in the parameter list.

12—Insert Record
If you want the program 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 register 1 and return to the program with a return code of 12. The program 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.

16—Terminate the Program
If you want to terminate the sort/merge program, return with a code of 16. The program then returns to its calling program or to the system with a return code of 16.

Notes:

1. If you use the E15 exit, the SORTIN DD statement may be omitted, but you must include a RECORD statement in the program control statements.

2. If you use the ATTACH, LINK, or XCTL macro instruction to initiate sort/merge and also use the E15 exit, sort/merge ignores the SORTIN data set.

3. If you omit the SORTIN DD statement, all input records will be passed to sort/merge through your routine at E15: the address of each input record in turn is placed in register 1, and you return to sort/merge with a return code of 12. When sort/merge returns to the E15 exit after last record has been passed, E15 returns with RC=8 in register 15 to indicate 'do not return'.

4. Remember to build an RDW for variable-length VSAM records (see OS/VS1 Data Management Services Guide or OS/VS2 MVS Data Management Services Guide).

E16 EXIT, HANDLING INTERMEDIATE STORAGE MISCALCULATION

For a tape or nonstandard disk sort, 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. This exit is ignored for a standard disk sort, since sort/merge defaults secondary allocation to a total area of up to one-fifth of primary space or one cylinder, whichever is larger. See Section 5, under "SORTWKnn DD Statement." See also Section 3, under "Exceeding Intermediate Storage Capacity."

Note: When using magnetic tape, bear in mind that the system will have used an assumed tape length of 2400 feet. If you use tapes of a different length, the Nmax figure will not be accurate; for shorter tapes, capacity could be exceeded before "NMAX EXCEEDED" is indicated.

Return Codes

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

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 ICE054I contains the number of records that sort is continuing with. You can sort the remainder of the data set on one or more subsequent runs, using the SKIPREC operand on the SORT statement to skip over the records already sorted. Then you can merge the sort outputs to complete the operation.

4—Try to Sort Additional Records
If you want the program to continue with all of the input data set, return with RC=4. Enough space may be available for sort/merge to complete processing, if tapes are used. If enough space is not available, the sort/merge program generates a message and terminates. Refer to Section 3 under "Exceeding Intermediate Storage Capacity."

8—Terminate the Program
If you want sort/merge to terminate, return with RC=8. Sort/merge then terminates with a return code of 16.

E17 EXIT, CLOSING DATA SETS

Your routine at this exit is executed once at the end of Phase 1. It can be used to close data sets used by your other routines in the phase or to perform any housekeeping functions for your routines.

<u>E18 EXIT, HANDLING INPUT DATA SETS</u>

USE WITH QSAM/BSAM

Your routines at this exit 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 also pass a fourth DCB field (EODAD) to sort/merge.

Note: If you are using the standard disk sorting technique, the EROPT option will be ignored.

Your routines are entered first at the beginning of each phase so that the sort/merge program can obtain the parameter lists. The routines are entered again during execution of the phase at the points indicated in the parameter lists. For example, if you choose the EXLST option, sort/merge enters your E18 exit routine early in the sort (input) phase. Sort/merge picks up the parameter list, including the EXLST address. Later in the phase, sort/merge enters your routine again at the EXLST address when the data set is opened.

Information Your Routine Passes to Sort/Merge

Before returning control to sort/merge, your routine passes the DCB fields in a parameter list, the address of which is placed in general register 1. 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.

If VSAM parameters are specified, they will be accepted but ignored.

The format of the list is shown below.

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

SYNAD
: This field contains the location 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 routine. When the routine receives control, it must <u>not</u> store registers in the save area pointed to by register 13.

EXLST
: This field contains the location of a list of pointers to your routines that you want used to check labels and carry out other tasks not handled by data management. The list, and the routines to which it points, should be included in your read error routine. This parameter cannot be used at the E18 exit if the program is reading concatenated input on unlike devices from the SORTIN data sets.

EROPT
: The EROPT code is a means whereby you can specify what action the program 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 byte 4 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 must contain the appropriate address in bytes 2-4; or, if no routine is available, zeros in bytes 2 and 3, and X'01' in byte 4. You can use the assembler instruction DC AL3(1) to set up bytes 2-4.

EODAD

This field is the address of your end-of-file routine. If you specify it, the end-of-file routine must be included in your own routine.

A full description of these DCB fields is contained in the OS/VS1 Data Management Macro Instructions or OS/VS2 MVS Data Management Macro Instructions.

USE WITH VSAM

If input to your sort is a VSAM data set, you can use the E18 exit to perform various VSAM exit functions and to insert passwords in VSAM input ACBs.

Your routine is entered early in Phase 1.

RESTRICTIONS WITH VSAM

If passwords are to be entered via an exit, the data set cannot be opened during Phase 0. This means that SIZE(MAX) must not be used, as the program cannot make the necessary calculations.

Information Your Routine Passes to Sort/Merge

When you return to sort/merge, you must place in Register 1 the address of a parameter list:

X'05'	Address of VSAM exit list
X'06'	Address of password list
Fullword of zeros	

If QSAM parameters are passed instead, they will be accepted but ignored.

Either of the address entries may be omitted; if they are both included, they may be in any order.

Password List

A password list included in your routine must have the following format:

Two bytes on halfword boundary:	No. of entries in list

Followed by the 16-byte entries:

8 bytes:	DDname
8 bytes:	Password

The last byte of the DDname field will be destroyed by the sort/merge program. This list should not be altered at any time during the program. SIZE(MAX) should not be used if this function is used.

Exit List

The VSAM exit list must be built using the VSAM EXLST macro giving the addresses of your routines handling VSAM exit functions. VSAM will branch direct to your routines, which must return to VSAM via Register 14.

Any VSAM exit function available for input data sets may be used, except EODAD. If you need to do EODAD processing, write a LERAD exit and check for X'04' in the FDBK field of the RPL: this will indicate input EOD. This field should not be altered when returning to VSAM, as it is also needed by the sort/merge program.

For details, see the OS/VS VSAM Programmer's Guide.

Below is an example of code your program could use to return control to the sort.

```
         ENTRY    E18
         .
         .
E18      LA       1,PARMLST
         RETURN
         CNOP     0,4
PARMLST  DC       X'01'
         DC       AL3(SER)
         DC       X'02'
         DC       AL3(LST)
         DC       X'03'
         DC       AL3(CODE)        ADDR OF EROPT CODE
         DC       A(0)
         DC       X'04'
         DC       AL3(QSAMEOD)
         DC       X'05'
         DC       AL3(VSAMEXL)
         DC       X'06'
         DC       AL3(PWDLST)
         DC       A(0)
         .
         .
VSAMEXL  EXLST    SYNAD=USYNAD,LERAD=ULERAD
PWDLST   DC       H'2'
         DC       CL8'SORTIN'      SORTIN DDNAME
         DC       CL8'INPASS'      SORTIN PASSWORD
         DC       CL8'SORTOUT'     SORTOUT DDNAME
         DC       CL8'OUTPASS'     SORTOUT PASSWORD
USYNAD   ...                       VSAM SYNCH ERROR RTN
ULERAD   ...                       VSAM LOGIC ERROR RTN
SER      ...                       QSAM ERROR RTN
LST      ...                       EXLST ADDRESS LIST
```

E19 EXIT, HANDLING OUTPUT TO WORK DATA SETS

This exit is used to handle write error conditions in Phase 1, when the sort/merge program is unable to correct a write error to a work data set. It cannot be used if the standard disk sorting technique is used; if supplied, it is ignored.

USE WITH QSAM/BSAM

Your routines at this exit can pass to sort/merge a parameter list containing the specifications for two DCB fields (SYNAD and EXLST).

Your routines are entered first early in Phase 1 so that sort/merge can obtain the parameter lists. The routines are entered again later in the phase at the points indicated by the options in the parameter lists.

Information Your Routine Passes to Sort/Merge

Before returning control to sort/merge, your routine passes the DCB fields in a parameter list, the address of which is placed in register 1. The list must begin on a fullword boundary and must be a whole number of fullwords long. The first 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.

If VSAM parameters are passed, they are accepted but ignored.

The format is shown below.

Byte 1	Byte 2	Byte 3	Byte 4
01	SYNAD field		
02	EXLST field		
00	0	0	0

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 own routine.

EXLST
The EXLST field contains the location of a list of pointers to the routines that you want used to process labels and carry out other tasks not handled by data management. This list, and the routines to which it points, must be included as part of your own routine.

A full description of these DCB fields can be found in OS/VS1 Data Management Macro Instructions or OS/VS2 MVS Data Management Macro Instructions.

E21 EXIT, OPENING DATA SETS/INITIALIZING ROUTINES

You might use routines at this exit to open data sets needed by your other routines in Phase 2, or to initialize your other routines. This routine can, if you wish, be designed for separate link-editing. Return codes are not used.

E25 EXIT, CHANGING RECORDS

The E25 exit is taken in the intermediate merge phase, after the records have been merged. Note that this phase may not always be entered—see Figure 13.

Note also, that even when it is taken, it may not be given all of the input records. The standard disk sort usually merges only part of the input at each pass, and some records may never be handled in Phase 2.

Some Uses

This routine can be used to:

- Change records leaving the intermediate merge phase—though control fields may not be changed at this exit.

- Summarize and/or delete records (before the final merge, thus improving performance).

- Terminate sort/merge.

If the EQUALS option is used, records have been expanded 4 bytes in this phase to contain the input sequence number of the record: bytes 1 through 4, if fixed length, and bytes 5 through 8, if variable length.

Notes:

1. You cannot retain information in this routine, since the entire intermediate merge phase (including your associated routines) may be reloaded into main storage several times. Any information you wish to retain, such as a counter of the number of records processed, should therefore be carried in the records themselves.

2. This exit cannot be used in a merge-only application, nor in a sort which bypasses the intermediate merge phase.

3. If you want to summarize only (with no deletion), it is more efficient to use the E35 exit instead of E25.

4. The program does not test for equal control fields before taking the E25 exit. Therefore, if you want to summarize records with equal control fields, you must test the fields in your own routine.

Information Supplied by Sort/Merge

Your E25 exit 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:

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

- The address of the record in the output area.

The sort/merge program places the address of a parameter list that contains these two record addresses in general register 1. The parameter list starts on a fullword boundary and is two fullwords long. The first byte of each word contains zeros. The format of the parameter list is:

Byte 1	Bytes 2-4
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	No Action/Record Altered
4	Delete Record or Summarize and Delete
16	Terminate Sort/Merge

0—No Action
If you want sort/merge to retain the record unchanged in the intermediate merge sequence, load the address of the record leaving the merge into register 1 and return to the program 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—Record Altered
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).

16—Terminate Sort/Merge
If you want to terminate sort/merge, return with a code of 16. Sort/merge then returns to its calling program or the system with a return code of 16.

E27 EXIT, CLOSING DATA SETS

Your routine at this exit is executed once at the end of Phase 2. It can be used to close data sets used by your other routines in the phase, or to perform any housekeeping functions for your routines.

E28 EXIT, HANDLING INPUT FROM WORK DATA SETS

See "E18 Exit, Handling Input Data Sets" earlier in this section for details of how to use E28 with QSAM/BSAM.

If you are using the standard disk sorting technique, then I/O error conditions cannot be handled through the E28 exit. If you still want to use this exit function, you must force one of the nonstandard disk sorting techniques (BALN or CRCX) by using the DEBUG program control statement (see Appendix A).

E29 EXIT, HANDLING OUTPUT TO WORK DATA SETS

See "E19 Exit, Handling Output to Work Data Sets" earlier in this section for details of how to use E29 with QSAM/BSAM.

If you are using the standard disk sorting technique, then I/O error conditions cannot be handled through the E29 exit. If you still want to use this exit function, you must force one of the nonstandard disk sorting techniques (BALN or CRCX) by using the DEBUG program control statement (see Appendix A).

E31 EXIT, OPENING DATA SETS

You might use routines at this exit to open data sets needed by your other routines in Phase 3, or to initialize your other routines. This routine can, if you wish, be designed for separate link-editing. Return codes are not used.

E32 EXIT, HANDLING INPUT TO A MERGE ONLY

This exit can only be used in a merge-only operation which is invoked from another program, and cannot be specified on the MODS statement. If activated, it must supply all input to the merge, and the parameter list passed to the program must indicate the number of input files.

If input is variable-length VSAM records, your E32 exit routine must build an extra 4-byte record descriptor word (RDW) at the beginning of each record before handing it to the merge. The format of an RDW is described in OS/VS1 Data Management Services Guide and OS/VS2 MVS Data Management Services Guide. (Alternatively, you could declare the records as fixed length, and pad them to the maximum length.)

Information Supplied by Sort/Merge

Your E32 exit routine is entered each time the merge program requires a new input record. The program passes a two-word parameter list to your routine. The address of the list is in Register 1.

The parameter list has the format:

Bytes 1-4
Number of next file to be used for input
Space for your return parameter

Before returning control to the merge program, you must:

- Place the address of the next input record from the requested data set in the second word of the parameter list.
- Put the return code in Register 15.

Return Codes

Your routine must pass one of the following return codes to the program:

8	End of the Data Set Requested (No Record Returned)
12	Insert Record
16	End of Merge

E35 EXIT, CHANGING RECORDS

The E35 exit is taken in the output phase after the records have been merged. Some uses are:

- Add, delete, or change records in the output data set.
- Terminate sort/merge.

Notes:

1. If you use the E35 exit, the SORTOUT DD statement may be omitted, but you must include a RECORD statement in the program control statements.
2. If you use the ATTACH, LINK, or XCTL macro instruction to initiate sort/merge and also use the E35 exit, sort/merge ignores the SORTOUT data set. Your E35 exit 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 to indicate 'do not return'.
3. Remember that if input records are variable length from a VSAM data set, they will have been prefixed by a 4-byte record descriptor word (RDW).

Information Supplied by Sort/Merge

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:

- The address of the record leaving the merge which would normally follow the record in the output area. This address is zero at the end of the 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 remain zero as long as you pass a return code of 4 (delete record) to sort merge; consequently, no sequence check can be performed.

  Note: If the record pointed to is variable length, it has a record descriptor word at this point, even if output is to a VSAM data set.

Sort/merge also passes your routine a third parameter, called the sequence-check switch, 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, which is ignored for all standard disk sorts.

The list is three fullwords long and begins on a fullword boundary. The high-order bytes of the first two words are not used. The format of the parameter list is:

Byte 1	Byte 2	Byte 3	Byte 4
xx	Address of Record Leaving Merge		
xx	Address of Record in Output Area		
00	00	00	Sequence check switch for inserted records: X'00' (check on) or X'04' (check off) (Ignored for standard disk sorts)

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	No Action/Record Altered
4	Delete Record
8	Do Not Return
12	Insert Record
16	Terminate Sort/Merge

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

0—Record Altered
If you want to change the record before having it placed output data set, move the record to a work area, make the change, load the address of the modified record into register 1, and return to sort/merge with a zero return code. If you change record size, you must communicate that fact to the sort/merge program in 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 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 register 1. Unless you are inserting records after the end of the data set, you must pass RC=8 when sort/merge indicates the end of the data set, which it does by passing your routine zero as the address of the record leaving the merge.

If you do not have a SORTOUT data set and would normally return with RC=8 before EOF, you can avoid getting the ICE025A message by specifying NOCHECK on the OPTION control statement (if CHECK=NO had not already been specified at installation time).

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 register 1 and return to sort/merge with a return code of 12. Sort/merge returns to your routine with the same address as before for the record leaving the merge, and places the address of the inserted record into the output area, so you can make more insertions at that point, or delete the record leaving the merge. Sort/merge does not perform sequence checking for standard disk sorts. For tape and nonstandard disk sorts, sort/merge does not perform sequence checking 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.

16—Terminate Sort/Merge

If you want to terminate sort/merge, return with a code of 16. Sort/merge then returns to its calling program or the system with a return code of 16.

Summarizing Records

You can summarize records in the ouptut 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 the same record remains in the output area, so that you can summarize further. If you do not delete the record leaving the merge, that record is added to the output area, and its address takes the place of the address of the previous record in the output area; sort/merge returns with the address of a new record leaving the merge.

E37 EXIT, CLOSING DATA SETS

Your routine at this exit is executed once at the end of the output phase. It can be used to close data sets used by your other routines in the phase or to perform any housekeeping functions for your routines.

E38 EXIT, HANDLING INPUT DATA SETS

Same as for E18. If you are using the standard disk sorting technique, then I/O error conditions cannot be handled through E38. If you still want to use this exit function, you must force one of the nonstandard disk sorting techniques (BALN or CRCX) by using the DEBUG program control statement (see Appendix A).

E39 EXIT, HANDLING OUTPUT DATA SETS

Same as for E19 for BSAM/QSAM. Same as for E18 for VSAM.

E61 EXIT, MODIFYING CONTROL FIELDS

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 parameter on the SORT or MERGE control statement must be specified for control fields changed by this routine as described in Section 4.

Some Uses

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 by the operating system before modifying control fields.

If you simply want to modify the collating sequence of EBCDIC data, for example, to permit the alphabetic collation of national characters, you can do so without the need for an E61 exit routine by use of the ALTSEQ control statement, as described in Section 4.

Information Supplied to Your Routine by Sort/Merge

Sort/merge places the address of a parameter list in register 1. The list begins on a fullword boundary and is three fullwords long. It contains the number (in hexadecimal) of the control field in the last byte of the first word; the address of the control field in the bytes 2 through 4 of the second word; and the length of the control field (in hexadecimal) in the bytes 3 and 4 of the third word. The control field length allows you to write a more generalized modification routine.

The parameter list for the E61 exit is as follows:

Byte 1	Byte 2	Byte 3	Byte 4
00	00	00	C.F. number
00	Address of Control Field Image		
Not used		Control Field Length 0001-0100	

The control field address passed to your routine is that of an extract area to which the program has moved the control field, separate from the record. Your routine, in effect, changes an image of the control field and not the control field itself.

For all fields except binary, the total number of bytes sort/merge passes to your routine is equal to the length specified in the m parameter of the SORT or MERGE statement.

All binary fields passed to your routine contain a whole number of bytes. If a binary field does not begin and end on a byte boundary, sort/merge pads it with zeros at the beginning and/or end. If the control field is greater than 256 bytes in length, sort/merge splits it up into fields of 256 bytes each and passes them one at a time to your routine.

Your routine cannot physically change the length of the control field. If you must increase the length for collating purposes, you must previously specify that length in the m parameter of the SORT or MERGE statement. If you must shorten the control field, you must pad it to the specified length before returning it to the sort/merge program. The field your routine returns to sort/merge must contain the same number of bytes as when the routine was entered.

Modified control fields are always ordered into absolute ascending sequence, that is, they are treated as if they were binary fields (or character ASCII, if ASCII data is being used). If you need some other sequence, you could modify the fields further; for example, if after carrying out your planned modification, and before handing back control to the sort/merge program, you reverse all bits, the field will in effect be collated in absolute descending order. You will not have affected the record itself, since it is only an extracted image you are modifying.

Note that if E61 is used to resolve ASCII collating for special alphabetic characters, substituted characters must be in EBCDIC, but the sequencing result depends upon the byte value of the ASCII translation for the substituted character.

SAMPLE ROUTINES FOR PROGRAM EXITS

E15: DELETING EXPIRED RECORDS

This routine checks each record's expiration date, and deletes records which are obsolete.

```
E15      CSECT
         USING   *,12                SET UP BASE REGISTER
         SAVE    (14,12)             SAVE REGISTERS
         LR      12,15               LOAD BASE REGISTER
         ST      13,SAVEAREA+4       CHAIN BACKWARD
         LR      11,13
         LA      13,SAVEAREA
         ST      13,8(11)            CHAIN FORWARD
*
         L       2,0(1)              LOAD ADDR OF RECORD INTO R2
         LA      2,0(,2)             CLEAR FIRST BYTE
         LTR     2,2                 IS ADDR=0?
         BZ      EMPTEST             YES-TEST FOR NO INPUT
         CLI     FIRSTIME,C'Y'       IS IT FIRST TIME THROUGH
         BNE     AROUND              BRANCH IF NO
         TIME    DEC                 OBTAIN TODAY'S DATE
         MVI     FIRSTIME,C'N'       INDICATE NOT FIRST TIME ANY MORE
         ST      1,DATE              SAVE DATE
RECDATE  EQU     4
DATLEN   EQU     4
RECBASE  EQU     2
AROUND   CLC     RECDATE(DATLEN,RECBASE),DATE   CHECK EXPIRATION DATE
         BNH     DELETE              IF OBSOLETE, DELETE RECORD
         L       13,SAVEAREA+4       RESTORE R13
         LM      14,12,12(13)        RESTORE REGS
         L       1,0(1)              POINT TO REC LEAVING MERGE
         SR      15,15               RC=0 (NO ACTION)
         BR      14
EMPTEST  CLI     FIRSTIME,'Y'        IS THIS FIRST RECORD?
         BNE     NORETRET            NO-END OF DATA SET
         L       13,SAVEAREA+4       YES-INPUT DATA SET EMPTY
         RETURN  (14,12),RC=16       'TERMINATE SORT' CODE
NORETRET L       13,SAVEAREA+4       RESTORE R13
         RETURN  (14,12),RC=8        'NO RETURN' CODE
DELETE   L       13,SAVEAREA+4       RESTORE R13
         RETURN  (14,12),RC=4        'DELETE' CODE
*
SAVEAREA DS      18F
DATE     DS      F
FIRSTIME DC      C'Y'
         END
```

E16: WHEN NMAX EXCEEDED, SORT CURRENT RECORDS

This routine tells the program to sort only the records it has already read in, when it issues the message "NMAX EXCEEDED."

```
E16    CSECT
       LA      15,0      SET RETURN CODE
       BR      14
       END
```

E35: SUMMARIZE WHEN CONTROL FIELDS EQUAL

This routine checks a control field (4 bytes starting at byte 4) in the current record with the same control field in the previous record. If they are equal, a 4-byte field starting at byte 8 is summarized. If they are not, no action is taken.

```
E35    CSECT
       USING   *,12            ASSIGN BASE REGISTER
       SAVE    (14,12)         SAVE REGISTERS
       LR      12,15           LOAD BASE REGISTER
       ST      13,SAVEAREA+4   *
       LR      11,13           *
       LA      13,SAVEAREA     * SAVE AREA CHAINING
       ST      13,8(11)        *
       LM      2,3,0(1)        LOAD PARAMETER REGS
*
* REG2 NOW HAS ADDR OF RECORD LEAVING MERGE
* REG3 HAS ADDRESS OF RECORD IN OUTPUT AREA
*
       LTR     2,2             ZERO AT END OF DATA
       BZ      DONOTRET
       LTR     3,3             ZERO FIRST TIME THROUGH
       BZ      NOACTRET
       CLC     4(4,2),4(3)     COMPARE CONTROL FIELDS
       BNE     NOACTRET        IF NOT EQUAL, RETURN
* SUMMARIZE:
       L       2,8(2)          GET AMT FR RECORD LEAVING MERGE
       A       2,8(3)          ADD
       ST      2,8(3)          STORE IN OUTPUT RECORD
       L       13,SAVEAREA+4
       RETURN  (14,12),RC=4    RETURN WITH 'DELETE' CODE
*
NOACTRET L     13,SAVEAREA+4
       LM      14,12,12(13)    RESTORE REGISTERS
       SR      15,15           RC=0 (NO ACTION)
       L       1,0(1)          POINT TO RECORD LEAVING MERGE
       BR      14
DONOTRET L     13,SAVEAREA+4
      RETURN  (14,12),RC=8   'DO NOT RETURN' CODE
*
SAVEAREA DS    18F
      END
```

E35: DELETING RECORDS

This routine checks byte 5 of each record. If the byte contains the letter 'N', it deletes the record.

```
E35     CSECT
        USING   *,15
        SAVE    (14,12)       SAVE REGISTERS
        L       1,0(1)        R1 GETS ADDR OF REC FR PARAMLIST
        LTR     1,1           IS ADDR ZERO?
        BZ      NOINPUT       YES-END OF INPUT
        CLI     4(1),X'D5'    DOES BYTE 5 CONTAIN 'N'?
        BE      DELETE        YES-DELETE RECORD
        RETURN  (14,12),RC=0  RETURN WITH 'NO ACTION' CODE
NOINPUT RETURN  (14,12),RC=8  RETURN WITH 'DO NOT RETURN' CODE
DELETE  RETURN  (14,12),RC=4  RETURN WITH 'DELETE' CODE
SAVEAREA DS     18F
        END
```

SECTION 7. INITIATING A PROGRAM USING SYSTEM MACRO INSTRUCTIONS

This section describes how you can initiate execution of the sort/merge program from within your own program (if written in assembler language) with a system macro instruction, instead of with the EXEC job control statement in the input stream.

Sort/merge can also be invoked from programs written in COBOL or PL/I. How to do this is described in the relevant COBOL and PL/I programmer's guides. JCL requirements are, however, the same as for assembler.

SYSTEM MACRO INSTRUCTIONS

System macro instructions are macro instructions provided by IBM for communicating service requests to the control program. You can use these instructions only when programming in assembler language; they are processed by the assembler program using macro definitions supplied by IBM and were placed in the macro library when the control program under which you operate was generated.

You can specify one of three different system macro instructions to pass control to the program: LINK, ATTACH, or XCTL.

When you issue one of these instructions, the first load module of the sort/merge program is brought into main storage. The linkage relationship between your program and the sort/merge program differs according to which of the instructions you have used. For a complete description of the macro instructions and how to use them, you will need to refer to OS/VS1 Supervisor Services and Macro Instructions or OS/VS2 MVS Supervisor Services and Macro Instructions.

RETURN CODES

Sort/merge returns a return code to the operating system (or other invoking program) upon successful completion. If completion is unsuccessful, a return code or an ABEND is issued, depending on what was specified at installation time. This code may be interrogated by succeeding job steps. The codes are:

0	Successful Completion
16	Unsuccessful Completion

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

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

HOW TO USE THE MACROS

In order to initiate execution of the sort/merge program with a system macro instruction, you must:

- Write the required job control language DD statements.
- Write the sort/merge program control statements as operands of assembler DC instructions.
- Write a parameter list containing the addresses of the program control statement images and other information to be passed to the sort/merge program. You must also write a pointer containing the address of the parameter list to pass to sort/merge.
- Prepare the macro instruction, in which you must specify the entry point name of sort/merge.
- The save area passed to sort/merge must begin on a fullword boundary.

In addition, the following rule applies for a disk sort:

- If you are invoking sort/merge recursively (for example, from E15 or E35 exit), you must always wait for the last invoked sort to end before you can give control back to any of your exits in an earlier invoked sort.

JCL DD STATEMENTS

JCL DD statements of the kind shown in Figure 16 are usually required. The statements and their necessary parameters are described in Section 5.

//GO.SORTLIB[1]	DD (parameters) Defines the data set containing the sort/merge program modules.
//GO.SORTIN	DD (parameters) Defines the data set to be sorted. Not needed if you activate exit E15.
//GO.SORTINnn	DD (parameters) Defines data sets to be merged (for a merge-only). Not needed if you activate exit E32.
//GO.SORTWKnn	DD (parameters) Defines work data sets. Needed for most sorting applications but not for a merge-only.
//GO.name[2]	DD SYSOUT=A Defines the output data set for sort/merge messages.
//GO.SORTOUT	DD (parameters) Defines the output data set. Not needed if you handle output through E35.

[1] The 'GO' prefixes are needed if you are assembling, linking and running your program in one job, using the cataloged procedure for assembling, linking and executing, as provided by IBM.

[2] A DDname is specified when the program is installed, for use when initiating the program by a macro instruction. Default is SYSOUT. You can use either (a) the name assigned at generation time, or (b) any other valid DDname of your choice, which you must then communicate to the program in the parameter list.

Figure 16. Example of DD Statements for an Invoked Sort

PROGRAM CONTROL STATEMENT IMAGES

The program control statements described in Section 4 are usually provided in the form of character constants defined by assembler DC instructions. Their addresses must be given in a parameter list. The rules for preparing the program control statements are:

- They must be in EBCDIC format.
- The SORT (or MERGE) and RECORD statements are always required. If E15 is specified, the RECORD statement must include the LENGTH parameter.
- The MODS statement is required only when exits other than E15 and E35 are to be used.
- ALTSEQ can be used to modify the EBCDIC collating sequence, as described in Section 4.
- DEBUG can be used to obtain detailed information on program execution, as described in Appendix A.
- At least one blank must follow the operation definer (SORT, MERGE, RECORD, ALTSEQ, DEBUG, or MODS). A control statement must start with one or more blanks and end with at least one blank. No other blanks are allowed.
- The content and format of the statements are as described in Section 4, except:
  - Labels are not allowed.
  - No continuation character is allowed (the statements are not specified in card image format).

- No comments are permitted.
- If you use ATTACH to initiate the program, you cannot use the checkpoint/restart facility and, therefore, should not specify CKPT in the SORT statement image.

SORT Statement Image Example

```
SORTBEG   DC   C' SORT FIELDS=(10,15,CH,A)'
SORTEND   DC   C' '
```

This form, with a trailing blank separately defined, allows you to refer to the last byte of the statement (SORT statement end address) by the name SORTEND.

Register 1

Address of pointer

HEX	DEC	X'80'	Pointer to beginning of the parameter list
-2	-2	Unused	Number of bytes in following list[1]
2	2	Starting address of SORT or MERGE statement[1]	
6	6	X'00'	Ending address of SORT or MERGE statement[1]
A	10	X'00'	Starting address of RECORD statement[1]
E	14	X'00'	Ending address of RECORD statement[1]
12	18	X'00'	Address of E15 or E32 routine (zeros if none)[1]
16	22	X'00'	Address of E35 routine (zeros if none provided)[1]
1A	26	X'02'	Starting address of MODS statement[2]
1E	30	Ending address of MODS statement[2]	
22	34	X'00'	Optional main storage value (hex)[3]
26	38	X'01'	Optional reserved main storage value (hex)[3]
2A	42	X'03'	Starting address of message DDname[3]
2E	46	X'04'	Number of input files to a merge-only (4)[3],[4]
32	50	X'05'	Starting address of DEBUG statement[3],[5]
36	54	Ending address of DEBUG statement[3],[5]	
3A	58	X'06'	Starting address of ALTSEQ statement[3],[6]
3E	62	Ending address of ALTSEQ statement[3],[6]	
42	66	X'F6'	Pointer to ALTSEQ translation table[3]
46	72	X'FE'	Pointer to 104-byte STAE work area (or zeros)[3]
4A	74	X'FF'	Message option (FLAG)[3]
4E	78	Optional characters for DDnames[3]	
52	82	Characters for DIAG (diagnostic messages option)[3]	
56	86	Optional sequence distribution characters[3]	

[1] Required entries which must appear in the relative positions shown.

[2] Optional entries which, when included, must appear in the relative positions shown.

[3] Optional entries which must appear directly after the other entries. They can appear in any order, except that those identified by [5] and [6] must be consecutive as shown.

[4] Must appear if the MERGE statement is present, and input is supplied through E32.

[5] Must appear in consecutive order.

[6] Must appear in consecutive order.

Figure 17. The Parameter List when Attaching the Program

PARAMETER LIST

Figure 17 shows the format of the parameter list and the pointer containing its address which you must pass to the sort/merge program. Detailed specifications for each of the entries in the parameter list follow.

Byte

-2 to -1 Unused. This halfword must begin on a fullword boundary.

0 to +1 The byte count. This halfword contains the length of the parameter list. The length is specified in bytes, in hexadecimal. This halfword is not included when counting the number of bytes occupied by the list.

The total length of the required entries is 24 (X'0018'). All optional entries are four bytes long, except those referring to control statement images, which are each eight bytes long.

2-5 The starting address of the SORT or MERGE statement image. Must be in the last three bytes of this fullword.

6-9 The ending address of the SORT or MERGE statement image. Must be in the last three bytes. The first byte must contain X'00'.

10-13 The starting address of the RECORD statement image. Must be in the last three bytes. The RECORD statement must include the LENGTH parameter if E15 is specified. The first byte must contain X'00'.

14-17 The ending address of the RECORD statement. Must be in the last three bytes. The first byte must contain X'00'.

18-21 The address of your E15 or E32 routine, if any; otherwise, all zeros. Must be in the last three bytes. The first byte must contain X'00'.

22-25 The address of your E35 routine, if any; otherwise, all zeros. Must be in the last three bytes. The first byte must contain X'00'.

26-29 The starting address of the MODS statement image. Must be in the last three bytes. (If present, it must be in this location.) The first byte must contain X'02'.

30-33 The ending address of the MODS statement image. Must be in the last three bytes. (If the MODS statement image is present, this entry must be in this location in the list.)

34-37 Main storage value (optional). The first byte must contain X'00'. The next three bytes contain either the characters MAX or a hexadecimal value. This value will override the SIZE option default, provided it is greater than the MINLIM value set at sort/merge installation time.

38-41 A reserved main storage value (optional). The first byte must contain a hexadecimal one (X'01'). The next three bytes contain a hexadecimal value that specifies a number of bytes to be reserved. This space is usually required for data handling by the invoking program while sort/merge is executing. The amount of space required depends upon what routines you have, how the data is stored, and which access method you use. The reserved space is not meant for the executable code itself. This space is in addition to the value specified in RESINV at installation time.

42-45 Message DDname (optional). The DDname for the output data set for program messages is assigned at generation time, either by default (in which case it is SYSOUT) or explicitly. If you wish to use a different name, you can do so. You must then include this parameter.

The first byte must contain X'03'. The following three bytes contain the address of an eight-byte field containing the name, padded with blanks if necessary. The name can be any valid DDname. Make sure it is unique.

46-49 Number of input files to a merge. This entry must be present if the MERGE statement image is present and input to the merge is being supplied through the E32 exit. The first byte must contain X'04'. The next three bytes contain the number of files, in hexadecimal.

50-53 The starting address of the DEBUG control statement image. The first byte must be X'05'.

54-57 The ending address of the DEBUG control statement image. Must be in the last three bytes.

58-61 The starting address of the ALTSEQ control statement image. The first byte must be X'06'.

62-65 The ending address of the ALTSEQ control statement image. Must be in the last three bytes.

66-69 Pointer to a 256-byte translate table supplied instead of an ALTSEQ statement. If this parameter is present, the '06' parameter is ignored. The first byte must contain 'F6'.

70-73 If the first byte contains X'FE', the STAE routine you provide will receive control. You can also include in the last three bytes the address of a 104-byte save area where the STAE work area will be saved; otherwise, these bytes must contain zeros. If this option is omitted, no STAE routine will receive control at program failure.

74-77 The message option. The first byte must contain X'FF'. The following three bytes contain the characters NOF, (I), or (U). This parameter replaces the FLAG option of the PARM field in the EXEC statement and specifies the printing of messages as follows:

NOF No messages printed;
critical messages appear on the console.
(I) All messages printed;
critical messages also appear on the console.
(U) Only critical (uncorrectable) messages are printed; they also appear on the console.

For compatibility reasons, the form MSG={NO|CC|CP|AC|AP|PC} is accepted in place of the flag parameter. These options may, therefore, still be specified in the parameter list, as described in the

OS Sort/Merge Programmer's Guide, relating to the Program Product 5734-SM1.

Note: In systems with multiple console support, diagnostic messages are printed on the system master console.

78-81 Characters for DDnames (optional). You must use this option when you dynamically invoke two or more program applications to execute at the same time.

The four characters must all be alphameric or national ($, #, or @). The first character must be alphabetic; otherwise, the four characters are ignored. Note also that you must not use characters that conflict with other parameters: do not use PEER, BALN, OSCL, POLY, CRCX, or DIAG.

Example: If you use ABC# as replacement characters, the statements SORTIN, SORTCNTL, SORTWKnn and SORTOUT will be converted internally to ABC#IN, ABC#CNTL, ABC#WKnn, and ABC#OUT.

82-85 The DIAG diagnostic message option. These four bytes contain the characters DIAG, normally specified in the PARM field of the EXEC statement. This option is a diagnostic aid at execution time when tape is used as work space or for a merge only application. However, it can impair program efficiency, so it should be specified only when you need a debugging tool.

For details about this option see "'PARM' Field Options" in Section 5.

86-89 Four characters defining the tape sequence distribution technique, normally specified in the PARM field of the EXEC statement; can contain one of the following valid entries: BALN, OSCL, or POLY. For further details, see "'PARM' Field Options" in Section 5.

The entries PEER and CRCX are accepted but ignored.

Examples of Parameter List

The following is an example of the format of the parameter list when choosing only one option: specifying a main storage value for program execution.

(Hex)(Dec)

-2	-2	Unused	X'001C'
2	2	Starting address of SORT statement	
6	6	Ending address of SORT statement	
A	10	Starting address of RECORD statement	
E	14	Ending address of RECORD statement	
12	18	Address of E15 routine	
16	22	Address of E35 routine	
1A	26	X'00'	Main storage value (in hexadecimal)

The following is an example of the format of the parameter list when you invoke a merge, supply input through exit E32, and wish control to be handed to the merge program's STAE routine if the program fails.

(Hex)	(Dec)		
-2	-2	Unused	X'0020'
2	2	Starting address of MERGE statement	
6	6	Ending address of MERGE statement	
A	10	Starting address of RECORD statement	
E	14	Ending address of RECORD statement	
12	18	Address of E32 routine	
16	22	Zeros (no E35 routine provided)	
1A	26	X'04'	Number of input files
1E	30	X'FE'	(Zeros—no work area address provided)

WRITING THE MACRO INSTRUCTION

When writing the LINK, ATTACH, or XCTL macro instruction, you must:

- Specify SORT (the entry point) in the EP parameter of the instruction. (This applies to both sorting and merging applications.)

- Load the address of the pointer to the parameter list into Register 1 (or pass it in the MF parameter of the instruction).

Note: If you are using ATTACH, you will probably also need the ECB parameter.

If you provide an E15 exit routine, the sort/merge program will ignore the SORTIN data set; your E15 exit routine must pass all input records to the sort program. The same applies for a merge if you specify an exit E32 address. This means that your routine must issue a return code of 12 ('insert record') until the input data set is complete, and then a return code of 8 ('do not return').

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

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

When a single task attaches two or more program applications, you must modify the standard DDnames (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 DDnames in SM1 program modules. See "Passing Parameters to the Sort."

If you ATTACH more than one sort/merge application from the same program, you will have to wait for the first to complete before attaching the next, and so on—unless the application is a standard disk sort, in which case the program is reenterable (provided that any exit routines you use are also reenterable).

When you initiate sort/merge via XCTL, you must give special consideration to the area where the parameter list, address list, optional parameters, and modification routines (if any) are stored. This information must not reside in the module that issues the XCTL, because the module will be 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 sort/merge in general register 1.

EXAMPLES

Three examples follow. The first illustrates passing parameters to the sort/merge program. The second is an assembler language coding example that shows how to set up the parameter list, address list, and optional fields; the third example shows how to use the SORTCNTL DD statement.

Example 1. Passing Parameters to the Program

Figure 18 shows how a parameter list might appear in main storage.

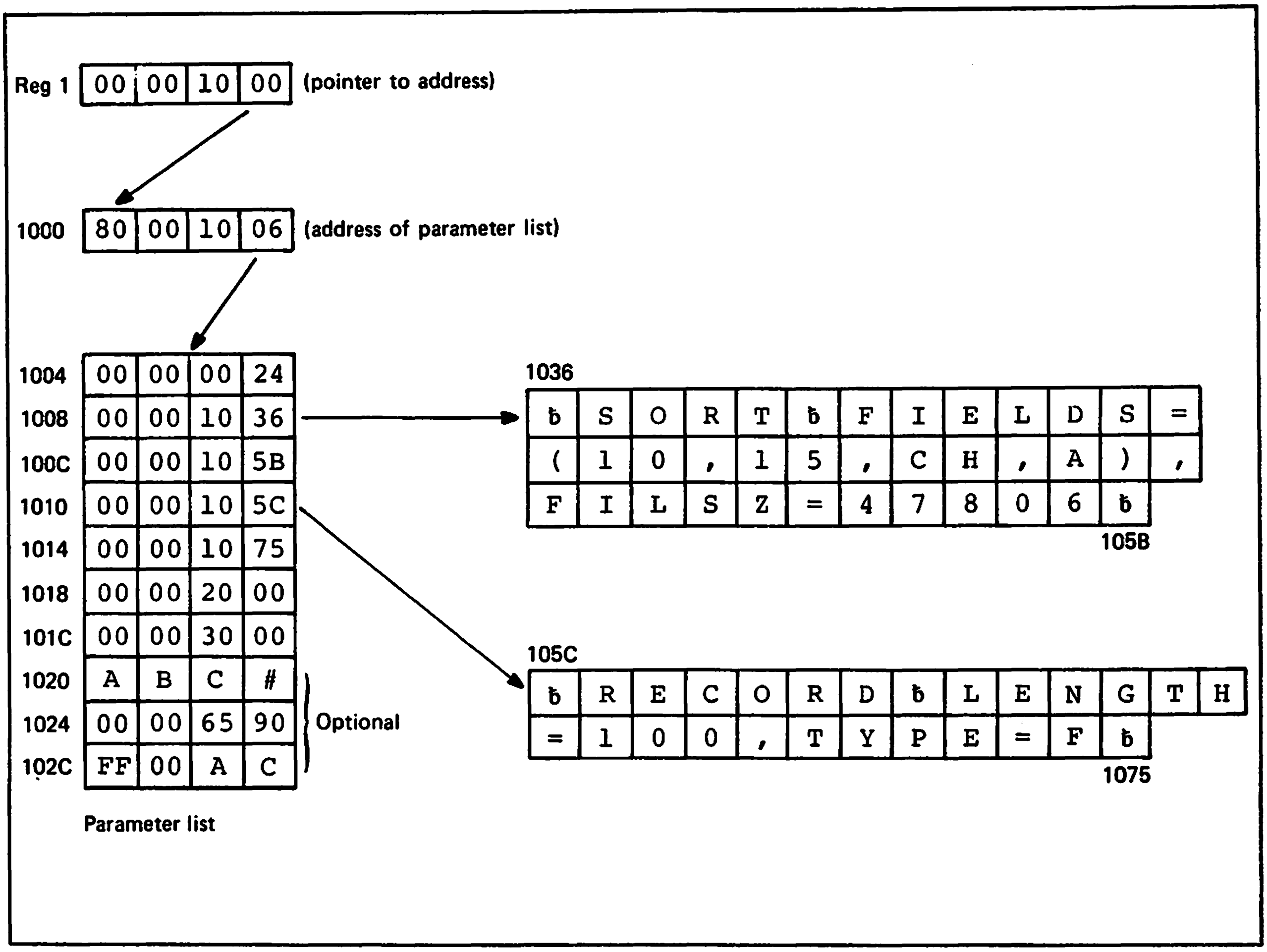


Figure 18. Passing Parameters to the Program

General register 1 contains a pointer to the address of the parameter list, which is at location 1000. The address points to the parameter list, which begins at location 1006. The first halfword of the parameter list contains, right-justified in hexadecimal, the number of bytes in the list (36 decimal).

The first two fullwords in the parameter list point to the beginning (location 1036) and end (location 105B) of the SORT control statement. The next two fullwords point to the beginning (location 105C) and end (location 1075) of the RECORD statement.

The fifth and sixth fullwords in the list contain the entry point addresses for the E15 exit (location 2000) and E35 exit (location 3000).

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

The next two fullwords in the list specify a main storage value for this application and a message option.

Example 2. Coding a Parameter List

The example in Figure 19 shows, in assembler language coding, how to set up the parameters and card images in Figure 18, and how to pass control to the program.

```
        LA   1,PARLST                 LOAD ADDR OF PARAM POINTER IN R1
        ATTACH EP=SORT                INVOKE SORT
        .
        .
        .
PARLST  DC   X'80',AL3(ADLST)         POINTER FLAG/ADDRESS OF PARAM LIST
        .
        .
        .
        CNOP 2,4                      ALIGN TO CORRECT BOUNDARY
ADLST   DC   AL2(LISTEND-LISTBEG)     PARAM LIST LENGTH
LISTBEG DC   A(SORTA)                 BEGINNING ADDRESS OF SORT STMT
        DC   A(SORTZ)                 END ADDRESS OF SORT STMT
        DC   A(RECA)                  BEGINNING ADDR OF RECORD STMT
        DC   A(RECZ)                  END ADDR OF RECORD STMT
        DC   A(MOD1)                  ADDR OF E15 RTN
        DC   A(MOD2)                  ADDR OF E35 RTN
        DC   C'ABC#'                  DDNAME CHARACTERS
        DC   F'72000'                 OPTIONAL MAIN STORAGE VALUE
        DC   X'FF'                    MESSAGE OPTION FLAG BYTE
        DC   C'(U)'                   MESSAGE OPTION
LISTEND EQU  *
SORTA   DC   C' SORT FIELDS=(10,15,CH,A),'   SORT CONTROL STMT
        DC   C'FILSZ=4780'            (CONTINUED)
SORTZ   DC   C' '                     DELIMITER
RECA    DC   C' RECORD LENGTH=100,TYPE=F'    RECORD CONTROL STMT
RECZ    DC   C' '                     DELIMITER
        DS   0H
        USING *,15
MOD1    (routine for exit E15)
        .
        .
        USING *,15
MOD2    (routine for exit E35)
```

Figure 19. Coding the Parameter List

Example 3. Using the SORTCNTL DD Statement

Sort/merge must be dynamically invoked to be able to use the SORTCNTL data set. By using the SORTCNTL DD statement, you can change or add sort/merge program control statements in an invoked program without recompiling the invoking program.

If you want to change an existing program control statement, you must respecify the complete statement.

For example, if you have a COBOL program that is invoking sort/merge to sort on the same fields as those specified in Figure 19, but you want to change the SORT statement to include the EQUALS parameter, and add the DEBUG statement, your input stream could be:

```
//COBSRT EXEC  PGM=COBSRT
//SYSOUT DD SYSOUT=A
//SORTWK01  DD  UNIT=SYSDA,SPACE=(CYL,(5))
//SORTWK02  DD  UNIT=SYSDA,SPACE=(CYL,(5))
//SYSIN     DD  *
   Input to your COBOL program
/*
//SORTCNTL  DD  *
      SORT FIELDS=(10,15,CH,A),FILSZ=4780,EQUALS
      DEBUG ABEND
/*
```

By specifying only the OPTION control statement in the SORTCNTL data set (see below), you can cause sort/merge to try to execute one of the Blockset techniques rather than being restricted to the Peerage or Vale techniques. If you specify any control statements other than OPTION in the SORTCNTL data set, one of the latter two techniques will be used.

```
.
.
.
//SORTCNTL  DD  *
     OPTION FILSZ=4780,EQUALS
/*
```

SECTION 8. IMPROVING PROGRAM EFFICIENCY

The sort/merge program automatically optimizes performance by analyzing the information given to it. This automatic optimization results in setting of optimization variables (such as buffer sizes) and selecting the proper sorting technique.

You can aid the program's optimization toward higher performance by:

- Avoiding installation options that are not performance oriented
- Planning your application development (including data formats) for efficient use of the program
- Being generous with main storage
- Trying to use the most efficient sorting technique
- Planning for most efficient use of work storage devices
- Specifying the input/output data set characteristics correctly
- Sparing the linkage-editor

These techniques are described in detail below.

INSTALLATION OPTIONS

You must be sure that the options you use do not result in unnecessary performance degradation to the sorting done at your installation. Specifically, BLKSET=NO, EQUALS=YES, SECALL=NO, VBLKSET=NO, and VERIFY=YES tend to degrade performance. Use these options only when absolutely necessary, and then by specifying the desired option at program execution time rather than at program installation time.

For more details on installation options and their effect on program performance, see the OS/VS Sort/Merge Installation Guide.

APPLICATION DEVELOPMENT

You should consider several factors when you design new applications. Some of these factors are discussed in the following sections.

EFFICIENT CONTROL FIELD SORTING

When you design new applications, you can improve the program's performance if you

- Put the control fields used for subsequent sorting at the beginning of your record in descending order of significance, and
- Use the most efficient control field data formats and control field descriptions.

Location of Control Fields: The following example illustrates the benefit of locating control fields at the beginning of a record.

Assume that your input record has the following layout:

A	1	B	2	C

where: 1 = the more significant sorting control field
2 = the less significant sorting control field

Internally, the program reorganizes the record fields prior to the actual sorting as follows:

1	2	A	B	C

Upon completion of the actual sorting, the record fields are restored to their original positions.

By designing your record format to conform to the second diagram, you can improve the program performance, since neither the reorganization nor the subsequent restore operation has to be performed by the program.

Control Field Data Formats and Descriptions: Whenever possible,

- Use either EBCDIC character or binary control fields.

- Place binary control fields so as to start and end on byte boundaries.

- Avoid using the alternative collating sequence character translation, since this function not only increases CPU time, but also increases the total length of the internal record.

- Specify fixed-point, packed decimal, and zoned decimal control fields (if you know they will always be positive) so that they can be sorted as if they were binary control fields.

- Use packed decimal format rather than zoned decimal, since sort/merge packs the control fields and also increases the total length of the internal record.

- If several contiguous character or binary control fields in the right order of significance are to be sorted in the same order (ascending or descending), specify them as one control field.

- Avoid overlapping control fields.

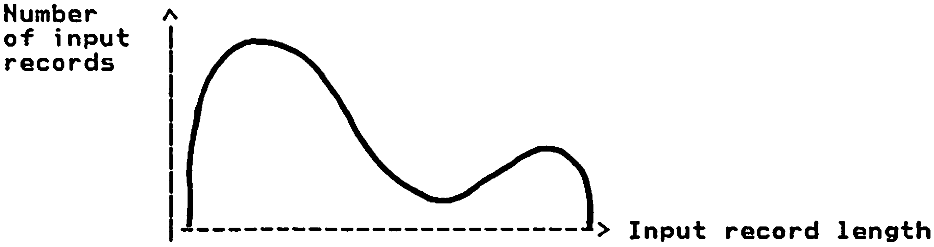
EFFICIENT BLOCKING

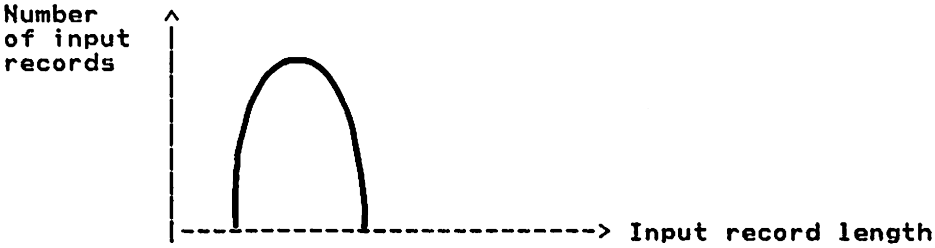
Performance of the sort/merge program is normally improved if you block input and output records.

VARIABLE-LENGTH RECORDS

You can help the program's optimization toward high performance if you

- Keep the difference between the longest and the shortest variable-length record as small as possible. By splitting your long logical record into several shorter physical records, you can achieve a record length distribution that improves the program's performance. The following two diagrams illustrate unfavorable record length distribution (top) and favorable record length distribution (bottom).





- Give the sort/merge program the correct information about your variable-length record sorting application. This includes, among other things, average and minimum record lengths.

By carefully designing your application from the beginning with the above-mentioned considerations in mind, you will experience improved performance for your sorting applications.

BE GENEROUS WITH MAIN STORAGE

In general, the more (virtual) main storage you make available to sort/merge (up to a certain limit), the better the performance. For the sort/merge program to be efficient, at least 72K bytes of main storage should normally be used, but to obtain best performance always try to allocate between 128K bytes and 512K bytes of main storage, depending on file size. However, the amount of virtual storage should be related to the amount of real storage available to the sort/merge program. As a guideline, use the total real storage available for page frames divided by the usual number of initiators in the system.

The relationship between SIZE=MAX, MAXLIM, and MINLIM (all specified at sort/merge generation time), SIZE (a PARM field operand), and the REGION field of the EXEC statement, might be described as a series of checks and balances.

The most efficient way to allocate main storage is to specify SIZE=MAX at sort/merge generation time. However, problems can arise if SIZE=MAX is used in a very large virtual region or partition, since the sort/merge program will attempt to use all the available address space. This is likely to result in excessive paging and may even cause program deactivation. To prevent this problem, an upper limit (MAXLIM) should have been set when the program was installed.

If you specify a value for SIZE (EXEC-initiated), it will override SIZE=value, provided the value does not exceed that specified for MAXLIM at installation.

If the SIZE value (EXEC-initiated) you have specified is less than the value specified for MINLIM, MINLIM will be used.

If, on the other hand, the MINLIM value is greater than that specified for REGION, sort/merge will attempt to use the value specified for MINLIM; if it fails to get the amount specified by MINLIM, sort/merge will still try to execute, provided at least 54K bytes are available for sorting purposes.

Changing the main storage allocation on the EXEC statement can improve system efficiency: By reducing the amount of main storage allocated, you impair performance of the sort/merge program in order to allow other programs to have the storage they need to operate simultaneously; and by increasing the allocation, you can run large sort/merge applications efficiently at the expense of other jobs sharing the multiprogramming environment.

The minimum amount of main storage required depends partly on the size of the buffers needed. Thus a program with large input blocks, or records, will need more main storage than one with small ones. Also, an increase in the number of intermediate storage devices will increase the minimum amount of main storage required.

A formula for calculating region size is given in Section 3 under "Main Storage."

SORTING TECHNIQUES

Depending on whether disk or tape devices are used as intermediate storage devices, the sort/merge program selects and executes different sorting techniques. Whenever possible, disk sorting techniques should be available to the sort/merge program, since tape techniques are seldom as efficient.

Note: The Blockset techniques may require more intermediate work space than Peerage or Vale. See "Efficient Use of Work Storage Devices" for more information.

DISK SORTING TECHNIQUES

There are four standard disk sorting techniques available to the sort/merge program:

- FLR-Blockset—fixed-length records
- VLR-Blockset—variable-length records
- Peerage—fixed-length records
- Vale—both fixed- and variable-length records

Sort/merge will select one of the Blockset techniques if all the conditions for its use are met (see "Conditions for Use of Blockset Sorting Techniques").

Disk Sorting Techniques for Fixed-Length Records

The sort/merge program's most efficient fixed-length record technique, FLR-Blockset, will be used for most sorting applications if the conditions listed in "Conditions for Use of Blockset Techniques" are met. If one or more of the conditions for the FLR-Blockset technique are not met, the Peerage or Vale technique will be used, where possible.

Disk Sorting Techniques for Variable-Length Records

The high-performance VLR-Blockset technique will be used for sorting variable-length records if all of the requirements listed in the following section are fulfilled. If not, the current variable-length disk sorting technique, Vale, will generally be used.

To enable sort/merge to attempt to select the best technique, whether VLR-Blockset or Vale, the following guidelines may be useful: If the average length of variable records is more than 350 bytes, you should specify the L5 operand on the RECORD control statement. If you specify an L5 operand that is between 350 and 1,000 bytes, sort/merge uses the Vale technique when the ratio of region size to number of records is large. When L5 is greater than 1,000 bytes, Vale is generally used. If the working storage is less than 100K bytes, sort/merge will attempt to select VLR-Blockset regardless of average record length. If you don't specify L5, sort/merge will try to use VLR-Blockset.

When used, the new VLR-Blockset technique will generally show processing time improvement over Vale.

CONDITIONS FOR USE OF BLOCKSET SORTING TECHNIQUES

The sort/merge program has two high-performance disk sorting techniques, FLR-Blockset and VLR-Blockset, for fixed- and variable-length records, respectively. The program will first attempt to use one of these techniques, providing the following conditions are fulfilled. If they are not, one of the other standard disk sorting techniques, Peerage or Vale, may be used where possible (Peerage or Vale for fixed-length records; Vale for variable-length records).

The first list below includes conditions common to both techniques. The second list includes conditions relevant to FLR-Blockset only, and the third, to VLR-Blockset only.

Conditions Common to Both Blockset Techniques

- More than about 64K bytes of main storage plus additional storage for buffers are available for sort and other possible modules in the region/partition. The larger the input/output block sizes are, the larger main storage must be.
- No program exits other than E15 and/or E35 (without overlay structures) provided they are prelink-edited.
- If a SORTCNTL DD statement is used, no control statements other than OPTION should be included.
- Tape work data set is not specified.
- Under MVS, up to 26 dynamically allocated sort work data sets may be used, depending on the complexity of the control field and use of SMF.
- Input or output is not a VSAM or an ASCII data set, or track overflow record format (RECFM=FT).

- Input is not a direct-access data set with key sequenced organization (BDAM).

- Input or output must not be a spool or dummy data set.

- Output cannot be padded or truncated records, or an old data set residing on tape.

- Multivolume disk data output is not requested.

- Checkpoint is not specified.

- Control fields do not exceed 248 bytes.

- Control fields that do not cause the intermediate record to expand by more than 30% of the total record length. Factors that might expand the record are overlapping fields, decimal fields, fields that require translation, or specification of EQUALS.

- All supported control field formats except those with leading, trailing, overpunched, or separate signs, or ASCII format control fields.

- Skipping of input records is not requested.

FLR-Blockset Conditions

- SORTIN record length plus 13 bytes and any additional bytes caused by control field expansion must not exceed the smallest SORTWK track capacity or 32K bytes, whichever is smaller.

- Record length is not to be changed by program exits E15 and/or E35.

- SORTWK data sets must be allocated in cylinders (MVS only).

VLR-Blockset Conditions

- VLR-Blockset minimum storage requirements are defined by the following computations (whichever results in the larger value should be used, but in no case should less than 69K bytes be used).

  In computing the amount of storage necessary to execute VLR-Blockset, use whichever one of the following computations that results in the largest value:

  1. 48K bytes of main storage **plus** the largest of three times:

     a. The maximum input block size, or

     b. The maximum output block size, or

     c. 2000 bytes.

  2. 48K bytes of main storage **plus** four times the size of the maximum record length, **plus** the largest of the following:

     a. The maximum input block size, or

     b. The maximum output block size, or

     c. 2000 bytes.

- Maximum record length does not exceed the track length for the SORTIN or SORTOUT disk data set, or 32000 bytes, whichever is smaller.

- Input or output is not spanned, variable-length records.
- Input or output is not Format D records (variable-length ASCII tape records).
- Work data sets are specified (a sort in main storage is not supported).
- The sort/merge program is not dynamically invoked by IMS/VS for variable-length record sorting applications.
- The control field does not overlap the record descriptor word (RDW).
- If the ratio of region size to the number of input records is large, and if the L5 operand specified on the RECORD control statement is greater than 350 bytes, sort/merge may, in some cases, choose to use the Vale technique. If L5 is not specified, sort/merge will execute VLR-Blockset if all other conditions are met.

BYPASSING THE BLOCKSET TECHNIQUES

You have several ways to bypass the FLR-Blockset or VLR-Blockset techniques.

- The BLKSET=NO specification on the ICEMAC installation macro will result in FLR-Blockset being bypassed; Peerage or Vale will then be the default technique used for fixed-length record sorting applications.
- The VBLKSET=NO specification on the ICEMAC installation macro will result in VLR-Blockset being bypassed; Vale will then be the default technique used for variable-length record sorting applications.

  Note: The BLKSET/VBLKSET installation defaults can be overridden by the BLKSET/NOBLKSET parameter specification on the OPTION control statement at execution time.
- Through the DEBUG control statement, you can force other techniques instead of the default Blockset techniques (for example, Peerage, Vale, BALN, or CRCX).

PEERAGE, VALE, AND CONVENTIONAL DISK SORTING TECHNIQUES

If the conditions for use of the Blockset sorting techniques are not met, sort/merge will attempt to use Peerage or Vale. Peerage is normally used if the following criteria are met:

Fixed-length records
Record length no greater than track length
No exits to be activated other than E15, E18, E35, E39, or E61
Control word not too long[1]

[1] No figure can be given for how long the control word can be if the Peerage technique is to be used; it depends on many variables, such as device type for work storage and amount of main storage available for buffers. However, the length limit is unlikely to be reached before 256 bytes, and will usually be considerably higher.

If any one of the conditions mentioned above is not satisfied, sort/merge will attempt to use Vale.

You normally need not be aware that these various standard disk techniques exist. However, you can specify either at installation time or at execution time (using the OPTION or DEBUG statement) that a Blockset technique should not be used (see "Bypassing the Blockset Techniques").

An informational message (ICE092I or ICE093I) states which of the standard disk techniques has been used.

The conventional disk sorts supplied with sort/merge (BALN and CRCX) can be forced by a parameter of the DEBUG statement. Care should be taken that the SORTWK requirements for the forced techniques have been met.

EFFICIENT USE OF WORK STORAGE DEVICES

Performance is enhanced when multiple channels are available. Performance is also improved if the device is connected so that two channel paths exist between each device and the central processing unit that is running the program.

The following table shows the relationship of file size and sorting technique to the number of cylinders used by work data sets. The numbers given are estimates of the number of SORTWK cylinders sort will use for a particular file size when secondary allocation is allowed. You can make primary and secondary allocations by means of the SORTWK DD statement or job control language (SPACE=). Automatic secondary allocation can be specified at installation time. However, even if you don't allow for secondary allocation and you allocate fewer cylinders than indicated in the table, the sorting technique may still run—but performance will generally be degraded.

SORTWK Cylinders Used[1]				
File Size in Bytes	Fixed		Variable	
	Peerage	Blockset	Vale	Blockset
500K	1	3	2	2
800K	2	3	2	2
1M	2	4	3	3
2M	4	7	5	7
4M	8	14	9	12
6M	11	19	14	19
8M	15	24	20	24
12M	18	36	27	34

[1] This example is based on jobs run with a SIZE parameter of 200,000 bytes and one SORTWK data set on a 3350.

DIRECT ACCESS WORK STORAGE DEVICES

Program performance is improved if you use devices, storage areas, and channels efficiently. If you specify a particular device type with the UNIT parameter on the DD statements that define intermediate storage data sets (for example, UNIT=3330), sort/merge assigns areas, and some optimization occurs automatically. But best performance is achieved if you follow these recommendations:

- If you can, assign only one data set per spindle.
- Try to use the same device type as far as possible.
- Use two channel paths to devices whenever you can.
- All data sets should be the same size, as nearly as possible.

- Assign SORTIN, SORTOUT, and SORTWK on different spindles and separate channels.

- Some improvement may be gained by specifying contiguous space for work data sets, and by making sure that there is enough primary space so that the automatic secondary allocation will not be needed.

Elapsed time is decreased when the sort/merge program can read input while writing to SORTWK, and write output while reading from SORTWK. If, for example, you have two channels, the best allocation of them is to have SORTIN and SORTOUT on one and the SORTWKs on the other.

Notes:

1. See Figure 6 in Section 3 for formulas used to calculate storage requirements when using different disk techniques.

2. See Appendix F for tables that show estimated total execution times for some sorting applications.

TAPE WORK STORAGE DEVICES

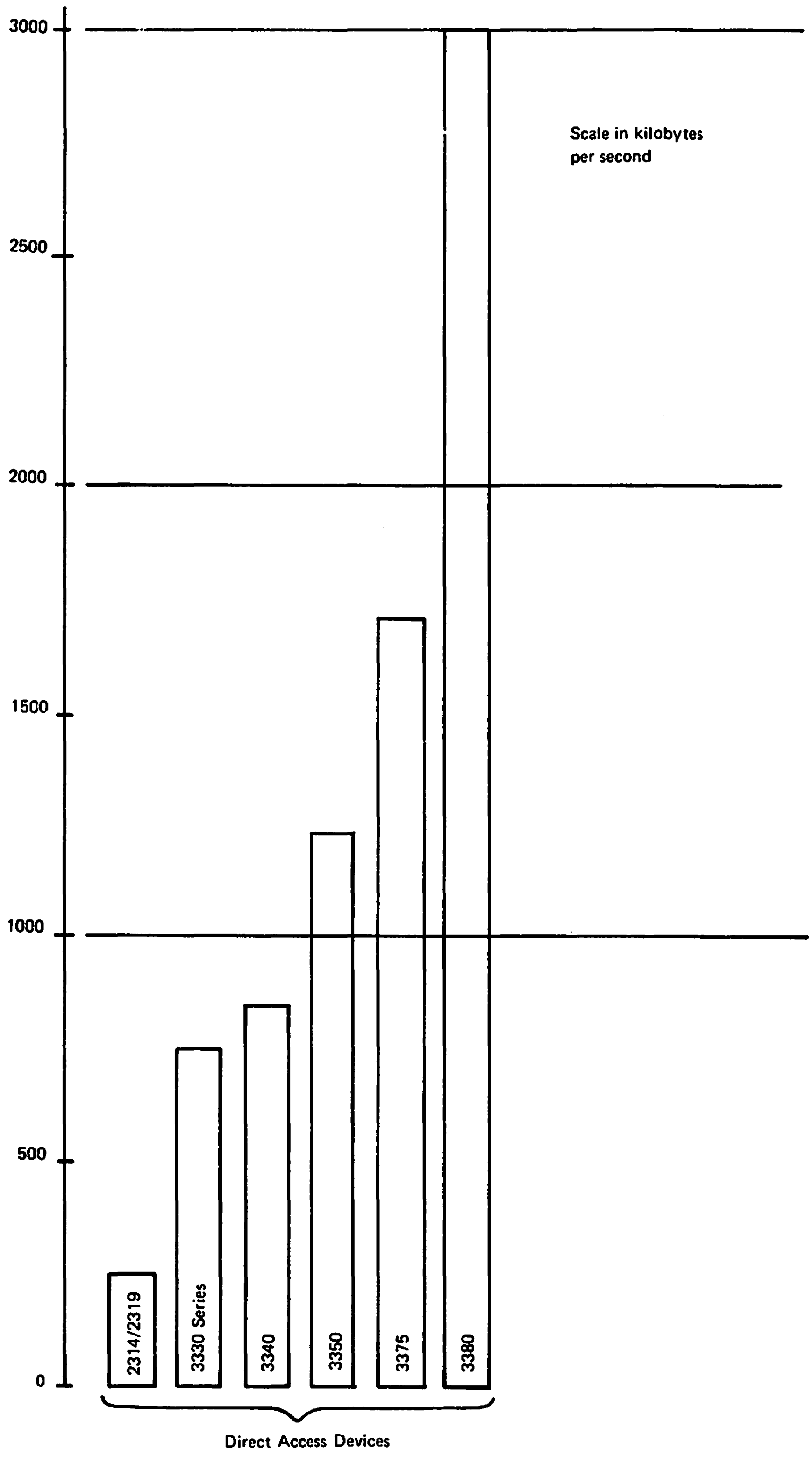
Best performance, using tape intermediate storage, is normally obtained when you use six or more tape drives of the fastest type. As a general rule, you should use as many tapes as you have available for intermediate storage. A larger number of tapes increases the number of strings that can be merged in one pass, and, therefore, decreases the number of passes required in the intermediate merge phase. This, in turn, reduces elapsed time and often the number of I/O operations.

However, increasing the number of work units also has the effect of reducing the block size used for intermediate storage; this could become a critical factor if you have relatively little main storage available for buffers. For example, if the sort/merge program has only 54K bytes in which to operate, you will probably achieve no improvement (and may find deterioration) if you use more than four tape work units. The general rule—to use as many tapes as you can—should, therefore, be taken to apply with more than, say, 100K bytes available for sort/merge.

Note: See Figure 5 in Section 3 for information on how to calculate storage requirements when using different tape techniques.

DEVICE DATA TRANSFER RATE

In general, the faster the data transfer rate of the storage device, the faster the sort. Figure 20 and Figure 21 should therefore be taken into consideration when planning for your sorting applications.



Note: The data transfer rate of any processor is limited by the speed of the channel to which it is attached.
The 3880 Model 2 or 3 with the Speed Matching Buffer Feature permits attachment of the 3380 to systems with block multiplexor channels with data rates less than 3 megabytes per second.

Figure 20. Comparative Data Transfer Rates of Disk Work Storage Devices

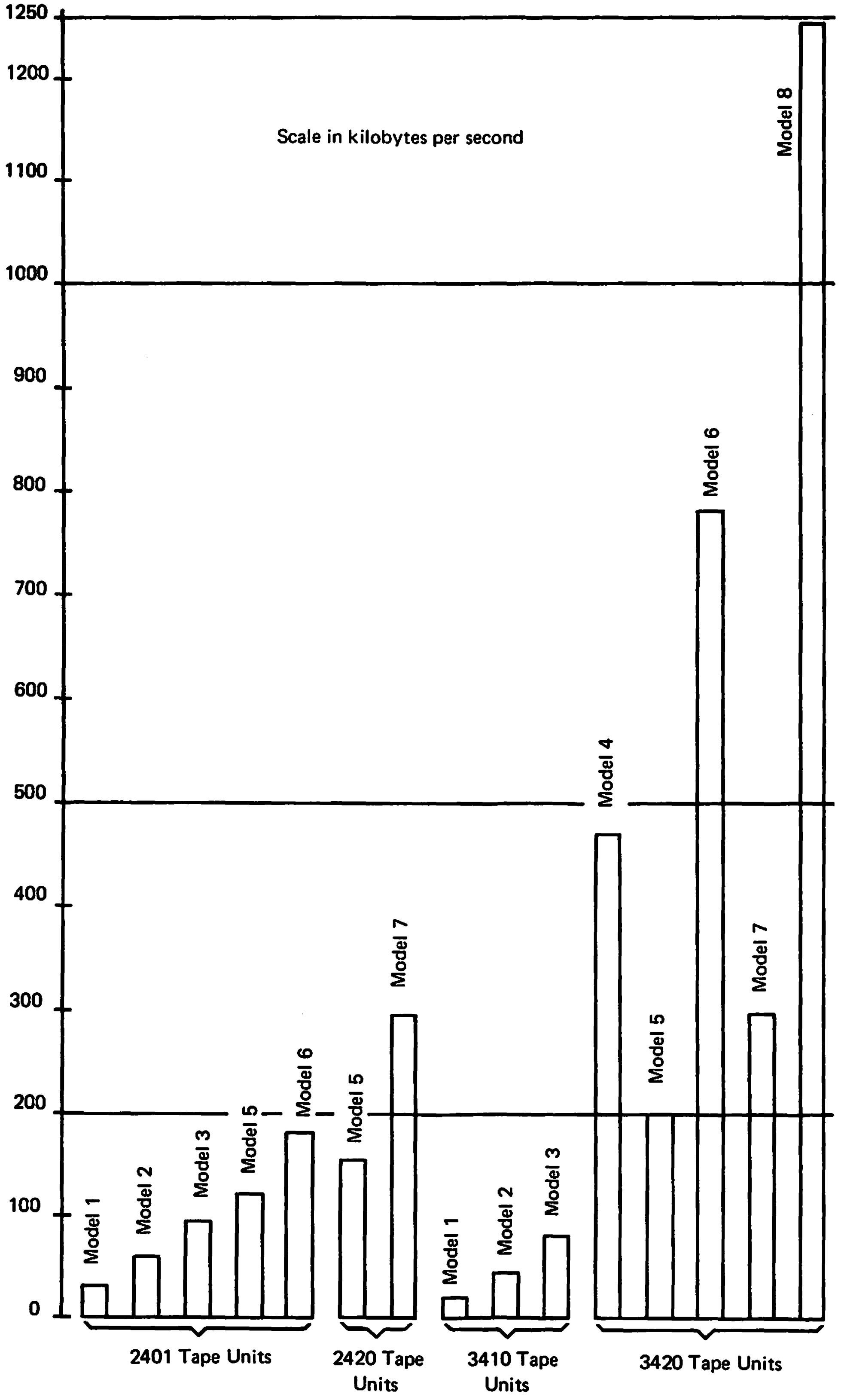


Figure 21. Comparative Data Transfer Rates of Tape Work Storage Devices

CORRECT SPECIFICATION OF INPUT/OUTPUT DATA SET CHARACTERISTICS

The sort/merge program uses the information given it about the operation it is to perform to optimize for highest efficiency. When you do not supply information such as data set size and record format, the program makes assumptions which, if incorrect, lead to inefficiency. Incorrect information can also lead to inefficiency or program termination.

SIMPLIFY CONTROL FIELD DESCRIPTIONS

When designing record formats, plan for sorting and merging the records efficiently. For example, specify the location and data formats of control fields such that they contain EBCDIC character or binary data (beginning and ending on byte boundaries) whenever possible—this decreases processor time. Fixed, packed, or zoned decimal data can be sorted as if it were binary if you know it will always be positive; and two or more contiguous character or binary fields may be sorted as one, provided they are in order of significance (with the most important first), and provided they are to be sorted in the same order.

DATA SET SIZE

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. This information is also important when dynamic allocation of the work files is requested (MVS only).

If you know the exact number of records to be sorted, use that number as the value of the FILSZ parameter in the SORT control statement. If you do not know the exact number, estimate it as closely as you can.

If you are using a tape sort, the most important information you can give the program is an accurate data set size in the FILSZ parameter of the SORT statement.

VARIABLE-LENGTH RECORDS

When the input data set consists of variable-length records, the maximum, minimum, and average record lengths should be specified correctly in the RECORD statement. This further enables the program to choose the best sort or merge technique.

Care should be taken to ensure that the LRECL parameter of the DCB corresponds to the actual maximum record length contained in your data set.

SPARE THE LINKAGE EDITOR

To save execution time, you should design your own routines so that they do not require link-editing each time they are used in a sort/merge application.

To avoid link-editing each time sort/merge executes, the following requirements must be met:

- Each routine must be a load module in a partitioned data set (library). The parameter S on the MODS statement that defines the routine must be the same as the name of the DD statement that defines the library.

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

- Each routine must have only one entry point, which is the name of the exit being used (E11, E15, etc.).
- The routines cannot have external references.
- All routines must be in the same library, or must be in concatenated data sets defined 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 link-edited and does not require further link-editing (see Figure 7 in Section 4).

If you use routines at program exits (E11, E21, or E31) that do not meet the requirements for bypassing the linkage editor, you can still save execution time by designing them for separate link-editing. To be eligible for separate link-editing, your 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 (for example, E11).
- The routine must be designed so that it can be overlaid after use.

To indicate that the routine is eligible for separate link-editing, code the parameter S for that routine on the MODS statement (see Figure 7 in Section 4).

If your routine opens data sets or communicates with running component routines, it will contain external references and, therefore, cannot be link-edited separately.

When your routine cannot bypass the linkage editor or be link-edited separately, code I (or do not code a fourth parameter) for that routine on the MODS statement. The routine is then link-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 link-editing, routines that can be link-edited separately, and routines that must be link-edited together.

TAPE SORTING TECHNIQUES

There are three standard tape sorting techniques available to the sort/merge program:

- Balanced (BALN)
- Polyphase (POLY)
- Oscillating (OSCL)

See Figure 5 in Section 3 for information on how to calculate storage requirements when using different tape techniques.

You should 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 not usually as efficient.

Forcing a Technique

If you believe that the sort/merge program is not choosing the most efficient tape technique for a particular application, you can request it to use another tape technique. It will comply if you provide enough main storage and work areas to meet the technique's requirements (see Figure 5 in Section 3). If the requirements are not met, the program will use another technique rather than terminate the program.

Refer to the discussion of the EXEC statement PARM field in Section 5 for information on how to force a technique for a tape sort.

APPENDIX A. WHAT TO DO IF THE PROGRAM STOPS

This appendix is intended to help you if sort/merge behaves in an unexpected way and you want to localize the problem and, if possible, solve or bypass it.

The first section describes how to localize a problem. The second describes various uses of the DEBUG control statement.

LOCALIZING A PROBLEM

If the sort/merge program is unable to successfully complete sorting or merging, you will get one or more program messages, and possibly also an ABEND code.

Appendix C gives you explanations of the various program messages, and suggestions as to how to cope with them. It is assumed that you have exhausted those explanations before turning to this section.

IS THIS A PROGRAM ERROR?

Your first task is to decide whether or not the problem is caused by an error in sort/merge code.

If your installation has just installed a new release or PTF level of sort/merge, it is worth checking that any necessary additional alias names have been added to module ICEMAN. If they have not, mixed levels of program modules can be executed, which can give rise to unpredictable abnormal terminations.

Otherwise, if sort/merge is run alone in its region, problems are unlikely to arise from the environment. If no routines of yours were invoking sort/merge, or being used at program exits, you can, therefore, work on the assumption that you have found a program error, and turn to "Bypassing the Problem."

However, if you are invoking sort/merge from a program of your own, or if you are using routines at program exits, you will need to eliminate your own programs as sources of error. In the example in Figure 22, for instance, one exit is used: E15.

```
ICE000I ---- CONTROL STATEMENTS/MESSAGES ---- 5740-SM1 REL 5.0 ...

           SORT FIELDS=(1,5,CH,A),EQUALS
           RECORD TYPE=F,LENGTH=(1200,,1000)
           MODS E15=(E15,79000,MODSLIB,N)
ICE074I - RECORD LENGTH L1 OR L3 OVERRIDDEN
ICE088I - SORTJOB.SORTSTEP, INPUT LRECL=1200, BLKSIZE=12000, TYPE=F
ICE093I - MAIN STORAGE = (MAX,524288,48528), NMAX=7300, BLOCKSET
ICE039A - INSUFFICIENT MAIN STORAGE - ADD 6K BYTES
```

Figure 22. A Sample Set of Messages

POTENTIAL PROBLEMS WITH ROUTINES AT PROGRAM EXITS

Use of Registers

The first thing to check with your routines is that they observe the standard linkage conventions. If they change Register 12, for example, results are unpredictable but almost certain to result in an ABEND of some kind.

Check, too, that you are not using registers for loading or storing that are accidentally causing overlay of sort code or work areas. If this happens, sort/merge could work without errors with one technique, but fail with another.

Space

The next thing to check is whether your routines are trying to use more space than you have allocated to them. Have you installed a new operating system release since the last time you used these routines? Each time you use an OPEN macro, for example, your program takes buffer space; but the amount it tries to take will depend upon such factors as the current release of the operating system.

A change of operating system could, therefore, lead to an ABEND in your own routine; or it could lead to too little space being left for sort/merge.

You can see whether too little space was left for sorting by studying the information in message ICE093I (see Figure 22). The second value following "MAIN STORAGE," 524288, shows the defaulted value taken from the installation option MAXLIM. The third value, 48528, tells you how much was actually left for sort/merge after your own routines have taken what they needed, in a region or partition of only 128K bytes.

Similar situations can occur if sort/merge is dynamically invoked using the MAX option, and a fairly large reserved value is passed to sort/merge or taken by default. Another problem could arise if the E15 routine issues a GETMAIN without a corresponding FREEMAIN at the end. This can be done indirectly, for example by leaving a data set open so that a buffer pool remains reserved.

Record Contents

If the output records do not appear to contain the same data as the input records, and either E15 or E35 has been used, check that your routine is handling register 1 correctly; especially, check that it is correct on return to sort/merge.

If, for example, you first load register 1 and then restore all registers (including register 1), it will probably have the wrong contents.

Equally, if you first restore all registers and then try to load register 1 from a changed base register, you will almost certainly pass the wrong information to sort/merge.

POTENTIAL PROBLEMS WITH INVOKING PROGRAMS

Space can also be a problem when you invoke sort/merge from another program, especially if you are using SIZE=MAX and invoking exit E15 or E35 (or, from COBOL, using an Input or Output procedure).

If you do this, and particularly if you open a file in your exit routine, check that you specify a sufficiently large amount of reserved storage.

BYPASSING THE PROBLEM

The simplest way of bypassing a problem in the sort/merge program is to force it to use a different technique.

Message ICE092I or ICE093I will tell you which sorting technique has been used, as shown in Figure 20.

You can use the DEBUG control statement, described below, to force the use or nonuse of a specific technique. Alternatively, if the problem is with either of the Blockset techniques, you can use the NOBLKSET parameter on the OPTION control statement to bypass the Blockset techniques.

DEBUG CONTROL STATEMENT

This statement is only valid when the program meets the criteria for the standard disk techniques. If it is supplied under other circumstances, it is ignored.

The statement is not intended for regular use; only the first two parameters are of general interest. The other parameters can be used to provide a temporary bypass, or to supply detailed information on program execution for use when optimizing or debugging the standard disk sort.

DEBUG can be passed to an invoked sort by means of the SORTCNTL DD statement, for example:

```
//SORTCNTL DD           *
      DEBUG PEERVALE
```

Note that the DD name might not always be SORTCNTL, because the first four letters of SORT special DD statement names can be changed for an invoked application. It might, for example, need to be called //TESTCNTL instead. See Section 7 on invoking sort/merge from another program.

If a DEBUG statement is included in a SORTCNTL data set, the Blockset techniques will not be used.

```
[label]   DEBUG  [ABEND|NOABEND]
                 [,DUMP|,NODUMP]
                 [,PEERVALE|,BALN|,CRCX]
                 [,BSAM]
                 [,CLOCK]
                 [,FLAG]
                 [,CTRx]
```

ABEND|NOABEND   Overrides the generated default for action to be taken when the program encounters an uncorrectable error, as described under "DEBUG Control Statement" in Section 4.

DUMP|NODUMP   Recognized but ignored.

In addition to these parameters, other parameters can be used to provide a temporary bypass, or to supply detailed information on program execution for use when optimizing or debugging the standard disk sort. The parameters and their uses are:

PEERVALE   With a disk sort, one of the standard techniques (FLR-Blockset, VLR-Blockset, Peerage, or Vale) is normally used. If you have encountered a problem when using one of the Blockset techniques (see message ICE092I or ICE093I), you can temporarily bypass this technique by specifying PEERVALE.

BALN|CRCX

With a disk sort, you can use this parameter to force either the balanced (BALN) or crisscross (CRCX) disk sorting technique and, therefore, bypass the standard disk sort technique used by the program. If either BALN or CRCX is forced, then the following restrictions apply:

- At least three work data sets on the same type of device are needed, with amount as specified in Figure 5. Mixed device types are not allowed.
- Maximum record length must be less than work device track length.
- Allocation must be contiguous (the CONTIG parameter is required), and only primary extents will be used.
- Six or more work data sets are required for the CRCX technique.
- For SORTWKnn: nn can be any number from 01 to 32. The first number must be 01 and the others must follow consecutively with no gaps.
- Unused work space will not be released; the RLSE parameter must not be specified.

BSAM

With the disk sort techniques Peerage and Vale, the EXCP access method is normally used for SORTIN and SORTOUT. If you encounter a problem related to this I/O activity, you can temporarily bypass it by specifying BSAM.

CLOCK

(Only for Peerage and Vale.) Instructs the program to measure elapsed and processor times for the different phases, and to produce the appropriate messages if FLAG is also specified.

FLAG(@)

(Only for Peerage and Vale.) Instructs the program to print information messages (ICE120-125). These messages are listed under "Messages Produced by Using the DEBUG Statement." To get the times printed you also need to specify CLOCK.

CTRx=value

Specifying this parameter will force Peerage or Vale to be used. The program will keep a count of the input or output records. When the count reaches the value specified, the program will ABEND and a formatted dump will be printed.

The numbers that may be assigned to x are:

2—Count of input records being moved from the input buffer.

3—Count of output records being moved to the output buffer.

4—Count of input records inserted by E15.

5—Count of output records deleted by E35.

Note: When the count reaches 'value', the program will ABEND. It will also terminate with message ICE025A if the 'value' is a number greater than the number of input records.

MESSAGES PRODUCED BY USING THE DEBUG CONTROL STATEMENT

Messages ICE120-125 are issued if the DEBUG statement is supplied with the appropriate parameter FLAG(@) (only for Peerage and Vale sorts).

ICE120O RL=a B=b IL=c IS=d IB=e RM=f EM=g BA=h IX=j OX=k

This message relates to the optimization part of Initialization Phase 0.

RL is the record length (within the sort);
B is the blocking factor used for work areas;
IL is the number of physical index blocks per logical index block;
IS is index entry size;
IB is the number of indexes/physical index block;
RM is the maximum number of strings to be merged in one pass of Phase 2;
EM is the maximum number of strings to be merged in Phase 3;
BA is the base bin size;
IX is the number of input buffers;
OX is the number of final output buffers.

ICE121C ET=a CT=b BN=c X=d TO=e SN=f G=g

This message relates to Sort (Input) Phase 1.

ET is the elapsed time taken in centiseconds;
CT is the processor time in centiseconds;
BN is the number of blocks handled;
X is the number of EXCPs issued;
TO is the number of tracks put out;
SN is the number of strings produced;
G is the number of records in the record storage area.

ICE122R ET=a CT=b BN=c X=d {G|RM}=e PN=f BT=g TO=h

This message relates to Intermediate Merge Phase 2.

ET is the elapsed time taken in centiseconds;
CT is the processor time in centiseconds;
BN is the number of work data set blocks handled;
X is the number of EXCPs issued;
G is the number of records in the record storage area;
RM is the maximum number of strings to be merged in one pass of Phase 2;
PN is the highest partition number;
BT is the number of tracks handled more than once.
TO is the number of tracks put out;

ICE123E ET=a CT=b BN=c X=d {G|EM}=e TO=f BT=g

This message relates to Output (Final Merge) Phase 3.

ET is the elapsed time taken in centiseconds;
CT is the processor time in centiseconds;
BN is the number of work data set blocks handled;
X is the number of EXCPs issued;
G is the number of records in the record storage area;
EM is the maximum number of strings to be merged in Phase 3;
TO is the number of tracks put out;
BT is the number of tracks handled more than once.

ICE124P ET=a CT=b PE=c RP=d CX=e CO=f C0=g CR=h G=i WB=j

This message relates to Intermediate Merge Phase 2.

ET is the elapsed time taken in centiseconds;
CT is the processor time in centiseconds;
PE is the 'peerage': the number of logical strings obtained by logically rearranging the tracks of physical strings;
RP is the number of partitions;
CX is the number of exempt blocks;
CO is the number of overflow blocks;
C0 is the number of blocks in partition 0;
CR is the number of blocks to be handled by partition 0;
G is the number of records in the record storage area;
WB is the number of blocks written back to work storage.

ICE125O CT=a GP=b SA=e X=d

This message relates to work I/O and is cumulative: it appears after each of Phases 1-3 and shows cumulative totals each time.

CT is the processor time in centiseconds;
GP is the number of work I/O blocks;
SA is the number of standalone seeks;
X is the number of EXCPs issued.

MESSAGES PRODUCED BY USING THE DIAG OPTION

Diagnostic messages are obtained when you specify the DIAG option in the PARM field of the EXEC job control statement. This option is only available for tape techniques, a merge-only application, or when forcing a nonstandard disk technique.

The DIAG option and its specifications are described under "'PARM' Field Options" in Section 5. Remember that the DIAG option impairs program performance, and should be removed as soon as it is no longer needed.

The diagnostic messages are as follows:

ICE900I GENERATED CORE END ADDRxx

ICE901I INPUT BFR TBL ADDRxxxx

ICE902I OUTPUT BFR ADDR xxxx,xxxx

ICE903I RSA TBL ADDR xxxx

ICE904I TREE ADR FROM xxxx TO xxxx

ICE905I MOVE RTN ADDR xxxx

ICE906I DCB TBL ADDR xxxx

ICE907I O/P CCW ADDR xxxx

ICE908I OUTPUT IOB ADDR xxxx

ICE909I OPEN LIST ADDR xxxx

ICE920I GENERATED CORE END ADDR xxxx

ICE921I INPUT BFR TBL ADDR xxxx

ICE922I OUTPUT BFR ADDR xxxx,xxxx

ICE923I MOVE RTN ADDR xxxx

ICE924I DCB TBL ADDR

ICE925I O/P CCW ADDR xxxx

ICE926I IOB TBL ADDR xxxx

ICE927I I/P CCW ADDR xxxx

ICE940I GENERATED CORE END ADDR

ICE941I INPUT BFR TBL ADDR xxxx

ICE942I OUTPUT BFR ADDR xxxx,xxxx

ICE943I MOVE RTN ADDR xxxx

ICE944I ECB TBL ADDR xxxx

ICE945I I/P CCW ADDR xxxx

ICE961I TECHNIQUE xxxx

ICE962I NO/SIZE OF BFRS, PH x, x, xxxx

ICE963I MAX.SYSGEN CORE xxxx

ICE964 CALC. CORE PH1=xxxx

ICE965I MERGE ORDER=xxxx

ICE988I ICEyyy LOC. AT xxxx[1]

ICE989I CLOCK - xx,xx,xx[2]

ICE990I NO OF STRINGS PROD BY PH1 xxxxxxx

[1] Appears frequently; provides the starting addresses of the program modules.

[2] Appears at the beginning of each phase (except Phase 0), and at the end of the program.

DUMPS

There are two types of failure that can cause dumps.

- Sort-program-detected uncorrectable errors which give critical error messages.

- Sort program failures that are detected by the operating system.

NORMAL ABEND DUMPS

The default ERETINV|ERETJCL=ABEND|RC16, which was set at sort/merge installation time, can be overridden in a standard disk technique sort by the DEBUG control statement (see Section 4, "DEBUG Control Statement") or, in other cases, by the PARM field option DIAG (see Section 5, "'PARM' Field Options"). To obtain a normal ABEND dump you must provide a SYSUDUMP, SYSMDUMP, or SYSABEND DD statement.

FORCING A SPECIALLY FORMATTED DUMP (ONLY FOR PEERAGE AND VALE)

The default ERETJCL|ERETINV=ABEND|RC16, which was set at sort/merge installation time, can be overridden in a standard disk technique sort by the DEBUG control statement (see Section 4, "DEBUG Control Statement").

To obtain a specially formatted dump for a sort, the CTRx=value must be specified in the DEBUG statement. This first prints a SNAP dump (corresponding to a normal SYSUDUMP dump), followed by formatted information as shown in Figure 23.

1 SYSTEM DUMP
SNAP dump corresponding to a normal SYSUDUMP dump.

2 FORMATTED DATA

2.1 Save areas
The standard save areas used by different levels of the program.

2.2 ABEND code
A fullword with the format X'xxsssuuu', where
xx is the standard ABEND code prefix,
sss is the system completion code at program failure (or zeros), and
uuu is the user completion code at uncorrectable error (or zeros).
This code is equal to zero for definition errors, and equal to the message number for other errors (for example, '046' would represent message ICE046A).

2.3 A fullword containing the address of the instruction at which failure occurred.

2.4 Register contents when program failure occurred: 16 fullwords giving the register contents in the order 0-15.

2.5 Contents of ICECOMMA (sort variables) formatted when program failure occurred, with offsets from Register 13, comments, labels, and definitions.

2.6 Trace of important events, in the form

x yyy

where:
x identifies the part of the program
yyy identifies the segment of code entered
x can be one of the following codes:

DEF - definition (ICEDEF)
C - creation (ICECRE, ICEVRE, ICEVRN)
P - partitioning (ICEPAR)
R - reduction (ICERED, ICEVED)
E - elimination (ICELIM, ICEVIM, ICELIV, ICEVIP)
A - appendage (for PCI, channel end, or end-of-extent)

The first event listed is the most recent;*
the last is the first that occurred (normally, DEF ENTRY).

*If one of the most recent events listed concerns an exit, the probable cause of program failure is a programming error in the exit routine.

Figure 23. Contents of a Specially Formatted Dump

Displacement (in hex) from the start of ICECOMMA

Comment from the source code

The data definition level: a 'level 3' area is always a subset of the preceding 'level 2' area, and so on.

Label from the source code

One of the standard PL/S data attributes, for example, PTR(31), meaning a fullword pointer

Content of the area when the dump was taken

```
DISPL. COMMENT                                   LEVEL LABEL     ATTR            VALUES

 0000  /* SUPERVISOR AND DM SAVE AREA*/          2     CSAVEOS   PTR(31)         00E2D4F1
 0004                                                                            000C4FB0
 0008                                                                            000C91F8
 000C                                                                       1    700C4E7A
 0010                                                                            000C632C
 0014                                                                            000D0000
 0018                                                                            000C9590
 ====================================================================================
 00D0                                                                            000E2478
 00D4                                                                            A00DFEA0
       /* LEVEL 3 ROUTINE SAVE AREA  */          2     CSAVEL3                  2
 00D8  /* ABEND - ABEND CODE         */           3    *         PTR(31)         800C1000
 00DC  /* ABEND - INTERRUPT PSW END  */           3    *         PTR(31)         600CA0FC
 00E0  /* ABEND - REGISTER  0        */           3    *         PTR(31)    3    FFFFFFFC
 00E4  /* ABEND - REGISTER  1        */           3    *         PTR(31)         000000D2
 00E8  /* ABEND - REGISTER  2        */           3    *         PTR(31)         00000000
 00EC  /* ABEND - REGISTER  3        */           3    *         PTR(31)         00000008
 00F0  /* ABEND - REGISTER  4        */           3    *         PTR(31)         000D4750
 00F4  /* ABEND - REGISTER  5        */           3    *         PTR(31)         000002D4
 00F8  /* ABEND - REGISTER  6        */           3    *         PTR(31)         000D4C7E
 00FC  /* ABEND - REGISTER  7        */           3    *         PTR(31)         000E051C
 0100  /* ABEND - REGISTER  8        */           3    *         PTR(31)         00000000   4
 0104  /* ABEND - REGISTER  9        */           3    *         PTR(31)         000CA0E0
 0108  /* ABEND - REGISTER 10        */           3    *         PTR(31)         000D0D4C
 010C  /* ABEND - REGISTER 11        */           3    *         PTR(31)         000D15FE
 0110  /* ABEND - REGISTER 12        */           3    *         PTR(31)         A00D0820
 0114  /* ABEND - REGISTER 13        */           3    *         PTR(31)         000C9240
 0118  /* ABEND - REGISTER 14        */           3    *         PTR(31)         600D1002
 011C  /* ABEND - REGISTER 15        */           3    *         PTR(31)         00000000
 0120  /*  WORK AREA       */                    2     CTEMP1    FIXED(31)       000C936C
       /* WORK AREA       */                      3    CWORK1    FIXED(31)
       /*  WORK AREA       */                      4   *         CHAR(1)
       /*  WORK AREA       */                      4   CTEMP124  PTR(24)
       /*  WORK AREA       */                       5  CWORK124  PTR(24)    5
       /*  WORK AREA       */                          *         CHAR(1)
       /*  WORK AREA       */                          CTEMP115  FIXED(15)
       /*  WORK AREA       */                          CWORK116  FIXED(16)
       /*  WORK AREA       */                          *         CHAR(1)
       /*  WORK AREA       */                          CTEMP108  PTR(8)
       /*  WORK AREA       */                          CWORK108  PTR(8)
```

1 *Save areas:* The standard save areas are allocated at the beginning of ICECOMMA.

2 *ABEND CODE:* In the example the program ended with system completion code X'0C1'.

3 *Last instruction:* The address of the failed instruction, in this case X'0CA0FC'.

4 *Register contents:* Shows the register contents when the program failed.

5 *ICECOMMA:*Remaining contents of ICECOMMA are shown in the same way. For example, field CTEMP1 (also known as CWORK1) contained X'000C936C' CTEMP124, a subset of the larger area, thus contained X'0C936C'.

Figure 24. Interpreting a Formatted Dump

APPENDIX B. DATA FORMAT EXAMPLES

The format descriptions refer to the assembled data formats as used with IBM System 360/370. If, for example, a data variable is declared in PL/I as FIXED DECIMAL, it is the compiled format of the variable that must be given in the 'f' field of the SORT control statement, not the PL/I declared format. In this case, the 'f' field would be PD (packed decimal) because the PL/I compiler converts fixed decimal to packed decimal form.

Format	Description
CH	(character EBCDIC, unsigned). Each character is represented by its 8-bit EBCDIC code. Example: AB7 becomes C1 C2 F7 Hexadecimal 11000001 11000010 11110111 Binary
ZD	(zoned decimal, signed). Each digit of the decimal number is converted into its 8-bit EBCDIC representation. The sign indicator replaces the first four bits of the low order byte of the number. Example: -247 becomes 2 4 - 7 Decimal F2 F4 D7 Hexadecimal 11110010 11110100 11010111 Binary The number +247 becomes F2 F4 C7 11110010 11110100 11000111
PD	(packed decimal, signed). Each digit of the decimal number is converted into its 4-bit binary equivalent. The sign indicator is put into the rightmost four bits of the number. Example: -247 becomes 2 4 7 - Decimal 24 7D Hexadecimal 00100100 01111101 Binary The number +247 becomes 247C in hexadecimal.
FI	(fixed point, signed). The complete number is represented by its binary equivalent in either halfword or full word format. The sign indicator is placed in the most significant bit position. 0 for + or 1 for -. Negative numbers are in 2's complement form. Example: +247 becomes in halfword form 00F7 Hexadecimal 0000000011110111 Binary The number -247 becomes FF09 Hexadecimal 1111111100001001 Binary

Format	Description
BI	(binary unsigned). Any bit pattern.
FL	(floating point, signed). The specified number is in the two-part format of character and fraction with the sign indicator in bit position 0. Example: +247 becomes 0 1000010 111101110000000....... + chara. fraction -247 is identical except that the sign bit is changed to 1.
AC	(character ASCII, unsigned). This is similar to format CH but the characters are represented with ASCII code. Example: AB7 becomes 41 42 37 Hexadecimal 01000001 01000010 00110111 Binary (ASCII code)
CSL	(signed number, leading separate sign). This format refers to decimal data as punched intocards, and then assembled into EBCDIC code. Example: +247 punched in a card becomes + 2 4 7 Punched numeric data 4E F2 F4 F7 Hexadecimal 01001110 11110010 11110100 11110111 Binary EBCDIC code -247 becomes - 2 4 7 Punched numeric data 60 F2 F4 F7 Hexadecimal 01100000 11110010 11110100 11110111 Binary EBCDIC code
CST	(signed numeric, trailing separate sign). This has the same representation as the CSL format except that the sign indicator is punched after the number. Example: 247+ punched on the card becomes F2 F4 F7 4E Hexadecimal
CLO*	(signed numeric, leading overpunch sign). This format again refers to decimal data punched into cards and then assembled into EBCDIC code. The sign indicator is, however, overpunched with the first decimal digit of the number. Example: +247 with + overpunched on 2 becomes +2 4 7 Punched numeric data C2 F4 F7 Hexadecimal 11000010 11110100 11110111 Binary EBCDIC code Similarly -247 becomes D2 F4 F7
CTO*	(signed numeric, trailing overpunch sign). This format has the same representation as for the CLO format except that the sign indicator is overpunched on the of the number. Example: +247 with + overpunched on 7 becomes F2 F4 C7 hexadecimal
*The overpunched sign bit is always X'C' for positive and X'D' for negative.	

Format	Description
ASL	(signed numeric, ASCII, leading separate sign). Similar to the CSL format but with decimal data assembled into ASCII code. Example: +247 punched into card becomes + 2 4 7 Punched numeric data 2B 32 34 37 Hexadecimal 0101011 00110010 00110100 00110111 Binary ASCII code Similarly -247 becomes 2D 32 34 37 hexadecimal
AST	(signed numeric, ASCII, trailing separate sign). This gives the same bit representation as the ASL format except that the sign is punched after the number. Example: 247+ becomes 32 34 37 2B hexadecimal

A detailed description of CH, ZD, PD, FI, BI, and FL data formats can be found in the _OS/VS - DOS/VSE - VM/370 Assembler Language Manual_, Section G.

APPENDIX C. ERROR AND INFORMATION MESSAGES

MESSAGES PRODUCED BY THE PROGRAM

This section lists, explains, and suggests appropriate responses to messages produced by the sort/merge program.

The sort/merge program generates two kinds of messages:

1. Those which result from critical error conditions, and
2. Those which give information about the program's operation.

Note: Messages produced by DEBUG and DIAG appear in Appendix A.

The printing of either all or only critical messages can be specified at sort/merge generation. The messages can appear either on a printer or at the appropriate console. The only exception is ICE097I, which will appear only on the master console, and cannot be overridden with any of the message options.

The message options set up at sort/merge generation can be overridden on a job-step by job-step basis by coding the FLAG parameter in the PARM field of the EXEC statement; see Section 5.

CONTROL STATEMENT CODING ERRORS

The sort/merge program analyzes control statements in two ways:

1. The general format (syntax) of control statements.
2. The information contained in the program control statements and job control language statements, for content errors. Each statement is scanned for errors. The first error detected stops the scan for that statement. Unless the printer output (normally SYSOUT) DD statement is in error or missing and such a statement is required because diagnostic messages and/or control statements are to be printed, sort/merge prints a message and continues the scan on successive statements.

When control statements are listed, and if an error occurs which can be associated with a specific statement, the diagnostic message will follow it in the listing. If the error can be associated with a specific operation, operand, or value, a pointer ($) will be printed on the line below the statement, close to the character in error.

When all control input has been analyzed and if an error has occurred, the program terminates.

MESSAGE STATUS

Messages produced by the program are all prefixed by the letters ICE.

They are all routed to the master console (routing code 2) except for ICE061A (codes 3,4,7), which is routed to the tape, direct-access, or unit record pool to which it applies.

They all have descriptor code 6 ('job status information'), except for ICE061A, which has code 4 ('system status').

CHECKLIST

If a problem should recur, make sure BEFORE CALLING IBM FOR PROGRAMMING SUPPORT that you have available full documentation on the failing job step:

- The associated job stream and master console log
- A list of all installation options specified at sort/merge generation
- Listings of all user routines being used at program exits, and/or the program calling the sort/merge (if any)

If necessary, rerun with:

- MSGLEVEL=(1,1) in the JOB statement.
- The FLAG(I) subparameter in the PARM parameter of the EXEC statement.
- The DIAG subparameter in the PARM parameter (for a tape sort); or the DEBUG control statement with FLAG(@), CLOCK parameters (for Peerage and Vale).
- The SIZE subparameter in the PARM parameter (if applicable).
- A SYSUDUMP DD statement is sufficient unless an I/O error has occurred, in which case a SYSABEND DD statement is necessary.

Keep the input to the failing job step, in case it is necessary to reproduce the error.

BYPASS

If you need a temporary bypass, a simple method may be to change the main storage allocation (increase by at least 8K bytes); or the intermediate storage allocation (preferably, change both type of device and size and number of areas).

Another bypass could be to force another technique in the program (see DEBUG Statement in Appendix A). See also "Bypassing the Blockset Techniques" in Section 8.

MESSAGE FORMAT

Component Name	ICE	
Program Producing Message	Sort/Merge Program Product 5740-SM1.	
Audience and Where Produced	For programmer and/or operator: SYSOUT data set or console (system generation option).	
Message Format	ICEnnns	text (for messages directed to a printer).
	ICEnnns	xxxxxxxx, yyyyyyyy, text (for messages directed to a console).
	nnn	Message serial number.
	s	For messages 120-124, phase indication. For other messages, severity code: A Error message; programmer action is required. I Information message; no programmer action is required.
	xxxxxxxx	Jobname.
	yyyyyyyy	Job or procedure stepname (if any).
	text	Message text.
Comments	If a problem recurs, see "Checklist."	

ICE000I --- CONTROL STATEMENTS/MESSAGES --- 5740-SM1 REL nn PTF xx...

Explanation: This is the heading printed on each new page when control statements are listed. This message never appears on the console. nn is the release level; xx is the PTF number most recently applied. The date follows.

ICE001A TEXT BEGINS IN WRONG COLUMN

Explanation: Critical. A continuation card following a card broken at a comma does not begin within columns 2-16; or a continuation card following a card broken at column 71 (with a punch in 72) does not begin in column 16.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check continuation cards for text beginning in a wrong column.

ICE002I DUPLICATE control STATEMENT

Explanation: This message is generated if a control statement type appears more than once (for example, both SORT and MERGE statements).

System Action: The program does not analyze duplicate statements. The first one encountered is used unless the SORTCNTL DD statement is present.

Programmer Response: No action necessary. For later runs, check control statements.

ICE003A CONTINUATION CARD MISSING

Explanation: Critical. A continuation card has been indicated by the previous card ending with a comma, or with a nonblank character in column 72, and no card follows.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check for keypunching error, an overflow of parameters into column 72, or a missing continuation card.

ICE004A INVALID OPERAND DELIMITER

Explanation: Critical. An operand ends with an incorrect delimiter.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check for keypunching errors.

ICE005A STATEMENT DEFINER ERROR

Explanation: A control statement does not contain one of the seven acceptable operation definers (SORT, MERGE, OPTION, RECORD, MODS, ALTSEQ, DEBUG, or END).

System Action: Termination when all control statements scanning is complete.

Programmer Response: Check for blank cards in SYSIN. Check all statements for incorrect, misplaced, or misspelled operation definers. Check that no definer begins in column 1 (in which case it will have been treated as a label). If you have a label, check that it begins in column 1 (otherwise it will have been treated as an operation definer). If the sort is invoked, check that the byte count field of the parameter list is on halfword boundary or E15/E35 routine starts on correct boundary (not byte boundary).

ICE006A OPERAND DEFINER ERROR

Explanation: Critical. The first operand of a control statement does not begin on the same statement as the operation definer.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check for statements, other than the END statement, that contain no operands.

ICE007A SYNTAX ERROR

Explanation: Critical. A control statement contains an error in syntax.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check the control statements for syntax errors. Some of the more common syntax errors are:

- Unbalanced parenthesis
- Missing comma
- Embedded blank

ICE008A FIELD OR VALUE GT 8 CHARACTERS

Explanation: Critical. A parameter of more than 8 characters has been specified.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check control statements for parameters with more than eight characters.

ICE010A NO SORT OR MERGE CONTROL STATEMENT

Explanation: Critical. All control statements have processed and no SORT or MERGE control statement has been found.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Supply a SORT or MERGE control statement.

ICE011A DUPLICATE OR CONFLICTING OPERANDS ON THE OPTION STATEMENT

Explanation: Critical. On an OPTION control statement, one of the following errors was found:

- A keyword was specified twice.
- A keyword and a variation of it were both specified. CKPT and CHKPT are variations, as are FILSZ and SIZE.
- A keyword and its opposite were both specified. EQUALS and NOEQUALS are examples of this.

Note: The Blockset techniques accept a keyword and its opposite, and use whichever is specified last in sequence as the intended specification.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check the OPTION control statement for the errors indicated in the explanation and correct the errors.

ICE012A NO FIELD OPERAND DEFINER

Explanation: Critical. A SORT or MERGE control statement does not contain a control field definition.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check SORT or MERGE control statement for lack of a control field definition (FIELD operand).

ICE013A INVALID SORT OR MERGE STATEMENT OPERAND

Explanation: Critical. An invalid keyword operand has been detected on a SORT or MERGE control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that the SORT or MERGE control statement does not contain an invalid keyword operand. Valid keywords are FIELDS, FORMAT, FILSZ or SIZE, CKPT or CHKPT, SKIPREC and EQUALS or NOEQUALS.

ICE014A DUPLICATE SORT OR MERGE STATEMENT OPERAND

Explanation: Critical. A keyword operand is defined twice on a SORT or MERGE control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check SORT or MERGE control statement for a duplicated keyword operand. Note that FILSZ and SIZE count as the same, as do CKPT and CHKPT as well as EQUALS and NOEQUALS.

ICE015A VARIABLE RECORD TOO SHORT

Explanation: Critical. A routine has detected a variable-length record too short to contain all control fields.

System Action: The program terminates.

Programmer Response: Check the input in both the SORTIN data set and all records inserted at exit E15 to see that all records contain all control fields. Remove any which are too short. Check your E15 routine and correct any errors.

ICE016A INVALID FIELDS OPERAND VALUE

Explanation: Critical. An invalid number of values is specified with a FIELDS operand on a SORT or MERGE control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check for valid formats of the FIELDS operand:

```
FIELDS=(location,length,format
,order...)

or

FIELDS=(location,length,order...)
,FORMAT=format
```

ICE017A CONTROL FIELD DISPLACEMENT OR LENGTH VALUE ERROR

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: Termination when all control statement scanning is complete.

Programmer Response: 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 4093; and that bit positions and lengths are specified for binary fields only, and do not exceed 7.

ICE018A CONTROL FIELD ERROR

Explanation: Critical. An error in specifying the type of control field defined in a SORT or MERGE control statement has been detected.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that all control field types are specified as either CH, AQ, ZD, PD, FI, BI, FL, AC, CSL, CST, CLO, CTO, ASL, or AST.

ICE019I INADEQUATE INDICATION OF RESIDENT/NONRESIDENT MODULES

Explanation: This message is generated for one of two reasons:

- Modules are resident but indicated non-resident
- Modules are non-resident but indicated resident

System Action: None.

Programmer Response: RESDNT field in ICEAM1 should be changed. See OS/VS Sort/Merge Installation Guide.

ICE020A INVALID RECORD STATEMENT OPERAND

Explanation: Critical. An invalid keyword has been found in a RECORD control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check for valid keywords: TYPE and LENGTH.

ICE021A NO TYPE OPERAND

Explanation: Critical. A TYPE operand is required for a tape or nonstandard disk sort, and is not present (or the RECORD statement is required but missing).

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check RECORD control statement for TYPE operand.

ICE022A RECORD FORMAT NOT F, V OR D

Explanation: Critical. An error in specifying the value associated with the TYPE operand of a RECORD control statement has been detected.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check RECORD control statement for keypunching or other errors resulting in TYPE operand value being some character other than F (fixed-length records), V (variable-length records), or D (variable-length ASCII records). Check also for a conflict between the SORTIN/SORTOUT DCB RECFM parameter and the RECORD control statement.

ICE023A NO LENGTH OPERAND

Explanation: Critical. The LENGTH operand of a RECORD control statement is missing, and input record length is not otherwise available, since no DD statement with the name SORTIN has been supplied.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check for missing RECORD statement; check RECORD control statement for lack of LENGTH operand; check for missing SORTIN DD statement.

ICE024A RECORD LENGTH VALUE ERROR

Explanation: Critical. An incorrect value is associated with the LENGTH operand of a RECORD control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Some of the more common errors are:

- Entry errors in length values. (Length values must not contain nonnumeric characters, negative numbers, more than 8 characters, a nonprintable character, etc.)
- Minimum length (L4) greater than maximum length (L2) or average length (L5).
- Average length (L5) greater than maximum length (L2).
- No LENGTH specified, and logical record length not specified on the SORTIN DD statement.

ICE025A RECORD COUNT OFF

Explanation: Critical. The program has compared the count of input records and output records (shown in message ICE054I), taken into account the numbers inserted or deleted (shown in message ICE055I), if any, and found a discrepancy.

The message is issued when the whole output data set has been written. The message is suppressed if CHECK=NO was specified at installation time or NOCHECK at execution time, and you have an E35 exit and no SORTOUT DD statement.

System Action: The program terminates.

Programmer Response: The most likely cause is that you have invoked Sort from another program, have specified E35, and from your E35 routine have passed a return code of 8 (end of file) too early, when there are still output records left. If this is not the cause, examine any exit routines (especially E15, E25, and E35) for possible return code or other errors. It is possible but less likely that the error was caused by an internal sort problem.

ICE026I SMF RECORD NOT WRITTEN TO THE SMF DATA SET (RC=xx)

Explanation: Nonzero return code was returned from the SMF record exit IEFU83.

System Action: Writing of the SMF record to the SMF data set was suppressed.

Programmer Response: Determine whether or not your IEFU83 record exit is correct and the SMF facility is properly installed and initialized on your system. Correct if necessary.

ICE027A CONTROL FIELD BEYOND RECORD

Explanation: Critical. A control field has been defined as extending beyond the maximum record length.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check SORT or MERGE control statement for incorrectly specified control field displacement. Check RECORD control statement for incorrectly specified l (the maximum input record length).

ICE028A TOO MANY EXITS

Explanation: Critical. An attempt has been made to specify in the MODS statement more than the maximum number of program exits allowed by the program.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that routines are specified for valid exits only, and that each exit is associated with only one routine. Exits which may be specified in the MODS statement 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 can be specified.)

ICE029A IMPROPER EXIT

Explanation: Critical. This message is generated for one of two reasons:

- An incorrect exit has been specified on a MODS control statement.
- An exit in the sort or intermediate merge phase of the program has been specified for a merge application.

System Action: Termination when all control statement scanning is complete.

Programmer Response: 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. Numbers which may be specified 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.)

ICE030A MULTIPLY DEFINED EXITS

Explanation: Critical. A program exit has been defined twice in MODS control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check MODS statement for multiply defined exits.

ICE031A INVALID MODS OP CHAR

Explanation: Critical. An invalid character in a parameter of a MODS control statement has been found.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check the parameters of the MODS control statement for a length field containing something other than numeric data, a source or name field beginning with something other than an alphabetic character, or containing a special character other than $, @, #.

ICE032A EXIT E61 REQUIRED

Explanation: Critical. A SORT or MERGE control statement defines a control field to be modified by a 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: Termination when all control statement scanning is complete.

Programmer Response: 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.

ICE033A CONTROL FIELD SEQUENCE INDICATOR 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: Termination when all control statement scanning is complete.

Programmer Response: 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.

ICE034A MODS STATEMENT OPERAND ERROR

Explanation: Critical. An incorrect number of parameters follows 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 DD statement is present.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that any MODS control statements have the following format:

```
  MODS exit=(name,size,
       {DDname of library|
       SYSIN}
       [,T|,N|,S])...
```

If SYSIN has been specified, make sure that a SORTMODS DD statement is also included in the step.

ICE035A DUPLICATE MODS ROUTINE OPERAND

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: Termination when all control statement scanning is complete.

Programmer Response: Make sure that the MODS control statement does not use duplicate names.

ICE036I B = xxxxxx

Explanation: This message communicates the blocking used by the Sort (nonstandard techniques) 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 (= sort block size) is substituted for xxxxxx in the message text.

System Action: None.

Programmer Response: None.

ICE037I 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 (old techniques). The number of records is substituted for the xxxxxx in the text of the message as shown above.

System Action: None.

Programmer Response: None.

ICE038I NMAX APPROXIMATELY = xxxxxx

Explanation: The message communicates an estimate of the maximum number of records that can be sorted using the intermediate storage and main storage available to sort/merge for the current application. The number replaces the xxxxxx in the text of the message as shown above. For magnetic tape, Nmax is calculated assuming that 2400-foot tapes are used. For disk, no secondary allocation is taken into account. For variable-length records, the value is based on maximum record length.

System Action: None.

Programmer Response: None.

ICE039A INSUFFICIENT MAIN STORAGE [- ADD xxK BYTES]

Explanation: Critical. There is not enough main storage available for a disk technique sort to execute, or main storage is fragmented.

System Action: The program terminates.

Programmer Response: The message normally indicates how much more main storage is needed. Add that amount to the main storage already allocated to the Sort program and recode the REGION parameter and/or the SIZE parameter in the PARM field of the EXEC statement. If the message does not indicate the amount of additional storage needed, then the reason is fragmented storage and/or too large reserved storage value or exit sizes compared to the total storage available to sort/merge. Respond according to the rules below. Alternatively, use the formula for calculating minimum storage requirements given in Section 8 under "Main Storage Requirements."

If routines are used at program exits, their size should be added to this minimum value. For efficient sorting, allow at least 50% more storage than the minimum required. Check also with the information given in message ICE092I or ICE093I.

Storage requirements can be reduced by decreasing either the input block size or the number of intermediate storage areas. See also message ICE092I or ICE093I.

ICE040A INSUFFICIENT WORK UNITS

Explanation: Critical. There are not enough work data sets to allow program execution. This can occur when work data sets are on tape; and when they are on disk, if the standard disk technique is not being used. In a merge-only application, this message may be caused by incorrect specification of one or more input units (SORTIN01, etc.).

System Action: Termination when all control statement scanning is complete.

Programmer Response: 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, starting with SORTWK01. If tape is used, make sure that at least three work data sets were assigned to the program. If direct-access devices are used, make sure that at least three areas of at least three tracks each are assigned.

ICE041A N GT NMAX

Explanation: Critical. The exact number of records specified in the FILSZ or SIZE operand of a SORT control statement is greater than the maximum Sort capacity calculated by the program (applies when the standard disk technique is not used).

System Action: The program terminates.

Programmer Response: Check FILSZ or SIZE operand of SORT control statement for error. If the 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.

ICE042A UNIT ASSIGNMENT ERROR xxxxxx

Explanation: Critical.

1. An invalid combination of input, work, and output devices has been assigned to sort/merge.
2. Duplicate DDnames have been specified. xxxxxx represents the DDname of the data set on which the error was encountered.
3. If xxxxxx says DYNALLOC, either wrong device type or too many work data sets are specified.

System Action: Termination when all control statement scanning is complete.

Programmer Response: For case (1), ensure that <u>no</u> 7-track tape units are assigned as intermediate storage if 7-track tape units are <u>not</u> used as input.

For case (2), eliminate duplicate DDnames.

For case (3), check that the device type specified is supported by the program (see Section 3 under "Storage Devices") and available at your installation; and check whether you have exceeded the maximum number of areas permitted for the storage type used (see Figure 4 in Section 3).

ICE043A INVALID DATA SET ATTRIBUTES SPECIFIED xxxxxx [yyyyyy]

Explanation: Critical. Either: DD statements that define input and output data sets contain information conflicting with each other, with information on the data set labels, or with the default values assumed for DCB subparameters by the program (See Figure 12 in Section 5 for a summary of DCB subparameters); or a DD statement for input or output specifies a cataloged disk data set which does not exist on the volume pointed to by the catalog entry.

xxxxxx is the name of the DD statement in error.

yyyyyy is the error description.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check DD statements for input and output data sets for conflict in the BLKSIZE (block size), RECFM (record format), and LRECL (logical record length) subparameters. Input and output must have the same record type (fixed or variable). When sorting variable length records and no exits are used, the maximum SORTIN LRECL must not exceed the maximum SORTOUT LRECL. Check the volumes of input data sets.

ICE044I EXIT Exx INVALID OPTION

Explanation: An invalid input/output option 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 ignored.

Programmer Response: Check the parameter list passed by the user-written routine against the table at the end of this appendix before rerunning the application. An _x_ in the table indicates an option which is allowed with the exit in question.

ICE045I END SORT PH

Explanation: The sort (input) phase has been successfully executed. Only appears when BALN or POLY tape technique is used.

System Action: None.

Programmer Response: None.

ICE046A SORT CAPACITY EXCEEDED [RECORD COUNT: xxxxxxx]

Explanation: Critical. Sort capacity has been reached. The count xxxxxxx is an approximation of the number of records that sort/merge can handle with the assigned primary intermediate storage plus the available amount of secondary allocated extents. If intermediate storage is on disk, and secondary allocations have been allowed, sort/merge will override any system B37 abend and continue processing; this message will only be issued when no more space is available on any allocated SORTWK disk pack.

System Action: The program terminates.

Programmer Response: 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 work data sets.

If a direct access device is used for intermediate storage, assign more tracks to sort/merge. Note that reverse sequence files may require more space. Alternatively, increase the main storage available to sort/merge.

If you have difficulty assigning sufficient disk space, check message ICE092I or ICE093I to see what technique is being used. If the message says BLOCKSET, you can save disk space by using the DEBUG statement to force sort/merge to use a different technique, as described in Appendix A.

ICE047A RCD COUNT OFF, IN xxxxxxx, OUT xxxxxxx

RECORD COUNT OFF, SPECIFIED xxxxxxx, RECEIVED xxxxxxx

Explanation: Critical. The number of records entering and leaving a program phase are not equal. The message appears if the number of records entering and leaving program phase 1 (and phase 2 of old technique sorts) are not equal, provided an actual value for the FILSZ or SIZE parameter was specified in the SORT control statement. The IN field will contain the specified value for FILSZ or SIZE. The OUT field will contain the end of phase record count, which has been adjusted by the number of records inserted or deleted by user-written routines.

If FILSZ or SIZE parameter actual values were not specified, the check is not made until the end of the output phase, where an unequal compare will cause message ICE025A to be issued together with messages ICE054I and ICE055I.

The second message text is used with the standard disk technique.

System Action: The program terminates.

Programmer Response: Make sure that the value of the FILSZ (or SIZE) parameter in the SORT control statement is accurate. See also message ICE025A above.

ICE048I NMAX EXCEEDED

Explanation: Sort/merge 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: No response necessary. (The number of records sorted is equal to the NMAX calculated by sort/merge. See message ICE038I.)

ICE049I SKIP MERGE PH

Explanation: For a tape sorting application, it is not necessary to execute the intermediate merge phase because the number of sequences created by the sort (input) phase is ≤ the merge order.

System Action: Control is passed directly from the sort (input) phase to the final merge (output) phase.

Programmer Response: None. Note that no E2x exits will be taken in this case.

ICE050I END MERGE PH

Explanation: A tape technique program's intermediate merge phase (Phase 2) has been successfully executed.

System Action: None.

Programmer Response: None.

ICE051A UNENDING MERGE

Explanation: Critical. Non-standard technique: there is not enough intermediate storage assigned to successfully complete the program's intermediate merge phase. Standard technique: there is not enough main storage available to merge two strings (5 buffers required)

System Action: The program terminates.

Programmer Response: Assign more intermediate storage or main storage and rerun the job. Note that reverse sequence files may require more space.

ICE052I END OF SORT/MERGE

Explanation: The program has been executed.

System Action: Return is made to the operating system or invoking program.

Programmer Response: None.

ICE053A OUT OF SEQUENCE

Explanation: The current record leaving phase 2 or 3 is not in collating sequence with the last record blocked for output.

System Action: The program terminates.

Programmer Response: If a user-written routine was modifying the records leaving the phase at the time this message was printed, check the routine thoroughly. If out-of-sequence records are to be inserted in phase 3 by your routine, make sure that the correct parameter to suppress the sequence check is returned to sort/merge (tape and nonstandard disk sorts only).

Check also whether the VERIFY installation option was in effect. If so, the problem may be a program error, and can be bypassed by forcing sort/merge to use a different sorting technique. This is done with the DEBUG control statement as described in Appendix A. (See also OS/VS Sort/Merge Installation Guide.)

ICE054I RECORDS - IN:xxxxxxx, OUT:xxxxxxx[, - END OF SORT]

Explanation: This message lists the number of records accepted by the sort/merge from the input data set and the number of records in the output data set. The numbers replace the xxxxxxx in the text of the message as shown above. Leading zeros are suppressed. If an exact file size has been specified, the number specified appears in the IN field. (Not the standard disk technique.) In a merging application, if file size has not been given the IN field is zero. If no other message follows, the sort/merge has been successfully terminated.

System Action: None.

Programmer Response: If you are using exit E15 and/or E35 and have any reason to suspect that you are 'losing' or 'gaining' records, check with message ICE055I. The sum of RECORDS IN plus INSERT should always be equal to the sum of RECORDS OUT plus DELETE. If it is not, you should also receive message ICE025A.

ICE055I 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: See message ICE054I above.

ICE056A SORTIN [SORTOUT] NOT DEFINED

Explanation: Critical. SORTIN and/or SORTOUT do not appear as DDnames on DD statements supplied to the 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: Check DD statements for error.

ICE057A 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: Check DD statements for error.

ICE058A 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: Check DD statements for error.

ICE059A RECORD LENGTH INVALID FOR device

Explanation: Critical. The record length in the input data set(s) is either less than 18 bytes when work units are tape or is too large for the assigned intermediate storage device. For example, if a nonstandard disk technique was used, a record which cannot be contained on one disk track is too large.

System Action: The program terminates.

Programmer Response: If the record is too small, redefine sort/merge 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	BALN	CRCX
2314 series	7284	7276
3330 series	13014	13014
3340	8364	8356
3350	19060	19052
Tape	32768	-
Tape (spanned records)	27400	-

If EQUALS is specified the maximum record length is reduced by 4 bytes.

ICE060A 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: Make sure that no DD statements are in error. Make sure that deferred mounting of direct access intermediate storage data sets is not specified.

ICE061A I/O ERROR, jobname, stepname, unit address, device type, DDname, operation attempted, error description, last seek address or block count, access method. (SYNADAF)

I/O ERROR, DDname, DEV address, ECB completion code, CSW status bytes, SENSE sense bytes.

Explanation: Critical. This message is generated for one of following reasons:

- The job control statements 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 spanned record on SORTIN could not be properly assembled.
- A permanent error occurred during an I/O operation on the indicated device.

The most likely cause is a hardware-related error.

System Action: If no user options are specified, the program terminates.

Operator Response: If the 'error description' field in the message does not contain 'WRNG. LEN. RECORD', execute the job again with the indicated unit offline, using an alternative unit and/or volume in its place during execution.

Programmer Response: 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 (except for BLKSIZE) specified in the SORTIN01 DD statement correctly describes the data in this device.

If the error persists, a bypass may be obtained by forcing sort/merge to use a different sorting technique. This is done with the DEBUG control statement described in Appendix A.

ICE062A LINK-EDIT ERROR

Explanation: Critical. The linkage editor found a serious error; execution of the sort/merge program is impossible.

System Action: The program terminates.

Programmer Response: 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 link-edited together do not contain duplicate entry point names.

ICE063A OPEN ERROR 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: Check for any of the following:

- A missing or invalid DD statement.
- Conflicting DCB information, for example, fixed block records and block size not a multiple of record length.
- Concatenated input without the largest block size specified for the first data set.
- Concatenated, fixed-length input with different LRECL specifications.
- A partitioned data set member specified as a user exit routine cannot be found.

ICE064A DELETE ERR

Explanation: Critical. The sort/merge program was unable to delete either itself or a user exit routine. This message should appear only when exit routines are used.

System Action: The program terminates.

Programmer Response: Make sure that the user exit routines are not modifying the sort/merge program code and information areas, and rerun the job.

ICE065A PROBABLE DECK STRUCTURE ERROR

Explanation: Critical. The end of the SYSIN data set was found before all needed user exit modules were read, or the end of the SYSIN data set was not found after all specified modules were read.

System Action: The program terminates.

Programmer Response:

1. Check that the MODS statement specifies the correct routines.
2. Be sure the SYSIN data set contains all exit routines that the MODS statement specifies it will contain, and only those.
3. Check for misplaced job control language statements, especially preceding a user exit routine on SYSIN.

ICE066I APROX RCD CNT xxxxxxxx

Explanation: Critical. Sort capacity has been reached. The count xxxxxxxx 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: Respond as indicated in the accompanying message, ICE046A.

ICE067I INVALID 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. If a parameter is entered more than once, the first entry is used (if valid).

System Action: Processing continues. Invalid parameters are ignored.

Programmer Response: No action is necessary. For later runs, make sure that the optional parameters are valid. Valid parameters are described in Section 5 under "'PARM' Field Options."

ICE068A OUT OF SEQ SORTINxx

Explanation: Critical. During a merge-only, a data set was found to be out of sequence. The xx is replaced by the data set identification (01 to 16). If input is being supplied through exit E32, then 01 signifies the first input file, 02 the second, and so on.

System Action: The program terminates.

Programmer Response: 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 on the same control fields, and that they all have a similar format. Check whether you have also received message ICE072A.

If input is being supplied through E32, check your routine to make sure records are passed to the merge from the correct file.

If you are reading in variable-length VSAM records through exit E32, check the format and accuracy of the RDW which you are building at the beginning of each record.

ICE069A INVALID SIGN

Explanation: Critical. The first byte of signed numeric data with leading separate sign, or the last byte of signed numeric data with trailing separate sign does not contain a valid sign character.

System Action: The program terminates.

Programmer Response: Check the description of data format in the FIELDS or FORMAT parameter of the SORT or MERGE statement.

ICE070I FILE SIZE xxxxxxxx

Explanation: This message appears when the balanced disk technique is used, and indicates that either the input file size was not specified (FILSZ or SIZE) in the SORT statement, or a file size of xxxxxxxx (decimal value) was specified.

System Action: Processing continues.

Programmer Response: No response necessary. If xxxxxxxx is 'NOT SPECIFIED', supply file size information for later runs to get better performance.

ICE071A INVALID RETURN CODE FROM EXIT Exx

Explanation: Critical. A user routine at the exit Exx (can be E15, E25, E32, or E35) has returned an invalid return code to the program, or a return code in 0 or 4 has been given at end of file.

System Action: The program terminates.

Programmer Response: Check the user routine concerned thoroughly and ensure that the return code is either 0, 4, 8, 12, or 16 (only 0, 4, or 16 for E25, and 8, 12 or 16 for E32).

Check also that:

- An E35 routine always finishes by returning 8 (do not return) or 16 (terminate).
- If no SORTOUT DD statement is provided, the E35 routine is processing <u>all</u> records passed by sort/merge before returning 8 (do not return).

ICE072I CONTROL FIELD NOT WITHIN RECORD
CONTROL FIELD NOT WITHIN MINIMUM RECORD LENGTH

Explanation: A RECORD statement specifies a minimum record length (L4) which cannot contain all control fields specified in the SORT or MERGE statement.

System Action: The L4 value is adjusted. Processing continues.

Programmer Response: Check that the L4 value is not smaller than the highest control field position.

ICE073A VARIABLE RECORD TOO LONG

Explanation: Critical. A deblock routine L1 or L2 value specified (or supplied by default) on the RECORD statement, or, if there was no RECORD statement, than the DCB LRECL on the SORTIN DD statement or data set label.

System Action: The program terminates.

Programmer Response: Check the input both at E15, if used, and in SORTIN. Then either delete the extra long records or increase the RECORD statement L1/L2 value and/or the SORTIN DD statement DCB LRECL value.

If you have VSAM records, remember that they are increased in length by the 4-byte record descriptor word added when they enter the sort/merge program. If you are reading input through E15, check the format of the RDW you are building at the beginning of each record.

ICE074I RECORD LENGTH L1 OR L3 OVERRIDDEN

Explanation: Either the L1 value for the LENGTH parameter of the RECORD statement is not the same as the LRECL value for SORTIN or SORTIN01; and/or the L3 value is not the same as the SORTOUT LRECL value. For VSAM, the equivalent of LRECL is maximum RECSZ.

System Action: Processing continues with the L value(s) overrridden.

Programmer Response: For subsequent runs, check all the record lengths. Take special note of the L2 value. If you did not specify one, it will have defaulted to the value you specified for L1 (and will not have been overridden by the LRECL value). If the L2 value is too small it can cause program termination at any of a number of points, and the error can be difficult to detect.

If you have variable-length records (shown in message ICE088I), check that the L1 value used is actually a maximum. The logical record length (LRECL) of the input file is also given in message ICE088I.

ICE075A VSAM CB ERROR (xx) AT aaaaaa

Explanation: aaaaaa represents the storage address at which the error was detected. xx is the VSAM return code, in decimal, from a GENCB, MODCB, SHOWCB, or TESTCB macro.

System Action: The program terminates, unless the error is detected during close, when the program will try to close all remaining VSAM data sets before terminating.

Programmer Response: Refer to the <u>OS/VS VSAM Programmer's Guide</u> for the meaning of the return code, and if possible take appropriate action.

ICE076A VSAM INPUT ERROR i(xxx) yyyyyyyy

Explanation: i is replaced by either P (physical) or L (logical), describing the type of error encountered. <u>xxx</u> is the VSAM feedback code from a GET macro, in decimal; and <u>yyyyyyyy</u> is either the DDname of the data set in error or (if available) the VSAM SYNAD message.

System Action: The program terminates.

Programmer Response: Refer to the <u>OS/VS VSAM Programmer's Guide</u> for the meaning of the return code, and if possible take appropriate action.

ICE077A VSAM OUTPUT ERROR i(xxx) [yyyyyyyy]

Explanation: i is replaced by either P (physical) or L (logical), describing the type of error encountered. <u>xxx</u> is the VSAM feedback code from a PUT macro, in decimal. <u>yyyyyyyy</u> (if available) is the VSAM SYNAD message.

System Action: The program terminates.

Programmer Response: Refer to the <u>OS/VS VSAM Programmer's Guide</u> for the meaning of the return code, and if possible take appropriate action.

ICE078A VSAM OPEN ERROR (xxx) yyyyyyyy

Explanation: <u>xxx</u> is the VSAM OPEN ERROR return code, in decimal. <u>yyyyyyyy</u> is the DDname of the data set on which the error was encountered.

System Action: The program terminates.

Programmer Response: Refer to the <u>OS/VS VSAM Programmer's Guide</u> for the meaning of the return code, and, if possible, take appropriate action. Check that the SORTIN and SORTOUT VSAM data set is not the same data set.

ICE079A VSAM CLOSE ERROR (xxx) yyyyyyyy

Explanation: xxx is the VSAM CLOSE ERROR return code, in decimal. yyyyyyyy is the DDname of the data set on which the error was encountered.

System Action: The program terminates.

Programmer Response: Refer to the VSAM Programmer's Guide for the meaning of the return code, and if possible take appropriate action.

ICE080I IN MAIN STORAGE SORT

Explanation: All records were sorted in main storage, that is, no sort work areas were used.

System Action: None.

Programmer Response: None.

ICE081A COMMUNICATION AREA NOT FULLY ADDRESSABLE

Explanation: The program has run out of addressability for certain dynamic areas and routines. This situation can only arise if a large number of intermediate storage areas is specified, at the same time as a very large number of control fields; it is more likely to occur if control fields are not EBCDIC character (CH) or binary (BI).

System Action: The program terminates.

Programmer Response: Specify fewer intermediate storage areas; and/or combine control fields which are adjacent; and/or redefine control fields as CH or BI, etc.

ICE082I CHECKPOINT CANCELLED

Explanation: When no more work data set tracks are available, the tracks allocated for CKPT (if requested) are given back to the Sort work data sets.

System Action: The program continues, but no checkpoints are taken.

Programmer Response: Increase work space allocation for next run.

ICE083A UNAVAILABLE RESOURCES DYNALLOC (xxxx)

Explanation: xxxx is the return code from the MVS dynamic allocation facility. The requested work data sets were not available on the system.

System Action: The program terminates.

Programmer Response: Be sure that the requested work files can be allocated on the available resources. See OS/VS2 MVS System Programming Library: Job Management for the codes.

ICE084I EXCP ACCESS METHOD USED FOR XXXX

Explanation: Written when Peerage or Vale disk techniques have used EXCP for SORTIN and/or SORTOUT data sets. FLR-Blockset and VLR-Blockset always use EXCP.

System Action: None.

Programmer Response: None, unless you have any problems reading SORTIN or writing SORTOUT. If you do, you can force sort/merge not to use EXCP by use of the DEBUG control statement, as described in Appendix A.

ICE085I xxx PERCENT OF PRIMARY WORK DATA SET EXTENTS REQUIRED [TRACKS USED FOR SECONDARY ALLOCATION yy]

Explanation:

1. Written for all record sorts using one of the Blockset disk techniques, except those done in main storage. xxx is the percentage required of the primary allocated work data set extents for the current file sorted. If this percentage exceeds 100, then secondary allocation was used.
2. Written for Peerage or Vale disk technique sorts if secondary allocation is used. yy is the number of tracks used for secondary allocation for SORTWK areas.

System Action: None.

Programmer Response: If the percentage is approximately 150% (or more) or the number of secondary allocation cylinders is approximately 50% (or more) of the number of primary cylinders specified in the SORTWK statement, you should consider allocating more primary cylinders to improve the program's performance.

ICE087I EXCPVR CANCELLED

Explanation: Not enough pages were available for page fixing. The program will use normal EXCP for its disk work files.

System Action: None.

Programmer Response: None.

ICE088I jobname.stepname, INPUT LRECL=xxxxxx, BLKSIZE= YYYYYY, TYPE={F|V|VS}

Explanation: Gives details of current job and step information. The types printed in the message are:

F fixed-length blocked or unblocked records
V variable-length records (EBCDIC or ASCII)
VS variable spanned records

System Action: None.

Programmer Response: None.

ICE089I jobname.stepname, INPUT LRECL=xxxxxx, TYPE={F|V}

Explanation: As for ICE088, but used when all records are supplied via exit E15.

System Action: None.

Programmer Response: None.

ICE090A CONFLICTING OPERANDS ON MODS STATEMENT

Explanation: A routine was defined in the MODS statement as being in SYSIN (s parameter), and as needing no link-editing (e parameter set to N).

System Action: The program terminates.

Programmer Response: Check the MODS statement.

ICE091I NONSTANDARD DISK TECHNIQUE USED

Explanation: Usually, you have used the DEBUG statement to force a disk technique other than the standard. On an exception basis, however, a nonstandard technique might have been selected by sort/merge if you have excessively long control fields.

System Action: The sort/merge continues, using a nonstandard technique, if sufficient work space is available.

Programmer Response: None.

ICE092I MAIN STORAGE = (x,y,z), NMAX = n,t

Explanation: Information related to the sort/merge application:

x is the main storage (SIZE) specified, or supplied by default.

y is the main storage theoretically available to sort/merge, taking into account any MAXLIM or MINLIM figures specified when the program was installed.

z is the main storage actually available to sort/merge, after any other program has taken what it needed from the partition or region (invoking program and/or exit routines).

n is the approximate number of records which can be sorted in available main storage. However, this is true only if there are no SORTWK data sets. If SORTWK is specified, then n = the approximate number of records that can be sorted on the SORTWK data sets.

t is the technique used.

System Action: None.

Programmer Response: None, unless sort/merge subsequently terminated abnormally. In that case, check the z value to see how much storage was really available to sort/merge. If space was the problem, you will probably also have received message ICE039A; but if storage was heavily fragmented, the result could instead be a system 80A abend in either sort/merge or one of your own routines. Note that you could need considerably more than the normal minimum if the partition or region is fragmented.

If you have difficulty in supplying enough main storage, check the t value: if it says that one of the Blockset techniques has been used, you can save some space by forcing sort/merge to use a different technique. This is done with the OPTION control statement, as described in Appendix A.

ICE093I MAIN STORAGE = (MAX,y,z), NMAX = n,t

Explanation: Information related to the sort/merge application:

MAX was specified or the value specified is the same as MAXLIM.

y is the main storage theoretically available to sort/merge, taking into account any MAXLIM or MINLIM figures specified when the program was installed.

z is the main storage actually available to sort/merge, after any other program has taken what it needed from the partition or region (invoking program and/or exit routines).

n is the approximate number of records which can be sorted in available main storage. However, this is true only if there are no SORTWK data sets. If SORTWK is specified, then n = the approximate number of records that can be sorted on the SORTWK data sets.

t is the technique used.

System Action: None.

Programmer Response: None, unless sort/merge subsequently terminated abnormally. In that case, check the z value to see how much storage was really available to sort/merge. If space was the problem, you will probably also have received message ICE039A; but if storage was heavily fragmented, the result could instead be a system 80A abend in either sort/merge or one of your own routines. Note that you could need considerably more than the normal minimum if the partition or region is fragmented.

If you have difficulty in supplying enough main storage, check the t value: if it says that one of the Blockset techniques has been used, you can save some space by forcing sort/merge to use a different technique. This is done with the OPTION control statement, as described in Appendix A.

ICE094I SMF FEATURE NOT PRESENT IN THE SYSTEM—SMF RECORD NOT WRITTEN

Explanation: The CVT control block indicates that the SMF facility is not present in the programming system.

System Action: The data collection for the record length statistics and the writing of the SMF record to the SMF data set will be bypassed.

Programmer Response: Determine whether or not the SMF facility is properly installed and initialized on your system. Correct as necessary.

ICE095A INVALID OPTION STATEMENT OPERAND

Explanation: Critical. An invalid keyword operand has been detected on an OPTION control statement.

System Action: The program terminates when all control statement scanning is complete.

Programmer Response: Make sure that the OPTION control statement does not contain an invalid keyword operand. See Section 4 for valid keywords.

ICE096I SUCCESSFUL RECOVERY FROM B37 ABEND(S) FOR WORK DATA SET(S)

Explanation: Sort/merge successfully recovered from one or more B37 ABENDs that occurred when sort attempted to acquire more disk space than was available on one of the work data sets allocated by sort.

System Action: Processing continues.

Programmer Response: None.

ICE097I SORT ATTEMPTING RECOVERY FROM B37 ABEND FOR SORTWK DATA SET

Explanation: Issued only to the master console after a B37 ABEND that occurred when sort attempted to acquire more disk space than was available on one of the work data sets allocated by sort.

System Action: Processing continues.

Programmer Response: None.

ICE098I AVERAGE RECORD LENGTH = xxxx BYTES

Explanation: xxxx is the number of bytes in the variable-length records (including the record descriptor word) divided by the number of sorted records. The number of sorted records includes all records received, added, and/or deleted before the E35 exit is taken.

System Action: None.

Programmer Response: If the value xxxx is more than 350, it should be included in the RECORD statement as the average record length (L5 parameter) for future sorts, so that sort/merge can optimize for the best sorting technique.

ICE099A BLDL FAILED FOR SORTIN DATA SET

Explanation: Critical. A bad return code was returned from a BLDL macro issued when SORTIN is defined as a PDS member.

System Action: The program terminates.

Programmer Response: Ensure that the PDS member specified as SORTIN exists.

ICE120-125

Explanation: Messages produced by using the DEBUG control statement; see Appendix A.

ICE900-990

Explanation: Messages produced by using the DIAG option; see Appendix A.

	Exit[1]					
Option	**E18**	**E19**	**E28**	**E29**	**E38**	**E39**
SYNAD	x	x	x	x	x	x
EXLST	x[2]	x	x	x	x	x
EROPT	x		x		x	
EODAD	x					
VSAM EXLST	x				x[3]	x
VSAM PASSWORD	x				x[3]	x

[1] See ICE044I for reference to this table.
[2] Cannot be used if input is concatenated on unlike devices.
[3] For merge-only applications.

APPENDIX D. EXAMPLES OF CONTROL STATEMENTS FOR SORT/MERGE APPLICATIONS

LIST OF EXAMPLES

The table below describes the examples which are provided in this appendix.

No.	Description	Input	Output
1	Disk sort	Blocked fixed-length records on 3350	Blocked fixed-length records on 9-track
2	3330 sort, with exits	Blocked fixed-length records on 3330	Blocked fixed-length records on 3330, same unit as input
3	3330 sort, one exit, PROC=SORT	Fixed-length unblocked records on a 3340 DASF	Fixed-length blocked records on a 3340 DASF
4	3330 sort, tape I/O, exits	Variable-length records on 3400 tape	Variable-length records on 3400 tape
5	3340 sort, ASCII tape I/O	Variable-length ASCII records on 9-track tape	Variable-length ASCII records on 9-track tape
6	3380 sort, ASCII tape I/O	Variable-length ASCII records on 9-track tape	Variable-length ASCII records on 9-track tape
7	Tape sort	Blocked fixed-length records on 9-track tape	Blocked fixed-length records on 9-track tape
8	Tape sort, with exits	Fixed-length blocked records on two unlabeled 9-track volumes	Fixed-length blocked records on one 9-track tape
9	Tape sort 7-track	Blocked fixed-length records on 7-track unlabeled tape	Blocked fixed-length records on 7-track labeled tape
10	3350 sort, exits	Variable-length blocked records on 3350	Variable-length blocked records on 3350
11	Sort with no SORTWK, 1 exit	Fixed-length blocked records on 3330	Fixed-length blocked records on 3340
12	Concatenated input, dynamically allocated work areas	A concatenation of three data sets on 3330-1, 2400, and 3340	Blocked fixed-length records on 9-track tape
13	3330-1 sort called from another program	Fixed- or variable-length records	Fixed- or variable-length records
14	Merge four unlabeled tapes	Blocked fixed-length records on four 9-track tapes	Blocked fixed-length records on one 9-track tape
15	Merge two 3330 files, exits	Variable-length blocked records on 3330	Variable-length blocked records on 3330
16	Merge three 7-track tapes	Blocked fixed-length records on three 7-track tapes	Blocked fixed-length records on one 7-track tape

SORT EXAMPLES

```
//EXAMP1   JOB  A402,PROGRAMMER,REGION=256K                     01
//SRT     EXEC  PGM=SORT,PARM='SIZE(MAX)'                       02
//SYSOUT    DD  SYSOUT=A                                        03
//SORTIN    DD  UNIT=3350,VOL=SER=000101,DISP=SHR,DSN=INPUT     04
//SORTOUT   DD  UNIT=3400-3,DSN=OUTPUT,VOL=SER=222222,          05
//              DISP=(,KEEP)                                    06
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(10))                     07
//SORTWK02 DD   UNIT=SYSDA,SPACE=(CYL,(10))                     08
//SYSIN     DD  *                                               09
   SORT  FIELDS=(5,12,CH,A),FILSZ=E2000                         10
/*
```

Example 1. DISK SORT

This example is the same as that shown in Section 2.

01 The JOB statement introduces this job to the operating system, and specifies a region of 256K bytes.

02 The EXEC statement calls the program by its alias SORT and specifies that the program should use all the main storage available to it.

03 The SYSOUT DD statement directs the sort messages to system output class A.

04 The SORTIN DD statement describes an input data set named INPUT. The data set is on a 3350 disk with the serial number 000101. The DISP parameter indicates that the data set is known to the operating system.

05-06 The SORTOUT DD statement describes the output data set. Output will be recorded on a 9-track tape and will be kept. The data set will be placed on a standard label tape with tape volume number 222222. By default, format, record length, and block size are the same as for SORTIN.

07-08 These DD statements define temporary work data sets. The two data sets are on SYSDA direct access devices. Ten cylinders are specified for each data set.

09 A data set follows in the input stream.

10 SORT statement. The FIELDS operand describes one field. It begins on byte 5 of each record, is 12 bytes long, contains character (EBCDIC) data, and is to be sorted into ascending order. The file size is estimated to be 2000 records.

```
INPUT      Blocked fixed-length records on 3330.

OUTPUT     Blocked fixed-length records on 3330, same unit as input.

INTERMEDIATE STORAGE   Three 3330 areas of 10 cylinders each.

USER ROUTINES    Four: two change records lengths, one changes control
                 fields, one decides what to do if Nmax is exceeded.

OPTIONS    Estimated data set size; maximum main storage allocation.

//EXAMP2    JOB   A402,PROGRAMMER
//STEP1     EXEC    SORT,PARM='SIZE(MAX)'                         01
//SORTIN    DD    UNIT=3330,VOL=SER=000101,DISP=(OLD,DELETE),     02
//                DSN=INPUT                                       03
//SORTOUT   DD    UNIT=AFF=SORTIN,VOL=SER-000101,DISP=(OLD,       04
//                KEEP),SPACE=CYL,(21,1)),DSN=OUTPUT,             05
//                DCB=(LRECL=80)                                  06
//SORTWK01 DD     UNIT=(3330,SEP=(SORTIN,SORTOUT)),               07
//                SPACE=(CYL,(10),,CONTIG)                        08
//SORTWK02 DD     UNIT=(3330,SEP=(SORTIN,SORTOUT)),               09
//                SPACE=(CYL,(10),,CONTIG)                        10
//SORTWK03 DD     UNIT=(3330,SEP=(SORTIN,SORTOUT)),               11
//                SPACE=(CYL,(10),,CONTIG)                        12
//MODLIB    DD    DSNAME=YOURRTNS,DISP=SHR                        13
//SORTMODS DD     UNIT=2314,SPACE=(CYL,(1,,3))                    14
//SYSIN     DD    *                                               15
   SORT     FIELDS=(3,8,ZD,E,40,6,CH,D),FILSZ=E30000              16
   RECORD   TYPE=F,LENGTH=(,100,80)                               17
   MODS     E15=(MODREC,784,MODLIB,N),E16=(E16,1024,MODLIB,N),    18
            E35=(ADDUP,912,SYSIN),E61=(CHGE,1000,SYSIN)           19
   END                                                            20
Object deck for ADDUP routine
Object deck for CHGE routine
/*
```

<u>Example 2</u>. 3330, PROC=SORT, EXITS

01 The EXEC statement specifies the SORT cataloged procedure (and not the SORTD procedure) because user-written routines that require link-editing are included in the application. SIZE(MAX) instructs the program to allocate the maximum amount of main storage available for program execution.

02-03 The SORTIN DD statement describes an input data set on a 3330 DASF. DCB parameters are supplied by the system (since DISP=OLD). The data set will be deleted after this job step.

04-06 The SORTOUT 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 and block size as the input records, so these values need not be supplied. They are shorter (see the RECORD statement), so LRECL must be specified.

07-12 The three SORTWKnn DD statements describe two work data sets on 3330. Each area contains 10 cylinders. The UNIT specification means that the intermediate storage area is not to be located on the same device as the SORTIN and SORTOUT data sets.

13 Defines the data set containing the load modules for the E15 and E16 user routines.

14 Defines a data set on which the routines in SYSIN specified in the MODS statement (ADDUP and CHGE) will be placed.

15 A data set follows in the input stream.

16 SORT statement. The FIELDS operand describes two control fields. The first will be changed by a user routine (at the E61 exit—see the MODS statement) before the program places it into ascending order. The second control field will not be modified and will be placed in descending order.

17 RECORD statement. The fixed-length records in the input data set are 120 bytes long. A user exit routine (at the E15 exit) changes them to 100 bytes during the sort phase. A user routine at the E35 exit again changes the length during the final merge phase, to 80 bytes each.

18-19 MODS statement. The statement describes four user routines. The first two are in a library that is defined on a job control statement with the ddname MODLIB; these two routines have the member names MODREC and E16, respectively. Neither routine requires additional link-editing. The next two routines are in object form in the input stream. Their names are ADDUP and CHGE, respectively. They must be link-edited together with other routines in their phases that require link-editing.

20 END statement. This statement is required because of the user routines in the input stream.

21-22 Object decks for your user exit routines must appear in the input stream in numerical exit number order. ADDUP is the routine for the E35 exit, so it appears before CHGE, the routine for the E61 exit.

23 Marks the end of the SYSIN data set.

```
INPUT       Fixed-length unblocked records on a 3340 DASF.

OUTPUT      Fixed-length blocked records on a 3340 DASF.

INTERMEDIATE STORAGE   Three 3330 areas, 1 cylinder each.

USER ROUTINES   E35 exit routine shortens each record by 30 bytes
                as it leaves the merge.

OPTIONS     Exact data set size, maximum sort main storage option,
            message option.

//EXAMP3     JOB   A402,PROGRAMMER
//STEP1      EXEC  PROC=SORT,PARM='SIZE(MAX),NOFLAG'           01
//SORTIN     DD    DSNAME=INFILE,VOL=SER=INP214,UNIT=3340,     02
//                 DCB=(RECFM=F,BLKSIZE=80),                   03
//                 DISP=(OLD,DELETE)                           04
//SORTOUT    DD    DSNAME=OUTFILE,VOL=SER=DLIB02,UNIT=3340,    05
//                 DCB=(RECFM=FB,LRECL=50,BLKSIZE=500),        06
//                 DISP=(NEW,KEEP),SPACE=(CYL,(8,1))           07
//SORTWK01   DD    UNIT=3330,SPACE=(CYL,(1))                   08
//SORTWK02   DD    UNIT=3330,SPACE=(CYL,(1))                   09
//SORTWK03   DD    UNIT=3330,SPACE=(CYL,(1))                   10
//USERLIB    DD    DSN=EX35,DISP=SHR                           11
//SYSIN      DD    *
   SORT      FIELDS=(10,5,CH,A),FILSZ=1000                     12
   RECORD    TYPE=F,LENGTH=(,,50)                              13
   MODS      E35=(E35,536,USERLIB,N)                           14
/*
```

Example 3. 3330 SORT, PROC=SORT, 1 EXIT

01 Invokes the SORT cataloged procedure; specifies that the maximum amount of main storage available is to be allocated for the program's execution, that only critical messages are to be produced, and that they are to appear on the appropriate console.

02-04 The input data set consists of fixed-length unblocked records on volume INP214 on a 3340 direct-access facility. The data set will be deleted after this job step.

05-07 The output data set is composed of fixed-length blocked records that will require 8 cylinders on a 3340. Each time space is exhausted, an additional cylinder will be allotted. The data set will be retained.

08-10 Intermediate storage consists of three 3330 areas of one cylinder each.

11 Defines the library that contains the E35 module.

12 SORT statement. The FIELDS operand describes one control field that begins on byte 10 of each record, is 5 bytes long, and contains character (EBCDIC) data; it is to be sorted into ascending order. The optional FILSZ operand indicates that the input data set contains exactly 1,000 records.

13 RECORD statement. Indicates that the input data set contains fixed-length records that will be shortened to 50 bytes each as they leave the final merge.

14 MODS statement. Describes a user routine that will receive control at program exit E35. The name of the routine is E35; it is 536 bytes long, is on the data set defined in the USERLIB DD statement, and needs no further link-editing.

```
INPUT        Variable-length records on 3400 tapes.

OUTPUT       Variable-length records on 3400 tapes.

INTERMEDIATE STORAGE   Two 3330 areas of 15 cylinders each.

USER ROUTINES   E11 routine performs initialization for the E16 Nmax
                routine.

OPTIONS      Estimated data set size.

//EXAMP4     JOB    B999,PROGRAMMER
//STEPN      EXEC   SORT,REGION=128K                              01
//SORTIN     DD     DSNAME=XFILE,VOL=SER=000230,UNIT=3400-3,     02
//                  DISP=OLD,DCB=(RECFM=VB,LRECL=120,             03
//                  BLKSIZE=1200)                                 04
//SORTWK01   DD     UNIT=3330,SPACE=(CYL,(15))                    05
//SORTWK02   DD     UNIT=3330,SPACE=(CYL,(15))                    06
//SORTOUT    DD     DSNAME=YFILE,VOL=SER=000258,UNIT=3400-3,     07
//                  DISP=(NEW,CATLG)                              08
//USERLIB    DD     DSNAME=MYRTNS,DISP=SHR                        09
//SYSIN      DD     *                                             10
   SORT      FIELDS=(20,5,AQ,A),FILSZ=E25500                      11
   RECORD    TYPE=V,LENGTH=(120,,,80,120)                         12
   MODS      E11=(PREPMOD,504,SYSIN,S),E16=(MODMAX,554,           13
             USERLIB,N)                                           14
   ALTSEQ    CODE=(5BEA,7BEB,7CEC)                                15
   END
Object deck for PREPMOD routine to be used at E11
/*
```

Example 4. 3330 SORT, TAPE I/O, PROC=SORT, EXITS

01 Calls the SORT cataloged procedure and indicates that a 128K-byte region is needed for program execution.

02-04 The input data set is named XFILE, resides on 9-track standard labeled tape on a 3400 series magnetic tape unit with the volume serial number 000230, is known to the system, and is not to be deleted. It consists of variable-length blocked records.

05-06 Two intermediate storage areas on 3330s are defined. Each consists of 15 cylinders.

07-08 The output data set is named YFILE, and is to be placed on 9-track standard-labeled tape on a 3400 series magnetic tape unit with the volume serial number 000258. It will contain records of the same format as the input data set. The data set is being created in this job step and is to be cataloged.

09 Defines the library that contains the E16 user routine.

10 Sort control statements follow.

11 SORT statement. Describes one control field that begins on byte 16 of each record data area (not byte 20, since the record descriptor word takes 4 bytes), is 5 bytes long, contains character data which is to be collated according to the modified sequence described in the ALTSEQ statement (format is AQ), and is to be sorted into ascending sequence. The input data set contains approximately 25,500 records.

12 RECORD statement. Indicates that the input data set contains variable-length records with a maximum record length of 120 bytes, a minimum record length of 80 bytes, and an average length of 120 bytes. The RECORD statement is not required for this example, but without it, the

program would assume a minimum record length of 24 bytes (large enough to contain the specified control field) and an average length of 72 bytes (the average of maximum and minimum lengths). Maximum length could have been supplied by default.

13-14 MODS statement. Describes two user routines. The first, PREPMOD, will receive control at exit E11. It is 504 bytes long, is included in SYSIN, and will be link-edited separately. The second user routine, named MODMAX, will receive control at exit E16. It is 554 bytes long. It resides in a library called MYRTNS that is described by the job control statement with the DDname USERLIB. It requires no further link-editing. Because E11 and E16 user routines are being used, the VLR-Blockset technique will not be used.

15 ALTSEQ statement. Specifies that the three characters $, #, and @ are to collate in that order after Z.

```
INPUT        Variable-length ASCII records on 9-track tape.

OUTPUT       Variable-length ASCII records on 9-track tape.

INTERMEDIATE STORAGE   Two 3340 areas of 15 cylinders each and two 3330
                       areas of 10 cylinders each.

USER ROUTINES   None.

OPTIONS      Estimated data set size.

//EXAMP5     JOB   A432,PROGRAMMER
//STEPM      EXEC  SORTD
//SORTIN     DD    DSNAME=SRTFIL,DISP=(OLD,DELETE),UNIT=2400,       01
//                 DCB=(RECFM=DB,LRECL=80,BLKSIZE=404,OPTCD=Q,      02
//                 BUFOFF=L),VOL=SER=311500                         03
//SORTWK01   DD    UNIT=3340,SPACE=(CYL,(15))                       04
//SORTWK02   DD    UNIT=3340,SPACE=(CYL,(15))                       05
//SORTWK03   DD    UNIT=3330,SPACE=(CYL,(10))                       06
//SORTWK04   DD    UNIT=3330,SPACE=(CYL,(10))                       07
//SORTOUT    DD    DSN=OUTFIL,UNIT=2400-3,LABEL=(,NL),              08
//                 DISP=(,KEEP),DCB=(OPTCD=Q,BUFOFF=L)              09
//SYSIN      DD    *
   SORT      FIELDS=(10,8,AC,D),FILSZ=E525000                       10
   RECORD    TYPE=D,LENGTH=(,,,20,23)                               11
/*
```

Example 5. 3340 SORT, ASCII TAPE I/O, PROC=SORTD

01-03 The input data set SRTFIL is on a 9-track tape with the volume serial number 311500. It is known to the system and is deleted after this job step. It consists of variable-length ASCII records which are blocked and have a maximum length of 80 bytes. For this job, the buffer offset is the block length indicator. The records are to be translated from ASCII to EBCDIC (OPTCD=Q).

04-07 Four intermediate storage data sets are defined, two on 3340s and two on 3330 disks.

08-09 The output data set is named OUTFIL. It will be written on a 9-track tape with a density of 1600 bpi. It will be kept. It has no labels. It contains records with the same RECFM, LRECL, and BLKSIZE values as the input (by default).

10 SORT statement. The FIELDS operand describes a control field that begins on byte 6 of each record data area (not byte 10, since the record descriptor word takes 4 bytes), and is 8 bytes long. This field contains character (ASCII) data, and will be sorted in descending order. The input data set contains approximately 525,000 records.

11 RECORD statement. All the records in the input data sets are ASCII records. Their maximum length is supplied by default; the minimum is 20. The average length is 23.

INPUT   Variable-length ASCII records on 9-track tape.

OUTPUT   Variable-length ASCII records on 9-track tape.

INTERMEDIATE STORAGE   One 3380 area of 6 cylinders.

USER ROUTINES   None.

OPTIONS   Estimated data set size.

```
//EXAMP6     JOB   A432,PROGRAMMER
//STEPM      EXEC  SORTD
//SORTIN     DD    DSNAME=SRTFIL,DISP=(OLD,DELETE),UNIT=2400,     01
//                 DCB=(RECFM=D,LRECL=400,BLKSIZE=404,OPTCD=Q,    02
//                 BUFOFF=L),VOL=SER=311500                       03
//SORTWK01   DD    UNIT=3380,SPACE=(CYL,(4))                      04
//SORTOUT    DD    DSN=OUTFIL,UNIT=2400-3,LABEL=(,NL),            05
//                 DISP=(,KEEP),DCB=(OPTCD=Q,BUFOFF=L)            06
//SYSIN      DD    *
   SORT      FIELDS=(10,8,AC,D),FILSZ=E26000                      07
   RECORD    TYPE=D,LENGTH=(,,,20,80)                             08
/*
```

Example 6. 3380 SORT, ASCII TAPE I/O, PROC=SORTD

01-03 The input data set SRTFIL is on a 9-track tape with the volume serial number 311500. It is known to the system and is deleted after this job step. It consists of variable-length ASCII records which are blocked and have a maximum length of 400 bytes. For this job, the buffer offset is the block length indicator. The records are to be translated from ASCII to EBCDIC (OPTCD=Q).

04 One intermediate storage data set is defined on a 3380.

05-06 The output data set is named OUTFIL. It will be written on a 9-track tape with a density of 1600 bpi. It will be kept. It has no labels. It contains records with the same RECFM, LRECL, and BLKSIZE values as the input (by default).

07 SORT statement. The FIELDS operand describes a control field that begins on byte 6 of each record data area (not byte 10, since the record descriptor word takes 4 bytes), and is 8 bytes long. This field contains character (ASCII) data, and will be sorted in descending order. The input data set contains approximately 26,000 records.

08 RECORD statement. All the records in the input data sets are ASCII records. Their maximum length is supplied by default; the minimum is 20. The average length is 80.

INPUT   Blocked fixed-length records on 9-track tape.

OUTPUT   Blocked fixed-length records on 9-track tape.

INTERMEDIATE STORAGE   Four 9-track tapes.

USER ROUTINES   None.

OPTIONS   FORMAT=xx for control fields of like format; estimated data set size.

```
//EXAMP7      JOB    A402,PROGRAMMER
//STEP1       EXEC   PGM=SORT,REGION=64K                            01
//SYSOUT      DD     SYSOUT=A                                       02
//SORTLIB     DD     DSNAME=SM01,SORTLIB,DISP=SHR                   03
//SORTIN      DD     DSNAME=INPUT,VOL=SER=000101,UNIT=2400,         04
//                   DISP=(OLD,DELETE),DCB=(RECFM=FB,               05
//                   LRECL=80,BLKSIZE=800)                          06
//SORTOUT     DD     DSNAME=OUTPUT,UNIT=2400,DISP=(NEW,CATLG),      07
//                   VOL=SER=000102                                 08
//SORTWK01    DD     UNIT=3400-3                                    09
//SORTWK02    DD     UNIT=3400-3                                    10
//SORTWK03    DD     UNIT=3400-3                                    11
//SORTWK04    DD     UNIT=3400-3                                    12
//SYSIN       DD     *
   SORT      FIELDS=(1,6,A,28,5,D),FORMAT=CH,FILSZ=E10000          13
/*
```

Example 7. TAPE SORT, PGM=SORT

01   This EXEC statement calls the program module by its alias, SORT, and indicates that it wants a 64K region in which to operate.

02   The SYSOUT DD statement directs the system output to system output class A.

03   The SORTLIB DD statement defines a private data set containing the sort program modules.

04-06   The SORTIN DD statement defines an input data set on 9-track tape with fixed blocked records, on volume 000101.

07-08   The SORTOUT DD statement defines an output data set with the same characteristics as the input data set, on volume 000102.

09-12   The SORTWK DD statements define four work tapes.

13   SORT statement. The FIELDS operand describes two control fields. The first control field begins on byte 1 of each record, is 6 bytes long, contains character (EBCDIC) data, and is to be sorted into ascending order. The second control field begins on byte 28 of each record, is 5 bytes long, contains character (EBCDIC) data, and is to be sorted into descending order. The file size is estimated at 10,000 records.

```
INPUT        Fixed-length blocked records on two unlabeled 9-track tape
             volumes.

OUTPUT       Fixed-length blocked records on one 9-track tape.

INTERMEDIATE STORAGE   Four 3400 9-track tapes.

USER ROUTINES   Four: two change record lengths, one changes control
                fields, one decides what to do if Nmax is exceeded.

OPTIONS      Estimated data set size; oscillating technique forced.

//EXAMP8     JOB   A402,PROGRAMMER
//STEP1      EXEC  SORT,PARM='OSCL'                                01
//SORTIN     DD    DSNAME=INPUT,VOL=SER=(000333,000343),          02
//                 UNIT=(2400,2),DISP=(OLD,DELETE),LABEL=(,NL),   03
//                 DCB=(RECFM=FB,LRECL=120,BLKSIZE=480)           04
//SORTOUT    DD    DSNAME=OUTPUT,UNIT=2400,DISP=(NEW,CATLG),      05
//                 VOL=SER=456,DCB=(RECFM=FB,LRECL=80,            06
//                 BLKSIZE=3200)                                  07
//SORTWK01   DD    UNIT=3400-3                                    08
//SORTWK02   DD    UNIT=3400-3                                    09
//SORTWK03   DD    UNIT=3400-3                                    10
//SORTWK04   DD    UNIT=3400-3                                    11
//MODLIB     DD    DSNAME=YOURRTNS,DISP=SHR                       12
//SORTMODS   DD    UNIT=3330,SPACE=(CYL,(1,,1))                   13
//SYSIN      DD    *                                              14
   SORT      FIELDS=(3,8,ZD,E,40,6,CH,D),FILSZ=E30000             15
   RECORD    TYPE=F,LENGTH=(120,100,80)                           16
   MODS      E15=(MODREC,784,MODLIB,N),                           17
      E16=(E16,1024,MODLIB,N),E35=(ADDUP,912,SYSIN),              18
      E61=(CHGE,1000,SYSIN)                                       19
   END                                                            20
Object deck for ADDUP routine                                     21
Object deck for CHGE routine                                      22
/*
```

Example 8. TAPE SORT, PROC=SORT, EXITS

01 Specifies the cataloged procedure SORT. OSCL in the PARM field directs the program to use the oscillating tape sequence distribution technique if it can, whether or not this technique appears to be the most efficient in this case.

02-04 Defines the input data set. The data set consists of fixed-length blocked records on two 9-track tape volumes; the UNIT parameter requests the system to provide two tape drives, one for each volume of the data set. Since the tape is unlabeled, DCB parameters must be supplied.

05-07 Defines the output data set, which also consists of fixed-length blocked records. It is on one 9-track tape.

08-11 Define four intermediate storage data sets on 3400-series tape units. Since the DSNAME parameter is omitted, the system will assign unique names to the data sets.

12 Describes a data set containing the load modules of the E15 and E16 user exit routines.

13 Defines a data set on which the ADDUP and CHGE routines specified in the MODS statement (lines 18 and 19) will be placed.

14 A data set follows in the input stream.

15 SORT statement. The FIELDS operand describes two control fields. The first will be changed by a user routine (at exit E61; see the MODS statement) before the program

places it into ascending order. The second control field will not be modified and will be placed in descending order.

16 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 user routine at the E35 exit again changes the length during the final merge phase, to 80 bytes each.

17-19 MODS statement. The statement describes four user routines. The first two are in a library that is defined on a job control statement with the ddname MODLIB; these two routines have the member names MODREC and E16, respectively. Neither routine requires additional link-editing. The next two routines are in object form in the input stream. Their names are ADDUP and CHGE, respectively. They must be link-edited together with other routines in their phases that require link-editing.

20 END statement. Required because of the user routines in the input stream.

21-22 Object decks in the input stream must be in numerical order of exit, so ADDUP (for E35) precedes CHGE (for E61).

```
INPUT        Blocked fixed-length records on 7-track unlabeled tape.

OUTPUT       Blocked fixed-length records on 7-track labeled tape.

INTERMEDIATE STORAGE   Six 7-track tapes.

USER ROUTINES   None.

OPTIONS      FORMAT=xx for control fields of like format; estimated data
             set size.

//EXAMP9     JOB   A402,PROGRAMMER
//STEP1      EXEC  SORT                                             01
//SORTIN     DD    DSNAME=INPUT,VOL=SER=000101,UNIT=2400-2,         02
//                 DCB=(DEN=2,RECFM=FB,LRECL=80,BLKSIZE=800,        03
//                 TRTCH=ET),DISP=(OLD,PASS),LABEL=(,NL)            04
//SORTOUT    DD    DSNAME=OUTPUT,UNIT=2400-2,DISP=(NEW,CATLG),      05
//                 VOL=SER=102,DCB=(DEN=2,TRTCH=ET)                 06
//SORTWK01   DD    UNIT=2400-2,LABEL=(,NL),DCB=(DEN=2,TRTCH=ET)     07
//SORTWK02   DD    UNIT=2400-2,LABEL=(,NL),DCB=(DEN=2,TRTCH=ET)     08
//SORTWK03   DD    UNIT=2400-2,LABEL=(,NL),DCB=(DEN=2,TRTCH=ET)     09
//SORTWK04   DD    UNIT=2400-2,LABEL=(,NL),DCB=(DEN=2,TRTCH=ET)     10
//SORTWK05   DD    UNIT=2400-2,LABEL=(,NL),DCB=(DEN=2,TRTCH=ET)     11
//SORTWK06   DD    UNIT=2400-2,LABEL=(,NL),DCB=(DEN=2,TRTCH=ET)     12
//SYSIN      DD    *
   SORT      FIELDS=(1,6,A,28,5,D),FORMAT=CH,FILSZ=E10000           13
/*
```

Example 9. TAPE SORT (7-TRACK), PROC=SORT

01 Invokes the SORT cataloged procedure. The SORTD procedure would be more efficient for this application, since there are no user routines that need link-editing, but SORT can also be used.

02-04 Defines the input data set named INPUT. It is on an unlabeled 7-track tape with serial number 000101. The DCB subparameters indicate that the tape was recorded at a density of 800 bpi (DEN=2), and is composed of fixed-length blocked records. TRTCH=ET indicates that the tape was recorded with even parity and that BCD 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 or only one of this unlabeled volume.

05-06 Defines the output data set named OUTPUT. It is recorded on 7-track tape on a volume with the serial number 102; and has the same characteristics as INPUT, except that the data set will be created in this job step and will be cataloged. The DCB subparameters not specified are the same as for SORTIN, by default.

07-12 Define intermediate storage for sort/merge. The storage is on six 7-track unlabeled tapes. These tapes are to be recorded with even parity and BCD to EBCDIC translation.

13 SORT statement. The FIELDS operand describes two fields. The first begins on byte 1 of each record, is 6 bytes long, contains character (EBCDIC) 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.

INPUT   Variable-length blocked records on 3350.

OUTPUT   Variable-length blocked records on 3350.

INTERMEDIATE STORAGE   One 3380 area of 3 cylinders.

USER ROUTINES   Initialization routine at the E11 exit and an NMAX error routine at E16.

OPTIONS   Message option (critical messages only); estimated data set size.

```
//EXAMP10  JOB   A402,PROGRAMMER
//STEPONE  EXEC  SORT,PARM='FLAG(U),LIST'                     01
//SORTIN   DD    UNIT=3350,DSNAME=PAY413,VOL=SER=335001,      02
//               DISP=(OLD,KEEP)                              03
//SORTOUT  DD    UNIT=3350,DSNAME=PAY414,VOL=SER=335004,      04
//               SPACE=(CYL,(15),RLSE),DISP=(NEW,KEEP)        05
//SORTWK01 DD    UNIT=3380,SPACE=(CYL,(6),,CONTIG)            06
//SORTMODS DD    UNIT=3330-1,SPACE=(TRK,(1,1,1))              07
//USERLIB  DD    DSNAME=JIMSMODS,DISP=SHR                     08
//SYSIN    DD    *
   SORT    FIELDS=(20,5,AQ,A),FILSZ=E17000                    09
   RECORD  TYPE=V,LENGTH=(,,,80,120)                          10
   ALTSEQ  CODE=(5BEA,7BEB,7CEC)                              11
   MODS    E11=(PREPMOD,504,SYSIN,S),E16=(MODMAX,554,         12
           USERLIB,N)                                         13
   END                                                        14
Object deck for PREPMOD
/*
```

<u>Example 10</u>. 3350 SORT, PROC=SORT, EXITS

01 Specifies the SORT cataloged procedure. The PARM options indicate that critical messages only are to be printed, and that program control statements are to be printed on SYSOUT.

02-03 The name of the input data set is PAY413, and it is on volume 335001 on a 3350. The data set is known to the operating system and is to be retained. The program will take the DCB parameters from the data set label. The records are variable-length, blocked.

04-05 The output data set is called PAY414, and will be on volume 335004 of a 3350. It is being created in this job step, and is to be retained. Data set DCB parameters will be the same as for SORTIN, by default. Unused space will be released.

06 One intermediate storage data set is defined on a 3380.

07 Defines an area to hold the PREPMOD module.

08 Defines a data set called JIMSMODS which contains the MODMAX user exit routine described on the MODS program control statement. The data set is known to the operating system and is not to be deleted after this job step.

09 SORT statement. The FIELDS operand describes one control field that begins on byte 16 of each record data area (not byte 20, since the record descriptor word takes 4 bytes), is 5 bytes long, contains character data which is to be collated according to the modified sequence described in the ALTSEQ statement (format is AQ), and is to be sorted into ascending sequence. The optional FILSZ operand indicates that the input data set contains approximately 17,000 records.

10 RECORD statement. Indicates that the input data set contains variable-length records with a minimum record length of 80 bytes, and an average length of 120 bytes. The RECORD statement is not required for this example, but without it, the program would assume a minimum record length of 24 bytes (large enough to contain the specified control field) and an average length equal to the average of maximum and minimum lengths.

11 ALTSEQ statement. Specifies that the three characters $, #, and @ are to collate in that order after Z.

12-13 MODS statement. Describes two user routines. The first, PREPMOD, will receive control at exit E11. It is 504 bytes long and can be link-edited separately. It is an object deck in the SYSIN input stream. The second routine, named MODMAX, will receive control at exit E16. It is 554 bytes long and the library in which it resides is described in the job control statement with the ddname USERLIB. It has been link-edited previously and requires no further link-editing prior to its use in this application.

14 END statement. Required because the PREPMOD object deck will follow it in SYSIN.

```
INPUT       Fixed-length blocked records on 3330.

OUTPUT      Fixed-length blocked records on 3340.

INTERMEDIATE STORAGE   None.

USER ROUTINES   One routine shortens the records as they leave the
                final merge phase.

OPTIONS     Exact data set size.

//EXAMP11   JOB    B600,PROGRAMMER
//STEP1     EXEC   PROC=SORT,PARM='SIZE(130000)'
//SORTIN    DD     DSNAME=INPUT,UNIT=3330,VOL=SER=333001,      01
//                 DISP=SHR
//SORTOUT   DD     DSNAME=OUTPUT,UNIT=3340,VOL=SER=334010,     02
//                 DCB=(RECFM=FB,LRECL=50,BLKSIZE=500),        03
//                 DISP=(NEW,KEEP),SPACE=(CYL,(1,1),RLSE)      04
//ERTNLIB   DD     DSN=EXITS,DISP=SHR                          05
//SYSIN     DD     *
   SORT     FIELDS=(10,5,CH,A),FILSZ=800                       06
   RECORD   TYPE=F,LENGTH=(,,50)                               07
   MODS     E35=(E35,534,ERTNLIB,N)                            08
/*
```

Example 11. SORT WITH NO SORTWK, PROC=SORT, 1 EXIT

No work areas are defined. If all records cannot be sorted in main storage, the program will terminate.

01 The input data set is named INPUT, is on a 3330 volume 333001, and consists of fixed-length records with a length of 80 bytes. The DCB information will be taken from the data set label.

02-04 The output data set, named OUTPUT, will be on volume 334010 of a 3340 and will contain fixed-length blocked records. One cylinder is requested for the data set; if the space is exhausted, additional cylinders are to be assigned one at a time. Unused space will be released. Records have been shortened at E35, so DCB information is different from SORTIN and therefore has to be specified.

05 Defines a library which contains the E35 routine.

06 SORT statement. The FIELDS operand describes one control field that begins on byte 10 of each record, is 5 bytes long, and contains character (EBCDIC) data; it is to be sorted into ascending order. The optional FILSZ operand indicates that the input data set contains exactly 800 records.

07 RECORD statement. Indicates that the input data set contains fixed-length records and that record length will be changed to 50 bytes as records leave the final merge.

08 MODS statement. Describes a user exit routine that will receive control at E35 exit. The name of the routine is E35; it is 534 bytes long, resides in the data set described in the ERTNLIB DD statement, and requires no further link-editing.

INPUT A concatenation of three data sets on 3330-1, 2400, and 3340.

OUTPUT Blocked fixed-length records on 9-track tape.

INTERMEDIATE STORAGE Two 3330 areas.

USER ROUTINES None.

OPTIONS FORMAT parameter for control fields of like format; estimated data set size.

```
//EXAMP12  JOB   A400,PROGRAMMER
//STEPT    EXEC  PGM=ICEMAN,REGION=128K                       01
//SYSOUT   DD    SYSOUT=A                                     02
//SORTLIB  DD    DSNAME=SYS1.SORTLIB,DISP=SHR                 03
//SORTIN   DD    DSNAME=INP1,DISP=OLD,UNIT=3330-1,            04
//               DCB=(RECFM=FB,BLKSIZE=7200,LRECL=80),        05
//               VOL=SER=XB0001                               06
//         DD    DSNAME=INP2,DISP=OLD,UNIT=2400,              07
//               DCB=(RECFM=FB,BLKSIZE=4000,LRECL=80),        08
//               VOL=SER=T33333                               09
//         DD    DSNAME=INP3,DISP=OLD,UNIT=3340,              10
//               DCB=(RECFM=FB,BLKSIZE=3600,LRECL=80),        11
//               VOL=SER=DISK01                               12
//SORTOUT  DD    DSNAME=OUTPUT,UNIT=3400-3,DISP=(NEW,CATLG),  13
//               VOL=SER=000102,DCB=(BLKSIZE=800)             14
//SYSIN    DD    *
   SORT    FIELDS=(1,6,A,28,5,D),FORMAT=CH                    15
   OPTION  FILSZ=E10000,DYNALLOC=(3330,2)                     16
```

Example 12. CONCATENATED INPUT, DYNAMICALLY ALLOCATED WORK AREAS

Example 12 differs from example 7 in three respects: the input is a concatenation of three input data sets on unlike devices; the region specified is 128K bytes; and work storage is dynamically allocated.

01 Indicates that a 128K bytes region is needed.

02 Sort messages are to be directed to system output class A.

03 Sort program modules are on SYS1.SORTLIB.

04-12 The SORTIN DD statement describes a concatenation of three input data sets on unlike devices.

The INP1 data set is on volume XB0001 of a 3330-1. It is known to the system, and consists of fixed-length blocked records with a record length of 80 and a block size of 7200. Note that this MUST be the largest block size of the data sets in the concatenation.

The INP2 data set is on a 9-track tape with serial number T33333. It is known to the system, and consists of fixed-length blocked records with a record length of 80 and a block size of 4000.

The INP3 data set is on a 3340 disk with the serial number DISK01. It is known to the system, and consists of fixed-length blocked records with a record length of 80 and a block size of 3600.

13-14 Block size is not the same for output as for input, and must therefore be specified.

15 SORT statement. The FIELDS operand describes two control fields. The first field begins on byte 1 of each record, is six bytes long, contains character (EBCDIC) data

(FORMAT=CH), and is to be sorted into ascending order. The second field begins on byte 28 of each record, is five bytes long, contains character (EBCDIC) data, and is to be sorted into descending order.

16 OPTION statement. Operands given on the OPTION statement override similar operands specified on a SORT control statement or at installation time. The FILSZ operand indicates that the input data set contains an estimated 10,000 records. The DYNALLOC operand indicates that two work data sets are to be dynamically allocated on 3330 (valid only when sort/merge is running under MVS).

```
INPUT        Fixed- or variable-length blocked records.

OUTPUT       Fixed- or variable-length blocked records.

INTERMEDIATE STORAGE   One 3330-1 area of 5 cylinders.

USER ROUTINES   None.

OPTIONS      Exact size file and alternate collating sequence for
             EBCDIC fields.

 //EXAMP13   JOB    A402,PROGRAMMER
 //SORT1     EXEC   PGM=MYPGM                               01
 //SYSOUT    DD     SYSOUT=A                                02
 //SYSPRINT  DD     SYSOUT=A                                03
 //SORTIN    DD     DSN=MY.INPUT.FILE,DISK=SHR              04
 //SORTWK01  DD     UNIT=3330-1,SPACE=(CYL,(5))             05
 //SORTOUT   DD     DSN=MY.OUTPUT.FILE,UNIT=3330-1,         06
 //                 SPACE(CYL,(3,2)),DISP=(NEW,CATLG)       07
 //SORTCNTL  DD     *                                       08
    OPTION   FILSZ=2270,CHALT                               09
 /*
```

Example 13. 3330-1 SORT USING SORTCNTL AND OPTION

01 Specifies the name of the program calling sort/merge.

02 Sort messages are to be directed to system output class A.

03 MYPGM output is to be directed to system output class A.

04 The SORTIN DD statement describes an input data set named MY.INPUT.FILE. The DISP parameter indicates that the data set is known to the operating system.

05 The SORTWK01 DD statement describes a work data set on a 3330-1. The area contains five cylinders.

06-07 The SORTOUT DD statement describes an output data set named MY.OUTPUT.FILE. The DISP parameter indicates that the data set is new and will be cataloged.

08 The SORTCNTL DD statement defines the data set that contains control statements used to modify the sort application.

09 OPTION statement. The file size is specified as exactly 2270 records and will override any size passed to sort in the program-provided parameter list. Both CH and AQ format record fields will be sorted as if they were AQ format.

MERGE EXAMPLES

```
INPUT       Blocked fixed-length records on four 9-track unlabeled tapes.

OUTPUT      Blocked fixed-length records on one 9-track tape.

INTERMEDIATE STORAGE   None required for a merge.

USER ROUTINES   None

OPTIONS     FORMAT=CH for control fields of like format; estimated data
            set size

//EXAMP14  JOB   A402,PROGRAMMER
//STEP1    EXEC  SORTD                                          01
//SORTIN01 DD    DSNAME-MERGIN01,VOL=SER-000111,DISP=OLD,       02
//               LABEL=(,NL),UNIT=3400-3,DCB=(RECFM=FB,         03
//               LRECL=80,BLKSIZE=240)                          04
//SORTIN02 DD    DSNAME=MERGIN02,VOL=SER=000222,DISP=OLD,       05
//               LABEL=(,NL),UNIT=3400-3,DCB=(RECFM=FB,         06
//               LRECL=80,BLKSIZE=240)                          07
//SORTIN03 DD    DSNAME=MERGIN03,VOL=SER=000333,DISP=OLD,       08
//               LABEL=(,NL),UNIT=3400-3,DCB=(RECFM=FB,         09
//               LRECL=80,BLKSIZE=240)                          10
//SORTIN04 DD    DSNAME=MERGIN04,VOL=SER=000444,DISP=OLD,       11
//               LABEL=(,NL),UNIT=3400-3,DCB=(RECFM=FB,         12
//               LRECL=80,BLKSIZE=240)                          13
//SORTOUT  DD    DSNAME=MERGOUT,VOL=SER=000101,DISP=(NEW,       14
//               KEEP),LABEL=(,NL),UNIT=2400                    15
//SYSIN    DD    *
   MERGE   FIELDS=(1,6,A,28,5,D),FORMAT=CH,FILSZ=E10000         16
/*
```

Example 14. MERGE FOUR UNLABELED TAPES, PROC=SORTD

01 The EXEC statement invokes the cataloged procedure SORTD.

02-13 The SORTINnn 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. Since they all have the same block size, the order in which they are specified is immaterial. Had they been different, the data set with the largest block size would have had to be specified first.

14-15 The result of the merge is recorded on 9-track tape at the same blocking factor and in the same format as the first input data set (SORTIN01), by default.

16 MERGE statement. The FIELDS operand describes two fields. The first begins on byte 1 of each record, is 6 bytes long, contains character (EBCDIC) 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. The input data sets contain a total of approximately 10,000 records.

```
INPUT       Variable-length blocked records on 3330.

OUTPUT      Variable-length blocked records on 3330.

INTERMEDIATE STORAGE   None.

USER ROUTINES   E35 (CALC) routine shortens records; E61 (MODRTN)
                routine modifies control fields.

OPTIONS     Exact input data set size.

//EXAMP15  JOB   A402,PROGRAMMER
//STEPONE  EXEC  SORT                                        01
//SORTIN01 DD    DSNAME=WEEKLY,VOL=SER=000101,UNIT=3330,     02
//               DISP=OLD,DCB=(RECFM=VB,LRECL=240,           03
//               BLKSIZE=4800)                               04
//SORTIN02 DD    DSNAME=DAILY,VOL=SER=000113,UNIT=3330,      05
//               DISP=(OLD,DELETE),DCB=(RECFM=VB,LRECL=240,  06
//               BLKSIZE=1200)                               07
//SORTOUT  DD    DSNAME=WEEKA,VOL=SER=000111,UNIT=3330,      08
//               DISP=(NEW,KEEP),SPACE=(TRK,(200,10)),       09
//               DCB=(RECFM=VB,LRECL=200,BLKSIZE=2000)       10
//USERLIB  DD    DSNAME=MYMODS,DISP=SHR                      11
//MODLIB   DD    DSNAME=XYZ,DISP=SHR                         12
//SYSIN    DD    *
   MERGE   FIELDS=(5,6,CH,E),FILSZ=8150                      13
   RECORD  TYPE=V,LENGTH=(,,200)                             14
   MODS    E35=(CALC,800,USERLIB),E61=(MODRTN,456,MODLIB,N)  15
/*
```

Example 15. MERGE TWO 3330 FILES; PROC=SORT, EXITS

02 Calls the SORT cataloged procedure.

02-04 The first of two input data sets for the merge. The data set, named WEEKLY, is on a 3330 disk with the volume serial number 000101. 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 block size of 4800.

05-07 The second input data set, which is named DAILY, is on a 3330 disk unit, with the volume serial number 000113. It is old, will be deleted after this job step, and contains records of the same format and length as the WEEKLY data set; the block size is smaller.

08-10 The output from the merge will be a data set named WEEKA. It is new and will be retained in the system on a 3330 disk with the serial number 000111. The data set will be recorded on 200 tracks. If this space is not sufficient, additional space will be allotted in blocks of ten tracks. The data set will consist of variable-length blocked records with a maximum record length of 200 (see 1 on the RECORD statement) and a block size of 2000.

11 The library on which the CALC routine for exit E35 resides.

12 The library on which the E61 (MODRTN) routine resides.

13 MERGE statement. The FIELDS operand describes one control field that will be modified (by the routine at exit E61 specified in the MODS statement) before it is examined by the merge. The start of the control field is given as byte 5; note that this points to the first byte of the record data itself, since for a variable-length record the first four bytes are occupied by the record descriptor word. The field is six bytes long. The exact size of the input data sets is given.

14 RECORD statement. Records in the input data sets are variable length. A modification routine (at exit E35) makes the maximum record length in the output data set 200 bytes.

15 MODS statement. A routine named CALC receives control at exit E35. It is approximately 800 bytes long, resides in the library defined on the job control statement with the DDname USERLIB, and must be link-edited together with other routines in its phase which require link-editing. At exit E61, the program transfers control to a routine from the library defined by the job control statement with the ddname MODLIB. The member name of this routine is MODRTN. It is 456 bytes long and does not need further link-editing.

```
INPUT       Blocked fixed-length records on three 7-track tapes.

OUTPUT      Blocked fixed-length records on one 7-track tape.

INTERMEDIATE STORAGE   None.

USER ROUTINES   None.

OPTIONS     Estimated input data set size.

//EXAMP16   JOB   A714,PROGRAMMER
//STEPA     EXEC  SORTD                                           01
//SORTIN01 DD     DSNAME=FILE1,VOL=SER=000123,UNIT=2400-2,        02
//                DCB=(DEN=2,TRTCH=ET),DISP=(OLD,DELETE)          03
//SORTIN02 DD     DSNAME=FILE2,VOL=SER=000225,UNIT=2400-2,       04
//                DCB=(DEN=2,TRTCH=ET),DISP=(OLD,DELETE)          05
//SORTIN03 DD     DSNAME=FILE3,VOL=SER=000179,UNIT=2400-2,        06
//                DCB=(DEN=2,TRTCH=ET),DISP=(OLD,DELETE)          07
//SORTOUT  DD     DSNAME=FILE123,VOL=SER=000111,UNIT=2400-2,      08
//                DCB=(DEN=2,TRTCH=ET),DISP=(NEW,KEEP)            09
//SYSIN    DD     *
   MERGE   FIELDS=(1,6,A,28,5,D),FORMAT=CH,FILSZ=E10000           10
/*
```

Example 16. MERGE THREE 7-TRACK TAPES, PROC=SORTD

01 Since there are no user routines, it is more efficient to use the SORTD cataloged procedure.

02-07 The three input data sets to the merge are all on 7-track standard-label tape (DCB information will be taken from the data set labels). TRTCH=ET indicates that the tape was recorded with even parity and that BCD to EBCDIC translation is required. SORTIN01 must have the greatest block size of the three inputs.

08-09 The output data set is also to be recorded on 7-track tape, and is to have the same characteristics as the first input data set, by default. It is to be kept.

10 MERGE statement. Describes two control fields. The first begins at byte 1, is six bytes long, and is to be collated in ascending sequence; the second is five bytes long, beginning on the 28th byte. Both are in EBCDIC character format, so the FORMAT option is used. The file size is estimated at 10,000 records.

APPENDIX E. EBCDIC AND ASCII COLLATING SEQUENCES

EBCDIC

The following table shows the collating sequence for EBCDIC character and unsigned decimal data. The collating sequence ranges from low (00000000) to high (11111111). The bit configurations which do not correspond to symbols (that is, 0 through 73, 81 through 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 are collated algebraically, that is, each quantity is interpreted as having a sign.

Collating Sequence	Bit Configuration	Symbol	Meaning
0	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	;	Semicolon
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	

Collating Sequence	Bit Configuration	Symbol	Collating Sequence	Bit Configuration	Symbol
134	10000110	f	.		
135	10000111	g	.		
136	10001000	h	209	11010001	J
137	10001001	i	210	11010010	K
.			211	11010011	L
.			212	11010100	M
145	10010001	j	213	11010101	N
146	10010010	k	214	11010110	O
147	10010011	l	215	11010111	P
148	10010100	m	216	11011000	Q
149	10010101	n	217	11011001	R
150	10010110	o	.		
151	10010111	p	.		
152	10011000	q	226	11100010	S
153	10011001	r	227	11100011	T
.			228	11100100	U
.			229	11100101	V
162	10100010	s	230	11100010	W
163	10100011	t	231	11100111	X
164	10100100	u	232	11101000	Y
165	10100101	v	233	11101001	Z
166	10100110	w	.		
167	10100111	x	.		
168	10101000	y	240	11110000	0
169	10101001	z	241	11110001	1
.			242	11110010	2
.			243	11110011	3
193	11000001	A	244	11110100	4
194	11000010	B	245	11110101	5
195	11000011	C	246	11110110	6
196	11000100	D	247	11110111	7
197	11000101	E	248	11111000	8
198	11000110	F	249	11111001	9
199	11000111	G	.		
200	11001000	H	.		
201	11001001	I	255	11111111	

ASCII

The following table shows the collating sequence for ASCII, character, and unsigned decimal data. The collating sequence ranges from low (00000000) to high (01111111). Bit configurations which do not correspond to symbols are not shown.

Packed decimal, zoned decimal, fixed-point normalized floating-point data, and the signed numeric data formats are collated algebraically; that is, each quantity is interpreted as having a sign.

Collating Sequence	Bit Configuration	Symbol	Meaning
0	00000000		Null
32	00100000	SP	Space
33	00100001	|	Logical OR
34	00100010	"	Quotation mark
35	00100011	#	Number sign
36	00100100	$	Dollar sign
37	00100101	%	Percent
38	00100110	&	Ampersand
39	00100111	'	Apostrophe, prime
40	00101000	(	Opening parenthesis
41	00101001	)	Closing parenthesis
42	00101010	*	Asterisk
43	00101011	+	Plus
44	00101100	,	Comma
45	00101101	-	Hyphen, minus
46	00101110	.	Period, decimal point
47	00101111	/	Slant
48	00110000	0	
49	00110001	1	
50	00110010	2	
51	00110011	3	
52	00110100	4	
53	00110101	5	
54	00110110	6	
55	00110111	7	
56	00111000	8	
57	00111001	9	
58	00111010	:	Colon
59	00111011	;	Semicolon
60	00111100	<	Less than
61	00111101	=	Equals
62	00111110	>	Greater than
63	00111111	?	Question mark
64	01000000	@	Commercial At
65	01000001	A	
66	01000010	B	
67	01000011	C	
68	01000100	D	
69	01000101	E	
70	01000110	F	
71	01000111	G	
72	01001000	H	
73	01001001	I	
74	01001010	J	
75	01001011	K	
76	01001100	L	
77	01001101	M	
78	01001110	N	

Collating Sequence	Bit Configuration	Symbol	Meaning
79	01001111	O	
80	01010000	P	
81	01010001	Q	
82	01010010	R	
83	01010011	S	
84	01010100	T	
85	01010101	U	
86	01010110	V	
87	01010111	W	
88	01011000	X	
89	01011001	Y	
90	01011010	Z	
91	01011011	[	Opening bracket
92	01011100	\	Reverse slant
93	01011101	]	Closing bracket
94	01011110	∧	Circumflex, Logical NOT
95	01011111	_	Underscore
96	01100000	`	Grave Accent
97	01100001	a	
98	01100010	b	
99	01100011	c	
100	01100100	d	
101	01100101	e	
102	01100110	f	
103	01100111	g	
104	01101000	h	
105	01101001	i	
106	01101010	j	
107	01101011	k	
108	01101100	l	
109	01101101	m	
110	01101110	n	
111	01101111	o	
112	01110000	p	
113	01110001	q	
114	01110010	r	
115	01110011	s	
116	01110100	t	
117	01110101	u	
118	01110110	v	
119	01110111	w	
120	01111000	x	
121	01111001	y	
122	01111010	z	
123	01111011	{	Opening Brace
124	01111100	|	Vertical Line
125	01111101	}	Closing Brace
126	01111110	~	Tilde

APPENDIX F. TIMING ESTIMATES

The tables in this appendix contain estimated total execution times for some sorting applications using the OS/VS Sort/Merge Program Product 5740-SM1 program. They are given for planning purposes only and could, therefore, deviate from similar actual runs.

The figures given for elapsed time (in seconds) are for sorting both fixed and variable-length records using the FLR- and VLR-Blockset sorting techniques. No figures are provided for merges.

Timing estimates are given for the 3350, 3375, and 3380 Direct Access Storage Devices. In addition, the last table shows multiplication factors for calculating the timing estimates for jobs run with processors other than the IBM 3031 Processor.

INPUT/OUTPUT BLOCKING

Input and output records were blocked and the block size used was 4000 on the average. The average record length was 500 bytes. If your own block sizes or record lengths are different, your results will, of course, vary from these.

INTERPOLATION/EXTRAPOLATION OF ELAPSED TIME

Interpolations can reasonably be made for main storage availability and for data set size.

Extrapolation for bigger data sets than are included in the tables can be performed. Bear in mind, though, that extrapolation will not give the same degree of accuracy as interpolation.

ASSUMPTIONS MADE IN PRODUCING ESTIMATES

All figures assume that the sort is not being multiprogrammed, that is, no other task is using the processor or input/output devices. It is also assumed that I/O operations are error-free. Jobs were run under a VS2/MVS Release 3.8-level system. Other assumptions are described below.

The control statements for the timing estimate applications are shown in Figure 25.

```
//xxx JOB ...
//xxx EXEC PGM=ICEMAN,PARM='SIZE=(xxx)',REGION=xxx
//xxx DD statements
  SORT FIELDS=(6,4,CH,A,15,6,CH,A),FILSZ=xxx
/*
```

Figure 25. Control Statements for Timing Estimate Applications

RECORDS AND CONTROL FIELDS

It is assumed that:

- The required sequence may be ascending (A) or descending (D).
- There are two control fields, which are EBCDIC characters or binary (on a byte boundary), up to 10 bytes long. Control fields of any other format, number, or length might increase elapsed time.
- No user exit routines are to be activated.

MAIN STORAGE

The figures shown under "Main Storage" (in the tables below) correspond to the SIZE parameter specified on the EXEC statement. The MVS region used was approximately 50K bytes larger.

It is assumed that:

- The sort is running in a region or partition in virtual mode.
- The number of real pages is equal to the virtual region or partition size, so that no time for page transfers is allowed.

DEVICES USED

The SORTIN and SORTOUT files and one SORTWK file reside on 3380 disk devices. The SORTWK file resides in one work area on a volume different from those used for SORTIN and SORTOUT.

To obtain estimates for the 3350 Direct Access Storage Device, use the figures given for the 3380 devices in the following tables and increase them by 45%.

To obtain estimates for the 3375 Direct Access Storage Device, use the figures given for the 3380 devices and increase them by 25%.

TABLES SHOWING ESTIMATED TOTAL EXECUTION TIMES IN SECONDS

IBM 3031 Processor using FLR-Blockset for sorting fixed-length records

File Size in MB	1	4	9	14	25	50	75
Main Storage							
60K	12	45	97	148	257	498	-
230K	8	30	65	98	171	331	490
400K	6	20	43	66	114	221	325

IBM 3031 Processor using VLR-Blockset for sorting variable-length records

File Size in MB	1	4	9	14	25	50	75
Main Storage							
60K	26	89	184	273	460	859	-
230K	14	47	96	143	241	451	650
400K	7	25	51	75	127	236	340

TIMING ESTIMATES FOR OTHER PROCESSORS

To obtain timing estimates for other processors (in seconds), multiply the entries in the above tables by the appropriate factor from the table below.

Processor	**VLR-Blockset**	**FLR-Blockset**
IBM System/370 Model 158	1.074	1.069
IBM System/370 Model 168	.833	.845
IBM 3032 Processor	.822	.835
IBM 3033 Processor	.781	.797
IBM 4341 Processor	1.132	1.123

INDEX

OS/VS Sort/Merge
Programmer's Guide
SC33-4035-7

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ____________________

Previous TNL ____________________

Previous TNL ____________________

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

SC33-4035-7

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape | Please do not staple | Fold and tape

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

SC33-4035-7

OS/VS Sort/Merge Programmer's Guide (File No. S370-33 (OS/VS)) Printed in U.S.A. SC33-4035-7

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601