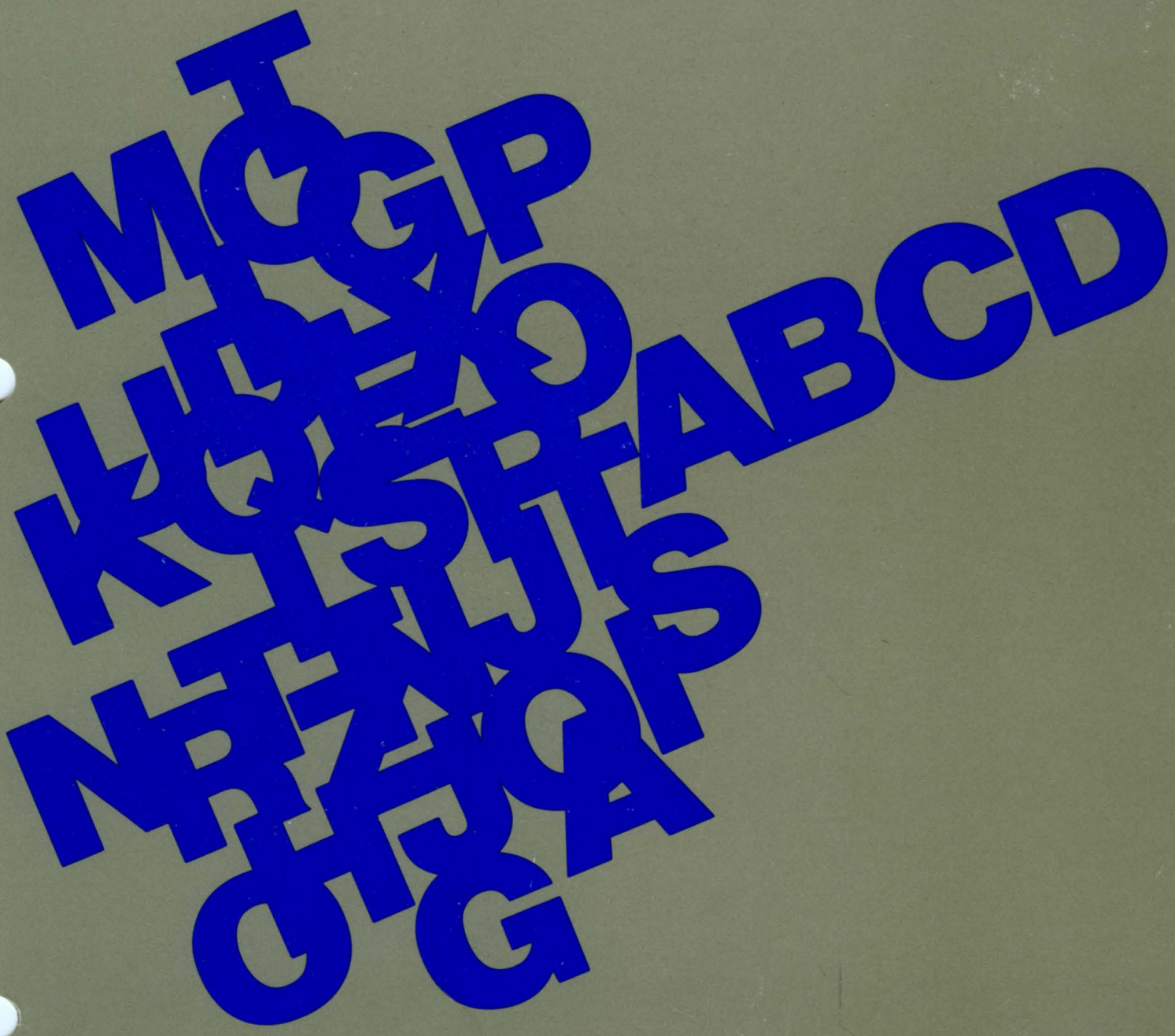
IBM

DFSORT Application Programming: Guide

Program Product



Order Number:
SC33-4035-11

Program Number
5740-SM1

Release Number
8.0

IBM

DFSORT Application Programming: Guide

Program Product

Order Number:
SC33-4035-11

Program Number:
5740-SM1

Release Number:
8.0

Twelfth Edition (March 1986)

This is a major revision of, and makes obsolete, SC33-4035-10.

This edition applies to Release 8.0 of IBM DFSORT, 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 made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

A form for readers' 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 whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This manual is for programmers who want to sort, merge, or copy files using DFSORT (Data Facility Sort) Program Product No. 5740-SM1.

Hereafter, the term "processing" refers to DFSORT's sorting, merging and copying capability.

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 process files. You will also be able to link your own routines to DFSORT to perform such services as altering, deleting, or inserting records as they are being processed.

If you are a new DFSORT user, consider reading the tutorial, *Getting Started with DFSORT*, before you begin using this manual. It provides an introduction to using the product, including sample sorting applications.

Organization of Manual

This manual contains the following sections:

- Chapter 1, "Introduction" on page 1 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. It also describes how to write a simple DFSORT program.
- Chapter 2, "Program Control Statements" on page 17 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 personal routines that you will use during program execution.
- Chapter 3, "Job Control Statements" on page 113 shows you how and what job control statements you must write to execute your DFSORT program.
- Chapter 4, "User Exit Routines" on page 135 describes how to insert a routine of your own into the DFSORT program, via program exits.
- Chapter 5, "Invoking DFSORT from an Assembler Program" on page 187 describes how to initiate execution of the program from within your own program, using a system macro instruction.

- Chapter 6, "Improving Program Efficiency" on page 207 gives advice on how you can achieve faster processing.
- Appendix A, "Sample Job Streams" on page 225 provides sample job streams for sort, merge, and copy applications.
- Appendix B, "Calculating Storage Requirements" on page 277 discusses the storage devices used for intermediate storage, the factors determining the amount of intermediate storage required for a DFSORT program, and the program's method of selecting a sorting technique.
- Appendix C, "Converting to the Extended Parameter List" on page 283 shows how to covert programs that use a 24-bit parameter list to use an extended parameter list.
- Appendix D, "Specification/Override of DFSORT Options" on page 285 shows the order of override when the same or corresponding options are specified in different sources.
- Appendix E, "Data Format Examples" on page 299 gives examples of the assembled data formats used with IBM System 360/370.
- Appendix F, "EBCDIC and ISCII/ASCII Collating Sequences" on page 303 lists the collating sequences from low to high order for EBCDIC and ISCII/ASCII characters.
- Appendix G, "SMF Record (TYPE 16)" on page 311 lists the SMF record produced by DFSORT.
- Appendix H, "DFSORT Messages and Codes," lists, explains, and suggests responses to all the error messages produced by this DFSORT program.

Reading List

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

For All Applications

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

MVS/Extended Architecture JCL, GC28-1148

OS/VS1 JCL Reference, GC24-5099

OS/VS2 MVS JCL, GC28-0692

For an explanation of SMF record type 16, which enables an installation to collect statistics for auditing its DFSORT activities, generating utilization reports, developing tuning information, and so forth, see:

MVS/Extended Architecture System Programming Library: System Management Facilities (SMF), GC28-1153

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 installation time and estimates of storage required by the program, consult:

DFSORT Planning and Installation, SC33-4034

For an overall discussion of DFSORT features, see:

DFSORT General Information, GC33-4033

For quick reference, see:

DFSORT Reference Summary, SX33-8001

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

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

For a primer on how to use DFSORT, see:

Getting Started with DFSORT, SC26-4109

For information on diagnosing DFSORT failures, see:

DFSORT Diagnosis Guide, SY26-3971

Planning Checkpoint/Restart

Complete information on the checkpoint/restart facility is contained in:

MVS/Extended Architecture Checkpoint/Restart User's Guide , GC26-4012

MVS/370 Checkpoint/Restart User's Guide, GC26-4054

OS/VS1 Checkpoint/Restart User's Guide, GC26-3876

OS/VS2 MVS Checkpoint/Restart User's Guide, GC26-3877

COBOL and PL/I Users

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

Assembler Language Users

Assembler H Version 2 Application Programming: Language Reference, GC26-4037

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

Program Initiation with System Macro Instructions

MVS/Extended Architecture System Programming Library: Supervisor Services and Macro Instructions, GC28-1154

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

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

Data Management

MVS/Extended Architecture Data Administration Guide, GC26-4013

MVS/Extended Architecture Data Administration: Macro Instruction Reference, GC26-4014

MVS/Extended Architecture System–Data Administration, GC26-4010

MVS/370 Data Administration Guide, GC26-4058

MVS/370 Data Administration: Macro Instruction Reference, GC26-4057

MVS/370 System–Data Administration, GC26-4056

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

OS/VS1 Data Management Macro Instructions, GC26-3872

OS/VS1 Data Management Services Guide, GC26-3874

OS/VS2 MVS Data Management Macro Instructions, GC26-3873

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

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

Dynamic Allocation

MVS/Extended Architecture System Programming Library: System Modifications, GC28-1152

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

ISCII/ASCII

MVS/Extended Architecture Data Administration: Macro Instruction Reference, GC26-4014

MVS/370 Data Administration: Macro Instruction Reference, GC26-4057

OS/VS1 Data Management Macro Instructions, GC26-3872

OS/VS2 MVS Data Management Macro Instructions, GC26-3873

ISO/ANSI Tape Labels

MVS/Extended Architecture Magnetic Tape Labels and File Structure Administration, GC26-4003

MVS/370 Magnetic Tape Labels and File Structure Administration, GC26-4064

OS/VS Tape Labels, GC26-3795

VSAM Users

MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference, GC26-4019

MVS/Extended Architecture VSAM Administration Guide, GC26-4015

MVS/Extended Architecture VSAM Administration: Macro Instruction Reference, GC26-4016

MVS/Extended Architecture VSAM Catalog Administration: Access Method Services Reference, GC26-4075

MVS/370 Integrated Catalog Administration: Access Method Services Reference, GC26-4051

MVS/370 VSAM Administration Guide, GC26-4066

MVS/370 VSAM Administration: Macro Instruction Reference, GC26-4074

MVS/370 VSAM Catalog Administration: Access Method Services Reference, GC26-4059

OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications, GC26-3819

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

MVS/Extended Architecture Data Facility Product: Planning Guide, GC26-4040

MVS/370 Data Facility Product: Planning Guide, GC26-4052

Planning for Enhanced VSAM under OS/VS, GC26-3842

For debugging aids, see

MVS/Extended Architecture Debugging Handbook, Volume 1, LC28-1164, Volume 2, LC28-1165, Volume 3, LC28-1166, Volume 4, LC28-1167, Volume 5, LC26-1168, or all volumes, LBOF-1015

OS/VS1 Debugging Guide, GC24-5093

OS/VS2 MVS System Programming Library: Debugging Handbook, Volume 1, GC28-0708, and Volume 2, GC28-0709

Summary of Amendments

Release 8.0, March 1986

New features added to DFSORT include:

- An enhancement to the variable-length record sorting technique, VLR-Blockset, which improves performance when sorting variable length records. On MVS/Extended Architecture (MVS/XA) systems, utilization of extended addressing capability is available with VLR-Blockset.
- More efficient use of processor cache memory, which improves performance when sorting fixed length records.
- The copy function, which copies a data set without performing any sorting or merging operation. It can be used with most of the same control statements, exits, and options available when sorting or merging.
- An enhancement to the Blockset technique, which can now be used to continue sorting when encountering a record too short to contain all specified control fields. New installation and execution options have been added for ease of use. In addition, the Blockset technique can now be used for VSAM input and output data sets.

In addition, the figures in Appendix D have been enhanced to show which options can be used with a particular function (sort, merge, or copy).

Release 7.1, June 1985

New enhancements added to DFSORT are the ability to:

- Preserve the original order of identically collating records when doing a Blockset merge, if the EQUALS option is used.
- Specify, using the EQUALS option, that the first record will be retained when summarizing identically collating records when doing a Blockset sort or merge.
- Use the Blockset technique when merging spanned variable-length records.

Features that were removed from prior releases and that are now being reimplemented:

- Processing of multivolume SORTOUT data sets with the EXCP access method rather than with the BSAM access method, whenever possible.
- Dynamic link-editing of user exit routines. Note that the SORT cataloged procedure has been changed to include link-edit DD statements.

  Note also that the SORT cataloged procedure has been changed to contain DD statements for dynamic link-editing.
- Writing program messages to the master console.

Release 7.0, January 1985

New features added to DFSORT are:

- For MVS/XA, the ability to reduce the processor time for sorting done in large storages by using IBM System/370 Extended Architecture Sorting Instructions.
- Extended addressing for MVS/XA:
  - Improved performance because certain buffers and modules can now be placed above 16-megabyte virtual when sorting fixed-length records.
  - The ability to specify:
    - The upper limit of the amount of main storage above and below 16-megabyte virtual available to DFSORT (TMAXLIM).
    - The number of bytes reserved above 16-megabyte virtual for system use (ARESALL).
    - The number of bytes reserved above 16-megabyte virtual for a program that invokes DFSORT (ARESINV).
- COBOL-related enhancements:

  - The ability to invoke DFSORT from VS COBOL II programs.
  - The ability to use the VS COBOL II FASTSRT compile-time option, which enhances performance.
  - The ability to write E15 and E35 exit routines in COBOL.
  - The ability to specify an alternative message data set when invoking DFSORT with JCL. This is especially useful when exits are written in COBOL.

- Support of the 3480 Magnetic Tape Subsystem.
- Removal of the upper limit of 48K bytes for the RESALL option.
- The ability to specify the maximum number of records to be accepted for sorting.

Service changes have also been added.

Contents

Figures

Chapter 1. Introduction

This chapter describes the relationship of the IBM OS/VS DFSORT Program Product 5740-SM1 (hereafter referred to as DFSORT) 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

DFSORT 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 *Supervisor Services and Macro Instructions*.)

Because DFSORT 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.

The operating systems supported by this release are:

- OS/VS1 Release 7
- OS/VS2 MVS Release 3.8
- MVS/Extended Architecture (MVS/XA)
- MVS/370 (OS/VS2 MVS with MVS/370 Data Facility Product installed)

DFSORT also executes on these systems under VM or VM/XA Migration Aid.

Throughout this manual, the term MVS is used to refer to OS/VS2, MVS, MVS/XA, and MVS/370, unless otherwise indicated.

Operation in MVS/XA Mode

Programs that invoke DFSORT, in addition to user exit routines, will be able to reside above or below 16-megabyte virtual, execute in 24-bit or 31-bit mode, and pass data that resides above or below 16-megabyte virtual to DFSORT.

For MVS/XA users who install DFSORT resident, most of the Blockset modules will be placed above 16-megabyte virtual in the extended link pack area. This provides more space in the link pack area for resident programs that cannot reside above 16-megabyte virtual.

Furthermore, when doing a Blockset sort, DFSORT can place selected buffers above 16-megabyte virtual, leaving more space below 16-megabyte virtual for user applications.

In addition, more and larger buffers provide greater optimization opportunities.

Use of IBM System/370 Extended Architecture Sorting Instructions (hereafter referred to as System/370-XA Sorting Instructions), which are part of the extended architecture hardware, reduces the processor time for sorting done in large storage areas.

What the Program Will Do

DFSORT has three 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.
- To copy records, you do not need to sort or merge the records first.

  You can copy your data sets, without sorting or merging them, using the COPY parameter on the OPTION control statement. See the "OPTION Control Statement" on page 72 for more detailed information on the syntax of this parameter. You can also use COPY as a value on the FIELDS parameter of the SORT or MERGE control statement. The original order of your input records will be retained.

  The copy function cannot be used with BDAM data sets.

  Dynamic link editing of exits is not allowed with copy.

Note: Copy uses only the Blockset technique.

Depending on various conditions, DFSORT selects among the following sorting and merging techniques:

- Disk work data set sorting techniques: Blockset (for fixed- and variable-length records), Peerage (for fixed-length records), and Vale (for both fixed- and variable-length records)
- Various tape work data set sorting techniques
- Merge only techniques: Blockset and conventional

These are discussed under "Using Efficient Sort/Merge Techniques" on page 211 and "Merging Techniques" on page 212.

Input and Output Data Sets

Sort and Copy Applications

Input to the sort or copy may be a blocked or unblocked QSAM or VSAM data set containing fixed- or variable-length records. QSAM input data sets may be concatenated even if they are on unlike devices, provided the conditions described in "SORTIN DD Statement" on page 127 are met.

Output from the sort or copy may be a blocked or unblocked QSAM or VSAM data set, regardless of whether the input is QSAM or VSAM, but must be of the same type (fixed or variable) as the input data set.

Merge Applications

Input to the merge may be up to 16 blocked or unblocked QSAM or VSAM data sets containing fixed- or variable-length records. The input data sets may be either QSAM or VSAM, but not both. The records in the input data sets must already be sorted into the order required for output. For further details, see "SORTINnn DD Statement" on page 128.

Output from the merge may be a blocked or unblocked QSAM or VSAM data set, regardless of whether the input is QSAM or VSAM, but must be of the same type (fixed or variable) as the input data set.

General Notes and Limitations

The input and output data sets must be on devices that can be used with QSAM or VSAM.

The length of the records that DFSORT can handle depends on the amount of main storage available. The length of a record can never exceed the maximum record length specified by the user. The maximum record length for variable-length records is 32756 bytes; for fixed-length records, it is 32760 bytes.

For spanned records, maximum lengths may be smaller. Conditions such as control fields of different formats, large number of control fields, or a large number of

intermediate 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 data sets is 18 bytes; the minimum record length is 14 bytes.

QSAM Data Set Notes and Limitations

If you use DSN=NULLFILE on your DD statement for an input data set, DFSORT cannot use the EXCP access method (this is a system restriction).

Input data sets can be empty.

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

ISO/ANSI Version 1 tape files cannot be used as output; they can only be used as input.

VSAM Data Set Notes and Limitations

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

The same data set must not be specified for both input and output.

A data set used for input or output must have been previously defined. A data set used for input must not be empty. If the data set is empty, VSAM returns an input error code (160) and DFSORT terminates.

Data sets cannot be contatenated.

If output is a keyed-sequential data set (KSDS), the key must be the major control field (or the key fields must be in the same ascending order as the major control field). VSAM does not allow you to store records with duplicate primary keys.

Any VSAM exit function available for input data sets may be used except EODAD. See the description of E18 use with VSAM in Chapter 4.

The VSAM exit list must be built using the VSAM EXLST macro instruction giving the address of your routines that handle VSAM exit functions.

When processing variable-length records with VSAM input and non-VSAM output, the SORTOUT LRECL must be 4 more than the maximum record size defined in the cluster. Variable-length records have a record descriptor word (RDW) field of 4 bytes at the beginning of each record, but VSAM records do not. Therefore, the record size defined in the cluster is 4 bytes less than the non-VSAM LRECL. DFSORT adds 4 bytes for the RDW when processing the record. These are removed if VSAM is used for both input and output.

For example,

```
Maximum record size in VSAM cluster      128 : up to 128 bytes of data

LRECL for variable-length record         128 : 4 bytes RDW and up to
                                               124 bytes of data

                                         132 : 4 bytes RDW and up to
                                               128 bytes of data
```

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 or merges them in ascending or descending order (according to which order you have specified).

All other fields you specify are called minor fields. Conceptually, if the major fields in two records are equal, DFSORT sorts or merges 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 with identical control fields is preserved on output if the EQUALS option is in effect.

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 composed of four control fields is shown in Figure 1 on page 6.

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

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

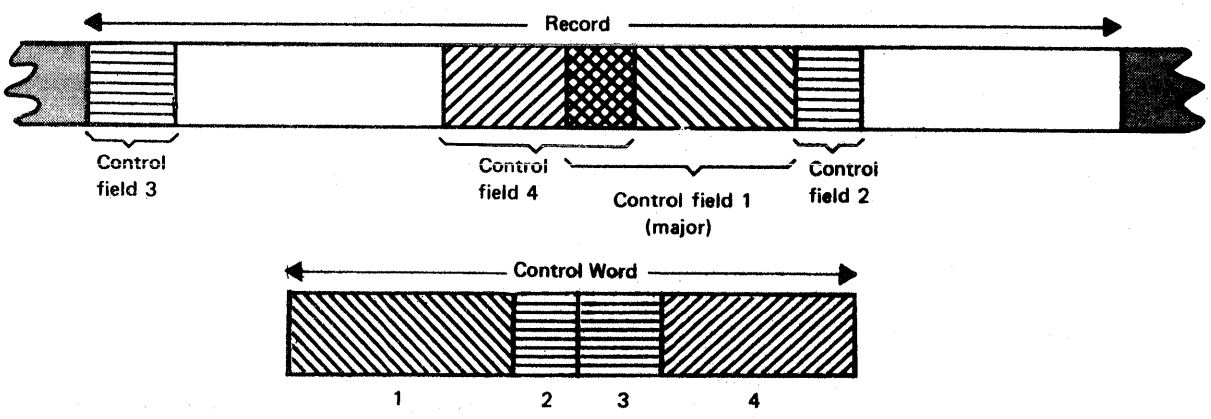


Figure 1. Control Fields

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.

Installation Options

Some of the DFSORT default values depend on the specifications made by your system programmer, through the ICEMAC macro, when DFSORT was installed. DFSORT installation is described in *DFSORT Planning and Installation Guide.*

The following is a summary of the DFSORT installation parameters and functions that may be set when the program is installed.

Parameters	Function
ALTSEQ	Alters the usual EBCDIC collating sequence.
ARESALL	Specifies, for MVS/Extended Architecture (MVS/XA), the number of bytes reserved above 16-megabyte virtual for system use.
ARESINV	Specifies, for MVS/Extended Architecture (MVS/XA), the number of bytes reserved above 16-megabyte virtual for the invoking program when DFSORT is dynamically invoked.
CHALT	Specifies whether character format fields should be translated using ALTSEQ.

CHECK Specifies whether record count should be checked for applications that use the E35 user exit routine without a SORTOUT data set.

COBEXIT Indicates whether E15 and E35 routines written in COBOL will be executed with the VS COBOL II library.

DYNALOC Specifies the default values for device name and number of work data sets to be dynamically allocated on MVS systems when DYNALLOC is specified at execution time (on either the SORT or OPTION statement) without these values.

EQUALS Specifies whether the input order of equally collating records should be preserved for output.

ERET Specifies the action to be taken if DFSORT encounters a critical error.

EXCPVR Specifies for OS/VS1 users whether the EXCPVR SVC can be used for SORTWKnn I/O.

IGNCKPT Specifies whether the checkpoint/restart facility is to be ignored if it is requested at execution time and the Blockset technique (which does not support the checkpoint/restart facility) can be used.

INV | JCL Indicates whether the defaults specified are to be used when DFSORT is JCL invoked or dynamically invoked.

LIST Specifies whether program control statements are to be printed.

MAXLIM Sets an upper limit to the amount of main storage available to DFSORT below 16-megabyte virtual, when SIZE/MAINSIZE=MAX.

MINLIM Sets a lower limit to the amount of main storage available to DFSORT below 16-megabyte virtual.

MSGCON Specifies the class of program messages to be written to the master console.

MSGDDN Specifies an alternate name for the message data set.

MSGPRT Specifies the class of program messages to be printed on the message data set.

OUTREL Specifies whether unused temporary SORTOUT data set space is to be released.

OUTSEC Specifies whether DFSORT should use automatic secondary allocation for SORTOUT data sets that are temporary or new.

OVERRGN Specifies the amount of main storage above the REGION value available to Blockset.

RESALL Reserves storage for system and application use.

RESDNTx Indicates for OS/VS1 whether DFSORT modules reside in the pageable supervisor area. The *x* is **B** for Blockset modules and **P** for Peerage and Vale modules.

RESINV Reserves space for programs invoking DFSORT.

SIZE Sets amount of main storage available to DFSORT.

SMF Specifies whether System Management Facilities (SMF) records are to be produced. See Appendix G for a description of the SMF record produced by DFSORT.

STIMER Specifies whether DFSORT should use the STIMER macro. If DFSORT does not use the STIMER macro, processor timing data does not appear in SMF records.

SVC Specifies a user SVC number for DFSORT.

TMAXLIM Specifies, for MVS/XA, an upper limit to the total amount of main storage above and below 16-megabyte virtual available to DFSORT when SIZE/MAINSIZE=MAX.

VERIFY Specifies whether the sequence of output records is to be verified.

VIO Specifies for MVS users whether virtual allocations for sort work areas should be dynamically reallocated to a real disk location.

VLSHRT Specifies whether to continue sorting or merging if a variable length input record is found that is not long enough to contain all specified control fields. VLSHRT does not apply to FLR processing.

WRKREL Specifies whether unused temporary SORTWKnn data set space is to be released.

WRKSEC Specifies whether DFSORT should use automatic secondary allocation for temporary SORTWKnn data sets.

Tables showing all the possible sources of specification and order of override for each option are in Appendix D.

Machine Requirements

DFSORT is designed to operate with all of the IBM processors supported by MVS/370, MVS/XA, or OS/VS1 and:

- Any device that is supported by MVS/370, MVS/XA, or OS/VS1 for program residence
- Any device that is used by QSAM or VSAM for input or output

Intermediate storage requirements are given in "Intermediate Storage" on page 275.

System/370-XA Sorting Instructions must be activated to be used.

Main Storage Requirements

DFSORT main storage is defined when the program is installed. If this is not appropriate, you can determine the requirements for your particular application and override the default value at execution time. To establish your requirements, see "Main Storage" on page 275 and "Tuning Main Storage" on page 207.

Program Execution

To execute DFSORT, you must prepare two types of statements: program control statements and JCL statements. Program control statements are processed by DFSORT; they describe your records and how you want them processed. A full discussion of the program control statements is contained in Chapter 2, "Program Control Statements" on page 17. JCL statements are processed by the operating system control program; they describe to the operating system the data sets required by the program, and may be used to initiate execution of DFSORT. A complete description of the format and of the specifications for the JCL statements required by the program is contained in Chapter 3, "Job Control Statements" on page 113.

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 JCL data definition statements—unless you use the DYNALLOC facility under MVS. The formulas for determining space requirements are described in "Intermediate Storage" on page 275.

Neither a merge nor a copy requires intermediate storage.

Application programmers must be aware of the interaction between control statements and user exits in terms of record handling.

Figure 2 on page 10 shows a simplified picture of the sequence of processing for record handling exits, statements, and options.

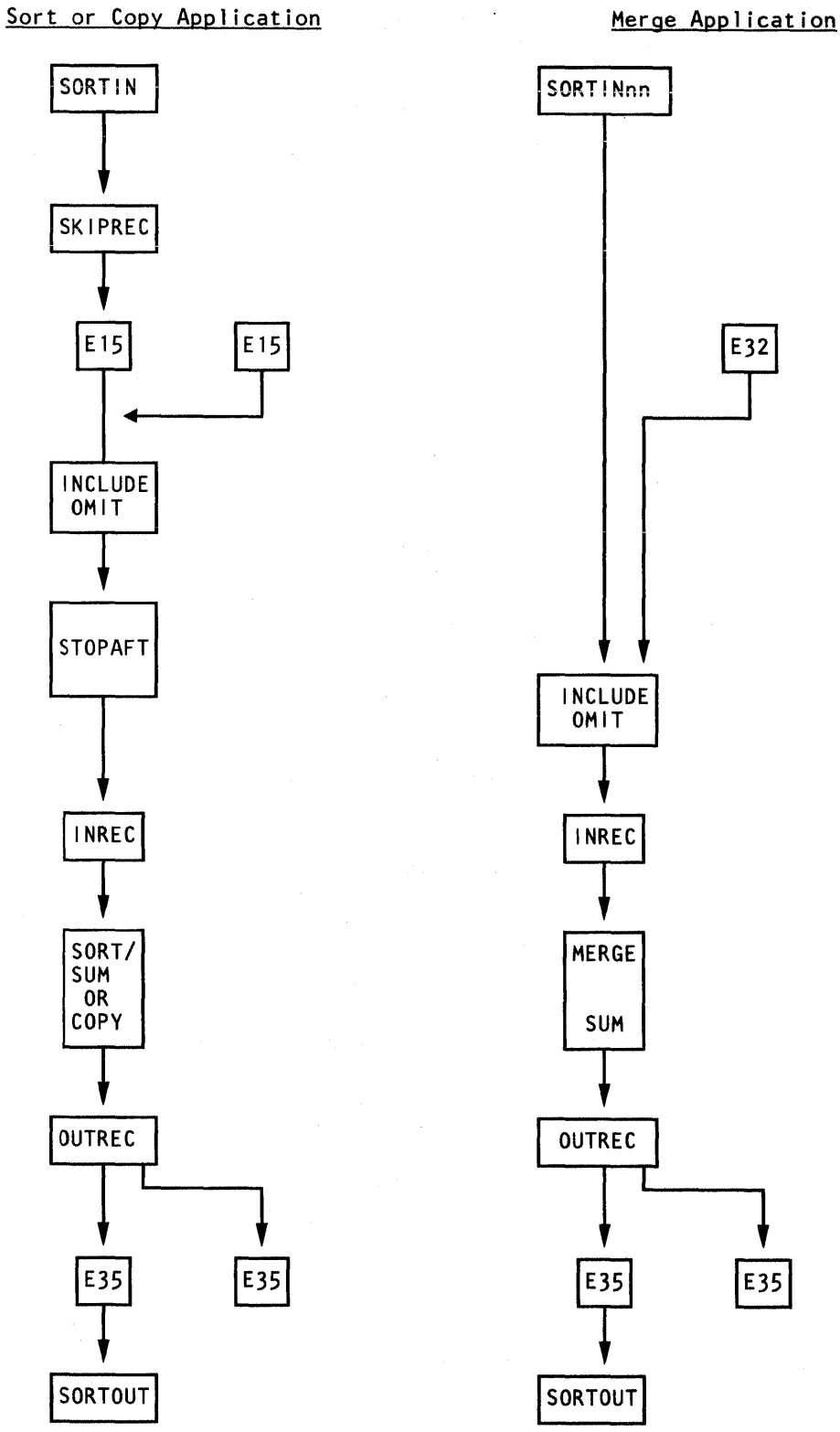


Figure 2. Record Processing Sequence

Figure 2 illustrates the following:

- Records are first read in from the SORTIN data set (if present) for a sort or copy application or the SORTINnn data sets (if present) for a merge application. If SORTIN is not present for a sort or copy application, all records must be inserted by an E15 exit. (This is also the case if DFSORT is invoked from a program with the address of an E15 exit in the parameter list, because SORTIN will be ignored.) A COBOL E15 routine may be used if the E15 exit is specified in the MODS statement. If SORTINnn is not present for a merge application, all records must be inserted by an E32 exit.

- Record processing is done if the SKIPREC option is in effect (sort or copy application with SORTIN present only). SKIPREC is processed before the E15 (if any) is called. Records are deleted until the SKIPREC count is satisfied, thus eliminating them before sort or copy processing, resulting in better performance.

- If an E15 routine is present (sort or copy application with SORTIN present only), it is given control next. A COBOL E15 routine may be used if the E15 exit is specified in the MODS statement. The E15 routine may insert, delete, or reformat records.

- Record processing is done next for an INCLUDE or OMIT statement, if it is present. If any changes have been made in the record format by an E15, the INCLUDE/OMIT field definitions must apply to the current format rather than to the original format. You can cause records to be deleted by your selection criteria, thus eliminating them before sort, merge, or copy processing, resulting in better performance.

- For a sort or copy, if the STOPAFT option is in effect, record processing for it is done next. Input will stop after a user-specified maximum number (n) of records has been accepted. Records are accepted under the following conditions:

  - read from SORTIN or inserted by E15
  - not deleted by SKIPREC
  - not deleted by E15
  - not deleted by INCLUDE/OMIT

- If an INREC statement is present, it is processed next. If any changes have been made in the record format, the INREC field definitions must apply to the current format rather than to the original format. INREC reformats the records. Note that INREC can be used to achieve better performance, by shortening records before they are processed.

- Record processing for the SORT, MERGE, or OPTION COPY statement is done next.

  For a sort, *all* input records are processed *before* any output records is processed. For a copy or merge, an output record is processed after an input record is processed. For a sort or merge, if a SUM statement is present, it is processed during the SORT or MERGE processing. Records are summarized

and duplicates deleted as soon as possible for better performance. If any changes have been made in the record format, the SORT or MERGE, and SUM field definitions must apply to the current format rather than to the original format.

- If an OUTREC statement is present, it is processed next. If any changes have been made in the record format, the OUTREC field definitions must apply to the current format rather than to the original format. OUTREC reformats the records.

- If an E35 exit is present, it is given control after all of the statement processing is complete. If any changes have been made in the record format, the E35 exit receives the records in the current format rather than in the original format. A COBOL E35 routine may be used if the E35 exit is specified in the MODS statement. The E35 exit may add, delete, or reformat records. If a SORTOUT data set is not present, the E35 exit must dispose of all the records (DFSORT views these records as deleted). (This is also the case if sort, merge, or copy is invoked with the address of an E35 exit in the parameter list, because SORTOUT will be ignored.)

- Finally, the records are written to the SORTOUT data set, if present.

Program Initiation

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

- 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 Chapter 3, "Job Control Statements" on page 113.

- In a program written in basic assembler language with a system macro instruction, as described in Chapter 5, "Invoking DFSORT from an Assembler Program" on page 187.

- 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.

Throughout this manual, a DFSORT program initiated by an EXEC statement is referred to as *JCL invoked*; a DFSORT program initiated from another program, written in assembler, COBOL, or PL/I, is referred to as *dynamically invoked.*

Program Modification

During execution, DFSORT 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 routines to summarize, insert, delete, shorten, or otherwise alter records as they are being processed. 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.

Your routines must reside in private libraries.

The program exits and their uses are explained in Chapter 4, "User Exit Routines" on page 135.

Messages

You can determine whether the DFSORT messages should be printed and/or written to the master console.

DFSORT can write three types of messages to the message data set: critical error messages, informational messages, and diagnostic messages. You can specify at installation or execution time which types of messages you want to be written.

DFSORT can write two types of messages to the master console: critical error messages and informational messages. You can specify at installation time which types of messages you want to be written.

Messages are discussed in detail in Appendix H.

Return Codes

DFSORT 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 a user abend is issued, depending on what you have requested. If the message data set is required, but is not provided, a return code of 20 is returned. See "Return Codes" on page 314.

Checkpoint/Restart

Checkpoint/Restart is a facility of the operating system that permits an automatic or deferred restart if a DFSORT sort or merge application abnormally terminates. You must specify certain parameters in the program control statements and prepare a JCL DD statement if you want to include this facility in a DFSORT execution (see Chapter 2, "Program Control Statements" on page 17).

No checkpoints are taken:

- If no work data set is specified.
- For a copy application.
- If an invoked merge is handling output through exit E35.
- If output from a merge application is to be a VSAM data set.
- If the output file for a merge application takes up less than one volume.

- If, for a merge application, you supply the address of your own exit list for the SORTOUT DCB at exit E39.
- If the Blockset technique is selected.
- Within a user exit routine. This includes SORT/MERGE input and output procedures with an invoking COBOL program.

Notes:

1. *Checkpoint/Restart does not apply to the copy function.*
2. *The Blockset technique does not support checkpoint/restart. If the Blockset technique is chosen, checkpoint/restart will be ignored. However, if necessary, the Blockset technique can be bypassed so that checkpoints can be taken, by specifying either IGNCKPT=NO on the ICEMAC installation macro or NOBLKSET on the OPTION statement.*

Also note that no ANSI Standard Tape label files can be *open* during checkpoint/restart.

If you want checkpoints taken, you must use the facility provided by DFSORT. You cannot use the system checkpoint at End of Volume.

For more information on the checkpoint/restart facility, see the list of books at the front of this manual under "Planning Checkpoint Restart".

Statistical Data Collection

If you want to collect statistics on execution time, record distribution, and so forth, you can use the SMF installation option. SMF is a parameter operand of the ICEMAC installation macro. Users who have properly installed a modified DFSORT SVC routine have this option available to them.

If SMF is specified, DFSORT causes an SMF record to be written for each sort, merge, or copy application 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 parameter on the ICEMAC installation option. A full SMF record will only be produced by DFSORT if requested (SMF=FULL), and only if the processing operation is for variable-length records.

By specifying STIMER=YES on the ICEMAC installation macro, you can have processor time data included in SMF records. This option can be overridden at execution time by specifying NOSTIMER on the OPTION statement if your exit(s) take checkpoints. Note, however, that the installation option STIMER=NO cannot be overridden at execution time.

For more information on statistical data collection, see Appendix H, "DFSORT Messages and Codes" on page 313, and *System Management Facilities (SMF).*

Maximum Efficiency

The objective of DFSORT is to process data as fast 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, and optimization takes place in two ways:

- Optimal values are calculated for many variables, such as buffer sizes.
- The most efficient technique is selected automatically.

The specifications you make in your program control and JCL statements affect program execution, efficiency, and speed. Suggestions for improving the performance of a DFSORT application are given in Chapter 6, "Improving Program Efficiency" on page 205.

Control Statement Example

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

```
//EXAMP    JOB  A402,PROGRAMMER,REGION=512K                          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
//SYSIN     DD  *                                                    08
   SORT   FIELDS=(5,12,CH,A),FILSZ=E2000                            09
/*
```

Line	Explanation
01	The JOB statement introduces this job to the operating system, and specifies a region of 512K 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 is recorded on a 9-track tape and is kept. The data set is 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 This DD statement defines a temporary work data set. The data set is on a SYSDA direct access device. Ten cylinders are specified for the data set.

08 A data set follows in the input stream.

09 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.

Chapter 2. Program Control Statements

Before DFSORT can operate on the input data, it must receive program control statements. Control statements can be received from the following sources:

- SYSIN data set
- SORTCNTL data set
- 24-bit parameter list
- Extended parameter list

Each of these sources is discussed in detail in later chapters.

Some control statements are always required, whereas others are optional and are required only for specific actions. The control statements describe:

- The type of operation to be performed
- Control field parameters
- Modifications to be made by your own routines
- Functions to be invoked
- Input and output files
- Options selected for particular applications

The program checks the validity of each statement before processing it. If the program finds an error, it issues a message. (See Appendix H, "DFSORT Messages and Codes" on page 311, for descriptions of these messages.)

Control Statement Summary

Control Statements

There are 13 control statements:

Statement	Function
ALTSEQ	Specifies modifications to the IBM EBCDIC collating sequence. The modified sequence is used for any control field whose format is specified as AQ.
DEBUG	For use when diagnostic information is required for debugging.
END	Causes DFSORT to discontinue reading SYSIN or SORTCNTL.
INCLUDE	Specifies that only records whose fields meet certain criteria are included.
INREC	Specifies how records are reformatted before they are processed.
MERGE	Provides information about control fields. Use this statement if your application is a merge (or copy).
MODS	This statement is required only when you include user routines in a DFSORT application. A description of how to write such routines and how they may be used in a DFSORT application is in Chapter 4, "User Exit Routines" on page 135.
OMIT	Specifies that records whose fields meet certain criteria are deleted.
OPTION	Provides overrides for installation defaults (such as EQUALS, CHALT, and CHECK), and optional information (such as DYNALLOC, SKIPREC, and COPY).
OUTREC	Specifies how records are reformatted before they are written.
RECORD	Provides record length and type information. This statement is required when you include user exit routines that change record lengths during DFSORT 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.
SORT	Provides information about control fields. Use this statement if your application is a sort (or copy).
SUM	Specifies that summary fields in records with equal control fields are summarized in one of the records, and that the other records are deleted.

An overview of the format and parameters of all the program control statements is given in Figure 3 on page 20.

Comment Statements

Comment statements are specified by placing an '*' in column 1. They are printed with the other control statements, but not otherwise processed.

Note: Comment statements are only allowed in the SYSIN and SORTCNTL data sets.

Notational Conventions

A uniform system of notation describes the format of the DFSORT control statements. This notation is not part of the language; it merely provides a basis for describing the structure of the commands.

The command-format illustrations in this chapter 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, and so forth) must be entered as shown.

Operation	Parameters
ALTSEQ	CODE=(fftt...,fftt) (See Appendix D for functions to which these parameters apply.)

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").

Operation	Parameters
DEBUG	[ABEND | NOABEND] [,ABSTP] [,BSAM] [,BUFFERS={ANY | BELOW}] [,CTRx=n] [,FMTABEND] [,NOASSIST] (See Appendix D for functions to which these parameters apply.)

Parameter	Explanation	Notes
ABEND NOABEND	An unsuccessful run is to: -Terminate with ABEND. -Terminate with return code of 16.	Overrides the ERET option specified at installation time.
ABSTP	During Blockset processing, forces an ABEND for an unsuccessful run preventing the loss of needed information in the dump.	Overrides ERET, ABEND, and NOABEND options.
BSAM	BSAM access method is used instead of EXCP.	

Figure 3 (Part 1 of 18). Control Statement Summary

Parameter	Explanation	Notes
BUFFERS	Specifies whether the buffers may be placed above or below 16-megabyte virtual.	
CTRx=n	Prints a formatted dump when the input or output count equals n.	
FMTABEND	Prints a formatted dump when DFSORT abends.	
NOASSIST	System/370-XA sorting instructions are not to be used.	

Operation	Parameters
END	None. The END statement causes DFSORT to discontinue reading SYSIN or SORTCNTL.

Operation	Parameters
INCLUDE	{COND=(p1,m1,f1,{EQ | NE | GT | GE | LT | LE} ,{p2,m2,f2 | constant}[{,AND | ,OR},...]) | COND=(p1,m1,{EQ | NE | GT | GE | LT | LE}, {p2,m2 | constant}[{,AND | ,OR},...])FORMAT=f} (See Appendix D for functions to which these parameters apply.)

Figure 3 (Part 2 of 18). Control Statement Summary

Parameter	Explanation	Notes
COND=	Describes the relational condition.	
p	Position within record.	
m	Length.	
f	Format.	Permissible formats are: CH – EBCDIC character, unsigned ZD – Zoned decimal, signed PD – Packed decimal, signed FI – Fixed-point binary, signed BI – Binary, unsigned AC – ISCII/ASCII character, unsigned CSL – EBCDIC numeric, leading separate sign CST – EBCDIC numeric, trailing separate sign CLO – EBCDIC numeric, leading overpunch sign CTO – EBCDIC numeric, trailing overpunch sign ASL – ISCII/ASCII numeric, leading separate sign AST – ISCII/ASCII numeric, trailing separate sign AQ – EBCDIC character, alternate collating sequence
EQ	Equal to.	
NE	Not equal to.	
GT	Greater than.	
GE	Greater than or equal to.	
LT	Less than.	
LE	Less than or equal to.	
Constant		Constant can be decimal, character, or hexadecimal.
AND	Logical AND.	The sign & may be used instead of the word AND.
OR	Logical OR.	The sign | may be used instead of the word OR.
FORMAT=f	Optional; may be used when all INCLUDE field data formats are the same.	The permissible values for f are listed above.

Figure 3 (Part 3 of 18). Control Statement Summary

Operation	Parameters
INREC	FIELDS=([s,]p,m[,a]...[,s][,p,m[,a]][,s]) (See Appendix D for functions to which these parameters apply.)

Parameter	Explanation	Notes
FIELDS=	Specifies the order of input and separation fields in the reformatted input record.	
p	Position within record of input field.	
m	Length of input field.	
a	Alignment of the input field in the reformatted input record.	Permissible values are: H – Halfword aligned. F – Fullword aligned. D – Doubleword aligned.
s	Separation field.	Permissible values are: nX – Bland separation. n bytes of EBCDIC blanks (X'40') are inserted (n=1-256). nZ – Binary zero separation. n bytes of binary zeros (X'00') are inserted (n=1–256).

Operation	Parameters
MERGE	{FIELDS=(p,m,f,s...,p,m,f,s) | FIELDS=(p,m,s...,p,m,s),FORMAT=f | FIELDS=COPY} [,FILES=n] [,CKPT] [,EQUALS | ,NOEQUALS] [,FILSZ=x | ,SIZE=y] (See Appendix D for functions to which these parameters apply.)

Figure 3 (Part 4 of 18). Control Statement Summary

Parameter	Explanation	Notes
FIELDS=		See explanation and notes for this parameter on the SORT statement.
COPY		See explanation and notes for this parameter on the OPTION statement.
FORMAT=f		See explanation and notes for this parameter on the SORT statement.
FILES=n	Optional. Specifies the number of input files for a merge when input is supplied through the E32 exit.	
CKPT		See explanation and notes for this parameter on the OPTION statement.
EQUALS NOEQUALS		See explanation and notes for this parameter on the OPTION statement.
FILSZ=x SIZE=y		See explanation and notes for this parameter on the OPTION statement.

Operation	Parameters
MODS	exit=(n,m,s[,e])..., exit=(n,m,s[,e])..., (See Appendix D for functions to which these parameters apply.)

Parameter	Explanation	Notes
exit=	The name of an exit to be activated.	Must be a valid exit name (for example, E15, E61). You may specify any exit, except E32.
n	The name of your routine, or member name of routine in a library.	
m	Size, in bytes, used by the routine.	This includes the size of the module and storage obtained by your routine.
s	Location of the routine.	The ddname of the data set containing the routine.

Figure 3 (Part 5 of 18). Control Statement Summary

Parameter	Explanation	Notes
e	Linkage editor requirements of your routine, and whether your routine is written in COBOL.	Permissible types are: • N — No link-editing required (default if e is not specified). • C — E15 or E35 exit written in COBOL and no link-editing required. • T — Routine must be link-edited together with other routines for the same phase. • S — E11 or E31 routine requires link-editing, but it must be link-edited separately from other routines.

Operation	Parameters
OMIT	{COND=(p1,m1,f1,{EQ | NE | GT | GE | LT | LE} ,{p2,m2,f2 | constant}[{,AND | ,OR},...]) | COND=(p1,m1,{EQ | NE | GT | GE | LT | LE}, {p2,m2 | constant}[{,AND | ,OR},...]),FORMAT=f} (See Appendix D for functions to which these parameters apply.)

Parameter	Explanation	Notes
COND=	Describes the relational condition.	
P	Position within record.	
m	Length.	
f	Format.	Permissible formats are: CH – EBCDIC character, unsigned ZD – Zoned decimal, signed PD – Packed decimal, signed FI – Fixed-point binary, signed BI – Binary, unsigned AC – ISCII/ASCII character, unsigned CSL – EBCDIC numeric, leading separate sign CST – EBCDIC numeric, trailing separate sign CLO – EBCDIC numeric, leading overpunch sign CTO – EBCDIC numeric, trailing overpunch sign ASL – ISCII/ASCII numeric, leading separate sign AST – ISCII/ASCII numeric, trailing separate sign AQ – EBCDIC character, alternate collating sequence
EQ	Equal to.	

Figure 3 (Part 6 of 18). Control Statement Summary

Parameter	Explanation	Notes
NE	Not equal to.	
GT	Greater than.	
GE	Greater than or equal to.	
LT	Less than.	
LE	Less than or equal to.	
constant		Constant can be decimal, character, or hexadecimal.
AND	Logical AND.	The sign & may be used instead of the word AND.
OR	Logical OR.	The sign | may be used instead of the word OR.
FORMAT=f	Optional; may be used when all OMIT field data formats are the same.	The permissible values for f are listed above.

Figure 3 (Part 7 of 18). Control Statement Summary

Operation	Parameters
OPTION	[ARESALL={n | nK}] [,ARESINV={n | nK}] [,CHALT | ,NOCHALT] [,CHECK | ,NOCHECK] [,CKPT] [,COBEXIT={COB1 | COB2}] [,COPY] [,DYNALLOC[={d | (d) | (,n) | (d,n)}]] [,EQUALS | ,NOEQUALS] [,FILSZ=x | ,SIZE=y | ,FILSZ=En | ,SIZE=En] [,LIST | ,NOLIST] [,MAINSIZE={n | nK | MAX}] [,MSGDDN=ddname] [,MSGPRT={ALL | NONE | CRITICAL}] [,NOBLKSET] [,NOOUTREL] [,NOOUTSEC] [,NOSTIMER] [,NOWRKREL] [,NOWRKSEC] [,RESALL={n | nK}] [,RESINV={n | nK}] [,SKIPREC=z] [,SORTDD=cccc] [,SORTIN=ddname] [,SORTOUT=ddname] [,STOPAFT=n] [,VERIFY | ,NOVERIFY] [,VLSHRT | ,NOVLSHRT] (See Appendix D for functions to which these parameters apply.)

Figure 3 (Part 8 of 18). Control Statement Summary

Parameter	Explanation	Notes
ARESALL=	Optional. Reserves storage (above 16-megabyte virtual) for system and application use.	Applicable to only MVS/XA.
n		Limit: 8 digits.
nK		Limit: 5 digits; K=1024.
ARESINV=	Optional. Reserves storage (above 16-megabyte virtual) for invoking programs.	Applicable to only MVS/XA when DFSORT is dynamically invoked.
n		Limit: 8 digits.
nK		Limit: 5 digits; K=1024.
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).
CHECK NOCHECK	Optional. Specifies whether record counters should be checked at end of program execution.	Applicable only for applications with output record processing in an E35 exit routine.
CKPT	Optional. Checkpoints are taken.	CHKPT is also accepted. This parameter is ignored if a Blockset technique is selected, unless IGNCKPT=NO was specified at installation time. Checkpoints cannot be taken during a merge operation with VSAM output or during an invoked merge handling output through E35.
COBEXIT=	Optional. Specifies COBOL libraries for E15 and E35.	
COB1	OS/VS COBOL library or no library.	
COB2	VS COBOL II library.	

Figure 3 (Part 9 of 18). Control Statement Summary

Parameter	Explanation	Notes
COPY	Optional. Specifies a data set is to be copied.	Regardless of input form, output can be either QSAM or VSAM. Output must be of the same type (fixed or variable) as input.
DYNALLOC	Optional. Dynamic allocation of intermediate work storage.	Applicable to only MVS.
d	Device type.	d can be any of 2314, 3330, 3330–1, 3340, 3350, 3375, 3380, 3400–3, 3400–4, 3480, 3850, 2400, 2400–3, 2400–4, or their user–assigned group name, such as SYSDA.
n	Number of devices (work data sets).	Number of work data sets (up to 16).
EQUALS NOEQUALS	Optional. Specifies whether order of equally collating records should be preserved from input to output.	
FILSZ=x SIZE=y FILSZ=En SIZE=En	Optional. The number of records to be sorted or merged.	If n is an estimate, the value must be preceded by the character E (FILSZ=En). If SIZE is used instead of FILSZ, the value should represent the number of records in the input file(s).
LIST NOLIST	Optional. Specifies whether control statements will be listed.	Will be processed only if passed in an extended parameter list. Overridden by a SORTDIAG DD statement.
MAINSIZE=	Optional. Specifies main storage size.	
n		Limit: 8 digits.
nK		Limit: 5 digits; K=1024.
MAX	Allocates amount of storage specified by MAXLIM or TMAXLIM installation option.	
MSGDDN= ddname	Optional. Specifies message data set ddname.	Applicable only when supplied in an extended parameter list.

Figure 3 (Part 10 of 18). Control Statement Summary

Parameter	Explanation	Notes
MSGPRT=	Optional. Specifies message type.	Applicable only when supplied in an extended parameter list. Messages are written to the message data set. Overridden by a SORTDIAG DD statement.
ALL	All Messages except diagnostic messages (ICE800I–ICE999I) will be printed.	Control statements will be printed only if LIST is in effect.
NONE	Messages and control statements will not be printed.	
CRITICAL	Critical (error) messages only will be printed.	Control statements will be printed only if LIST is in effect. *Note:* Control statements will not be printed if MSGPRT=CRITICAL is in effect and DFSORT is dynamically invoked using the 24-bit parameter list.
NOBLKSET	Optional. Specifies bypass of Blockset technique.	The higher performance Blockset technique will be used whenever possible. You can bypass the Blockset technique (for example, if you want to use checkpoint/restart) by specifying this parameter.
NOOUTREL	Optional. Specifies that unused temporary SORTOUT data set space is not to be released.	
NOOUTSEC	Optional. Specifies that automatic secondary allocation for temporary or new SORTOUT data sets should not be used.	
NOSTIMER	Optional. Specifies that DFSORT should not use the STIMER macro.	

Figure 3 (Part 11 of 18). Control Statement Summary

Parameter	Explanation	Notes
NOWRKREL	Optional. Specifies that unused temporary SORTWKnn data set space is not to be released.	
NOWRKSEC	Optional. Specifies that automatic secondary allocation should not be used for SORTWKnn data sets.	
RESALL= n nK	Optional. Reserves storage below 16-megabyte virtual for system and application use.	Applicable only when SIZE/MAINSIZE=MAX is in effect. Must be 4096 or greater (limited to 8 digits). Must be 4 or greater (limited to 5 digits). K=1024.
RESINV= n nK	Optional. Reserves storage below 16-megabyte virtual for invoking programs, when SIZE/MAINSIZE=MAX.	Applicable only when DFSORT has been dynamically invoked, and SIZE/MAINSIZE=MAX is in effect. Limit: 8 digits. Limit: 5 digits; K=1024.
SKIPREC=z	Optional. The program will skip z records at the beginning of the input data set.	
SORTDD= cccc	Optional. Specifies prefix for certain "SORT" ddnames.	Applicable only when supplied in an extended parameter list. A four character prefix. The first character is alphabetic; the next three characters are alphameric or national.
SORTIN= ddname	Optional. Specifies ddname to be associated with the input data set.	Applicable only when supplied in an extended parameter list. Not affected by SORTDD keyword. SORTIN is the default, unless SORTDD=cccc is specified, in which case ccccIN is the default.

Figure 3 (Part 12 of 18). Control Statement Summary

Parameter	Explanation	Notes
SORTOUT= ddname	Optional. Specifies ddname to be associated with the output data set.	Applicable only when supplied in an extended parameter list. Not affected by SORTDD keyword. SORTOUT is the default, unless SORTDD=cccc is specified, in which case ccccOUT is the default.
STOPAFT=n	Optional. Indicates the number of records to be accepted for sorting or copying (read from SORTIN or inserted by E15 and not deleted by SKIPREC, E15, or INCLUDE/OMIT).	
VERIFY NOVERIFY	Optional. Specifies whether sequence checking on final output record sequence should be done.	
VLSHRT NOVLSHRT	Optional. Specifies whether to continue processing variable length records if one is too short to contain all specified control fields.	VLSHRT is not used if INCLUDE/OMIT, INREC, SUM, or OUTREC is specified.

Operation	Parameters
OUTREC	FIELDS=([s,]p,m[,a]...[,s][,p,m[,a]][,s]) (See Appendix D for functions to which these parameters apply.)

Parameter	Explanation	Notes
FIELDS=	Specifies the order of input and separation fields in the reformatted output record.	
p	Position within record of input field.	
m	Length of input field.	

Figure 3 (Part 13 of 18). Control Statement Summary

Parameter	Explanation	Notes
a	Alignment of the input field in the reformatted output record.	Permissible values are: H – Halfword aligned. F – Fullword aligned. D – Doubleword aligned.
s	Separation field.	Permissible values are: nX – Blank separation. n bytes of EBCDIC blanks (X'40') are inserted (n=1–256). nZ – Binary zero separation. n bytes of binary zeros (X'00') are inserted (n=1–256).

Figure 3 (Part 14 of 18). Control Statement Summary

Operation	Parameters
RECORD	[TYPE=x][,LENGTH=(L1,L2,L3,L4,L5,L6,L7)] (See Appendix D for functions to which these parameters apply.)

Parameter	Explanation	Notes
TYPE=x	Used when all records are supplied via E15 or E32 or when VSAM data sets are used for input.	x must be: F–(fixed length), V–(variable length EBCDIC), or D–(variable length ASCII).
LENGTH=	Describes fixed-length records.	
L1	Used when no SORTIN DD statement is supplied. L1=SORTIN LRECL.[1]	
L2	Used when length is changed at E15. L2=length after E15.	
L3	Used when SORTOUT LRECL[1] is not equal to SORTIN and no SORTOUT LRECL[1] is available. L3=SORTOUT LRECL.[1]	
LENGTH=	Describes variable-length records.	
L1	Used when no SORTIN DD statement supplied. L1=maximum record length; otherwise, overridden to default.	If processing records with VSAM input and non-VSAM output, add 4 to the input record length.[2]
L2	Used when length changed at E15. L2=length after E15.	If the E15 is reading VSAM input, add 4 bytes to the input record length.
L3	Used when SORTOUT LRECL is not equal to SORTIN and SORTOUT LRECL is not available. L3=SORTOUT LRECL.	If the E15 is reading VSAM input, add 4 bytes to the input record length.
L4	Minimum length	If the E15 is reading VSAM input, add 4 bytes to the input record length.
L5	Average length.	If the E15 is reading VSAM input, add 4 bytes to the input record length.
L6	Accepted but not used; reserved for future use.	
L7	Accepted but not used; reserved for future use.	

1 For a VSAM data set, the equivalent of LRECL is maximum record

2 See "VSAM Data Set Notes and Limitations" on page 4 for more information.

Figure 3 (Part 15 of 18). Control Statement Summary

Operation	Parameters
SORT	{FIELDS=(p,m,f,s...,p,m,f,s) | FIELDS=(p,m,s...,p,m,s),FORMAT=f | FIELDS=COPY} [,CKPT] [,DYNALLOC[={d | (d) | (,n) | (d,n)}]] [,EQUALS | ,NOEQUALS] [,FILSZ=x | ,SIZE=y | ,FILSZ=En | ,SIZE=En] [,SKIPREC=z] (See Appendix D for functions to which these parameters apply.)

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.	The sum of lengths must not exceed 4092 bytes.
f	Format.	Permissible formats are: CH – EBCDIC character, unsigned ZD – Zoned decimal, signed PD – Packed decimal, signed FI – Fixed-point binary, signed BI – Binary, unsigned FL – Floating point, signed AC – ISCII/ASCII character, unsigned CSL – EBCDIC numeric, leading separate sign CST – EBCDIC numeric, trailing separate sign CLO – EBCDIC numeric, leading overpunch sign CTO – EBCDIC numeric, trailing overpunch sign ASL – ISCII/ASCII numeric, leading separate sign AST – ISCII/ASCII numeric, trailing separate sign AQ – EBCDIC character, alternate collating sequence
s	Desired sequencing.	Must be one of the following: A – Ascending. D – Descending. E – User-modified control field that will be sorted or merged in ascending order.
COPY		See explanation and notes for this parameter on the OPTION statement.

Figure 3 (Part 16 of 18). Control Statement Summary

Parameter	Explanation	Notes
FORMAT=f	Optional; may be used when all control field data formats are the same.	The permissible values for f are listed above.
CKPT		See explanation and notes for this parameter on the OPTION statement.
DYNALLOC=		See explanation and notes for this parameter on the OPTION statement.
EQUALS NOEQUALS		See explanation and notes for this parameter on the OPTION statement.
FILSZ=x SIZE=y		See explanation and notes for this parameter on the OPTION statement.
SKIPREC=z		See explanation and notes for this parameter on the OPTION statement.

Operation	Parameters
SUM	{FIELDS=(p,m,f...,p,m,f) | FIELDS=(p,m...,p,m),FORMAT=f | FIELDS=NONE} (See Appendix D for functions to which these parameters apply.)

Parameter	Explanation	Notes
FIELDS=	Describes summary fields.	
p	Position within record.	
m	Length.	
f	Format.	Permissible values are: BI – Binary, unsigned. FI – Fixed-point binary, signed. PD – Packed decimal, signed. ZD – Zoned decimal, signed.
NONE	Eliminates records with duplicate keys.	No summation is performed.

Figure 3 (Part 17 of 18). Control Statement Summary

Parameter	Explanation	Notes
FORMAT=f	Optional; may be used when all control field data formats are the same.	The permissible values for f are listed above.

Figure 3 (Part 18 of 18). Control Statement Summary

Control Statement Compatibility

The control statements INPFIL and OUTFIL, which are used by other IBM sort programs, are accepted by this release, but not processed. The information contained in the INPFIL and OUTFIL statements is supplied to the program in DD statements.

Because the OPTION control statement is now used by DFSORT, any job streams from other IBM sort programs that still contain an OPTION control statement causes DFSORT to terminate unless the parameters conform to the new OPTION control statement.

The program accepts SORT, MERGE, RECORD, END, and ALTSEQ statements prepared for other IBM System/360 or System/370 sort/merge programs; any obsolete parameters are ignored. However, because of the difference in parameter specifications, the program does 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/VS1 and MVS program, the reverse is not necessarily true, because this program provides facilities that the others do not.

General Coding Rules

See "Comment Statements" on page 19 for a description of comment statements.

All other DFSORT control statements have the same general format, shown in Figure 4.

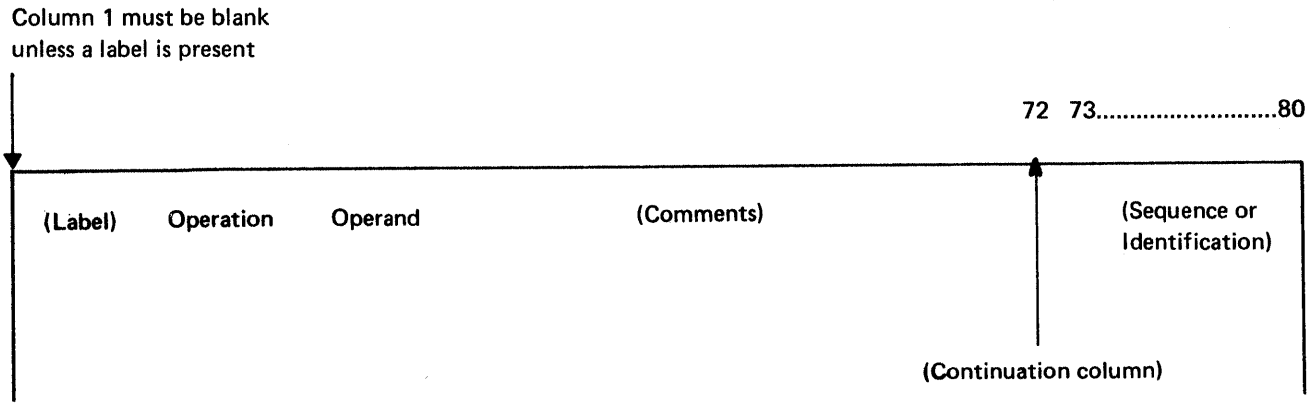


Figure 4. Control Statement Format

The control statements are free-form; that is, the operation definer, operand(s), and comment field may appear anywhere in a statement, provided 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 line. 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 line. It contains a word (for example, SORT or MERGE) 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 composed 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 line, this field must begin on the first line. Each operand has an operand definer, or parameter (a group of characters that identifies the operand type to DFSORT). A value or values may be associated with a parameter. The three possible operand formats are:

- parameter
- parameter=value
- parameter=(value1,value2...,value*n*)

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 last 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 line. However, as long as the last character of the operand field on a line is a comma followed by a blank, the program assumes that the next line is a continuation line. The nonblank character in column 72 is required only when a comments field is to be continued or when an operand is broken at column 71.

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

Continuation Lines

The format of the DFSORT continuation line is shown in Figure 5 on page 40.

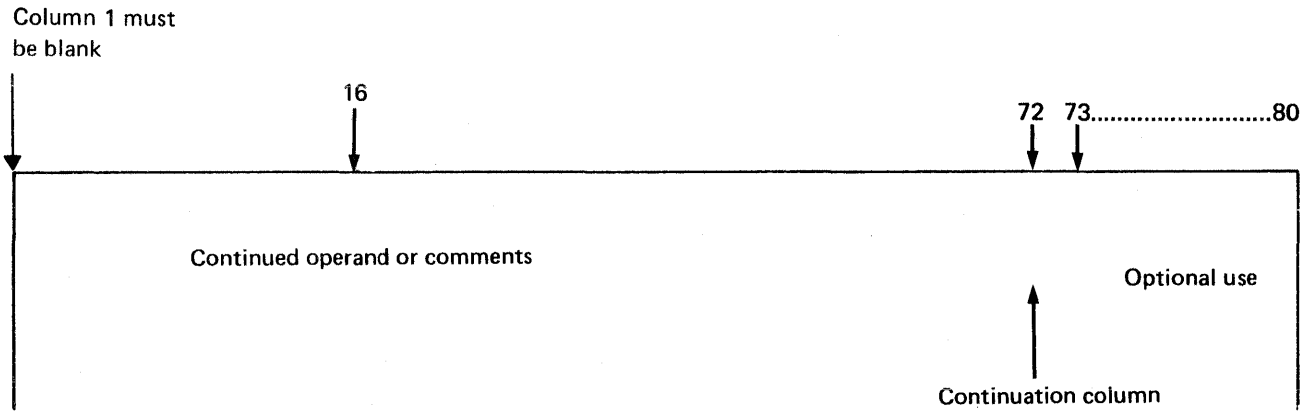


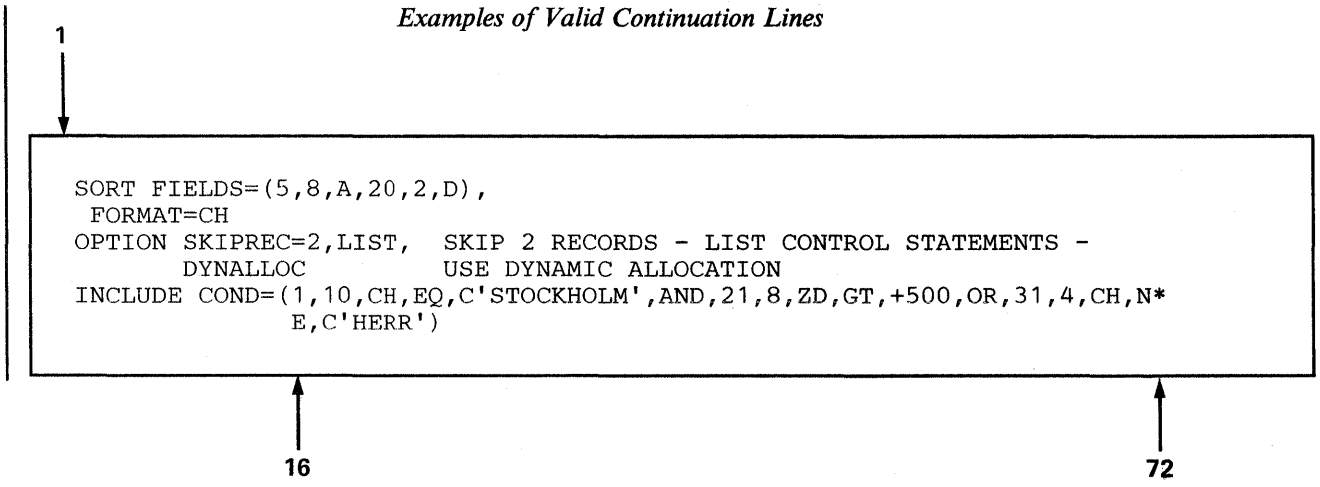
Figure 5. Continuation Line Format

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

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

- If a comments field is broken or is to be started on a new line, column 72 must contain a nonblank character. The continuation can begin in any column from 2 through 71.
- If an operand field is broken after a comma, the continuation column (72) can be left blank, and the continuation can begin in any column from 2 through 71. If the comma is in column 71 and column 72 contains a nonblank character, the continuation must begin in column 16.
- If an operand field is not broken after a comma, the operand field must be broken at column 71. Column 72 must contain a nonblank character. The continuation must begin in column 16.

Examples of Valid Continuation Lines



Summary of Restrictions

The following rules apply to control statement preparation:

- Column 1 of each control statement must be blank unless a label or comment statement is present (a comment statement is indicated by an asterisk in column 1).
- 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 line of a control statement.
- The first operand must begin on the first line 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 within a single source (for example, the SYSIN data set).

ALTSEQ Control Statement

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

The ALTSEQ statement is used if you want to change the collating sequence of EBCDIC character data; it only changes the order in which it is collated, not the data itself. If a modified version of the collating sequence is available by default at your installation, the ALTSEQ statement overrides it.

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

CODE=(*fftt,fftt...*)
The modifications are described in this form 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 not important.

Notes:

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

2. *Use of ALTSEQ can degrade performance.*

Default: Usually the installation option, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

ALTSEQ Statement Examples

ALTSEQ Example 1

```
ALTSEQ    CODE=(5BEA)
```

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

ALTSEQ Example 2

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

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

DEBUG Control Statement

```
DEBUG  [ABEND | NOABEND]
       [,ABSTP]
       [,BSAM]
       [,BUFFERS={ANY | BELOW}]
       [,CTRx=n]
       [,FMTABEND]
       [,NOASSIST]
```

For a tape work sort or a conventional merge, only the ABEND | NOABEND parameters of the DEBUG statement are used.

The statement is not intended for regular use; only the first three parameters (ABEND | NOABEND, ABSTP and BSAM) are of general interest. For more information about problem diagnosis, see *DFSORT Diagnosis Guide.*

ABEND | NOABEND
indicates whether DFSORT abends or terminates with a return code of 16 if your sort or merge is unsuccessful.

ABEND
If you specify this parameter and your sort or merge is unsuccessful, it abends with a user completion code equal to the appropriate message number. It also causes an abend if the unsuccessful sort or merge was dynamically invoked.

NOABEND
An unsuccessful sort or merge terminates with a return code of 16.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

ABSTP
This option prevents loss of needed information in a dump when Blockset terminates during execution phase processing. If the DFSORT application is unsuccessful, an abend is forced with a completion code equal to the appropriate message number. The message is *not* written. This option overrides ERET, ABEND, and NOABEND.

Default: None; optional

Applicable Functions: See Appendix D.

BSAM
DFSORT normally uses the EXCP access method for SORTIN and SORTOUT. If you encounter a problem related to this I/O activity, you can

temporarily bypass it by specifying this parameter. It is ignored for VSAM SORTIN and/or SORTOUT data sets.

Note: Use of this option may degrade performance.

Default: None; optional.

Applicable Functions: See Appendix D.

BUFFERS={ANY | BELOW}
DFSORT normally allocates RSA and buffers above 16-megabyte virtual. You can temporarily bypass the default by specifying BELOW.

ANY
specifies that the record storage area (RSA) and input/output buffers may be allocated either above or below 16-megabyte virtual.

BELOW
specifies that the record storage area (RSA) and input/output buffers must be allocated below 16-megabyte virtual.

Note: BSAM buffers are always allocated below 16-megabyte virtual.

Default: ANY

Applicable Functions: See Appendix D.

CTRx=*n*
The program keeps a count of the input or output records. When the count reaches the value specified (n), the program abends and a specially formatted dump is printed on the message data set.

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 (ignored for Blockset)

5 Count of output records deleted by E35 (ignored for Blockset)

Default: None; optional.

Applicable Functions: See Appendix D.

FMTABEND
Specifying this parameter causes a specially formatted dump to be printed on the message data set when DFSORT abends. The specially formatted dump is the same as that produced when CTRx=n is specified.

Default: None; optional.

Applicable Functions: See Appendix D.

NOASSIST

DFSORT uses System/370-XA Sorting Instructions on MVS/XA, when possible. If you do not want to use these instructions, you can temporarily bypass them by specifying this parameter.

Default: None; optional.

Applicable Functions: See Appendix D.

Forcing a Specially Formatted Dump

The default ERET=ABEND | RC16, which was set at DFSORT installation time, can be overridden by the DEBUG control statement.

To obtain a specially formatted dump on the message data set when DFSORT terminates, CTRx=n or FMTABEND 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 6 on page 47.

Figure 7 on page 48 shows how to interpret a formatted dump for Peerage or Vale. (The formatted dump for Blockset is similar to the formatted dump for Peerage or Vale.)

SYSTEM DUMP
SNAP dump corresponding to a normal SYSUDUMP dump.

FORMATTED DATA

1 Save areas
The standard save areas used by different levels of the program.

2 Abend code
A fullword with the format X'xxsssuuu', where

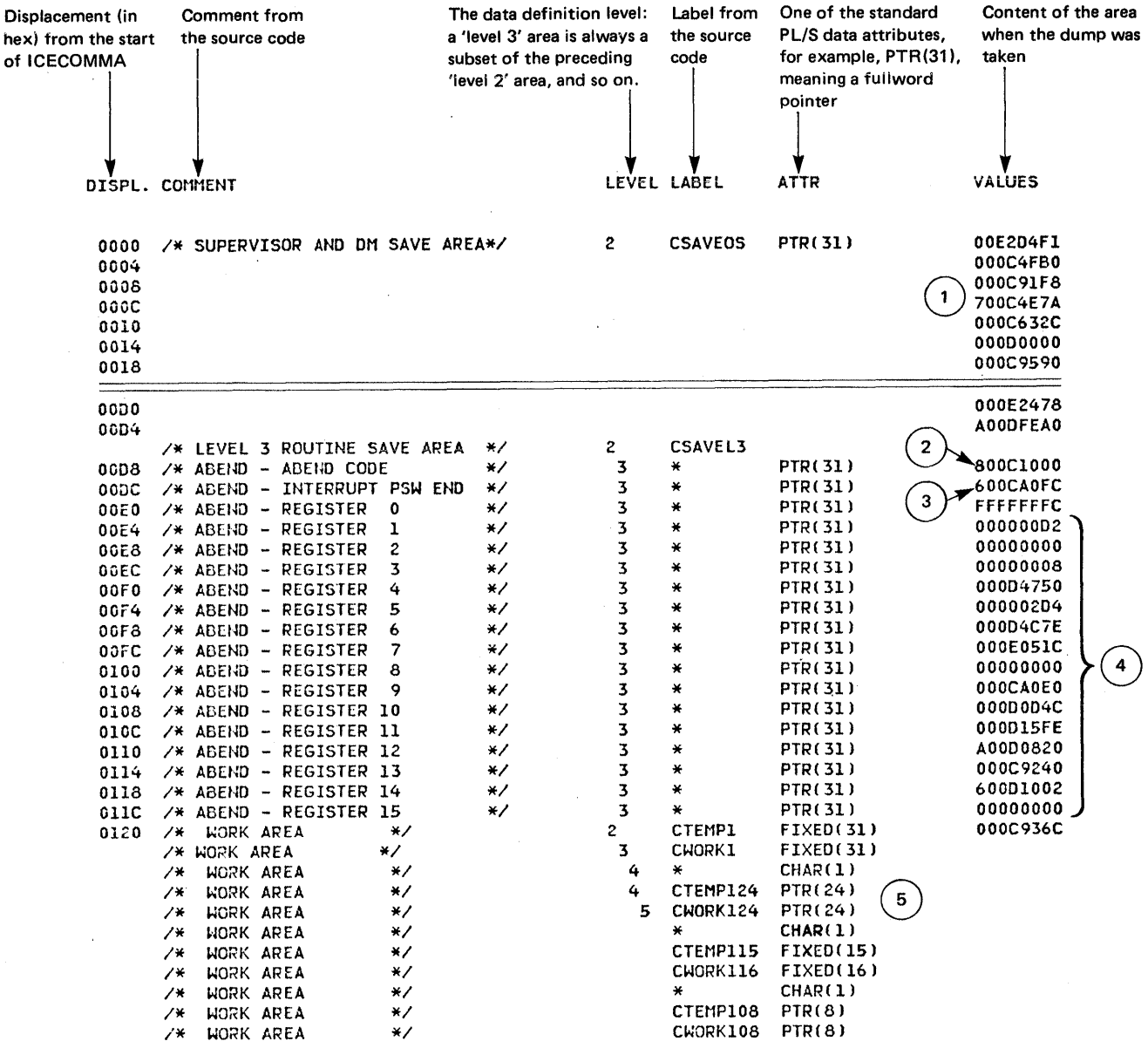
xx is the standard abend code prefix,
sss is the system completion code at program failure (or zeros), and
uuu is the user completion code at uncorrectable error (or zeros).
This code is equal to the message number (for example, '046' would represent message ICE046A).

3 A fullword containing the address of the instruction at which failure occurred.

4 Register contents when program failure occurred: 16 fullwords, giving the register contents in the order 0 through 15.

5 Contents of the communication area (D4COMMON for Blockset or ICECOMMA for Peerage or Vale) formatted when program failure occurred, with offsets from register 13, comments, labels, and definitions.

Figure 6. Contents of a Specially Formatted Dump



(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 7. Interpreting a Formatted Dump (Shown for Peerage or Vale)

END Control Statement

```
END
```

The END statement is required if you want DFSORT to discontinue reading SYSIN or SORTCNTL before end-of-file.

When you link-edit user exit routines dynamically, the END statement marks the end of the DFSORT control statements and the beginning of exit routine object decks in SYSIN.

END Statement Examples

END Example

```
//SYSIN DD *
  SORT FIELDS=(1,6,A,28,5,D),FORMAT=CH
  RECORD TYPE=V,LENGTH=(200,,,,80)
  END
  OPTION DYNALLOC
```

Because the OPTION statement appears after the END statement, it cannot be read.

END Example with SYSIN Input for Dynamic Link-Editing

```
//SYSIN DD *
  SORT FIELDS=(5,8,CH,A)
  MODS E15=(E15,1024,SYSIN,T)
  END
<object deck for E15 exit here>
```

The END statement precedes the E15 exit routine object deck in SYSIN.

INCLUDE Control Statement

```
INCLUDE  {COND=(p1,m1,f1,{EQ | NE | GT | GE | LT | LE}
         ,{p2,m2,f2 | constant}[{,AND | ,OR},...]) |
         COND=(p1,m1,{EQ | NE | GT | GE | LT | LE}
         {p2,m2 | constant}[{,AND | ,OR},...]),FORMAT=f}
```

An INCLUDE statement is used if you want only certain records to appear in the output data set. By using the INCLUDE statement, you select the records that qualify for inclusion.

The INCLUDE statement defines a logical expression (that is, one or more comparisons logically combined) based on fields in the input record. Each comparison may be between two input fields or between an input field and a constant. If the logical expression is true for a given record, the record is included in the output data set. For example, you could compare the first 6 bytes of each record with its last 6 bytes, and include only those records in which those fields are identical. Or you could compare a field with a specified date, and include only those records with a more recent date.

You must not supply both an INCLUDE and an OMIT statement to the same DFSORT run.

COND

The logical expression of the COND parameter can be represented at a high level by the following format: **COND=**(*relational condition1*[{**,AND | ,OR**},*relational condition2*...])

Default: None; must be specified.

Applicable Functions: See Appendix D.

FORMAT=*f*

FORMAT=f can be used only when all the fields in the whole COND expression have the same format. The permissible field formats are shown under the description of f for fields.

Default: None; optional.

Applicable Functions: See Appendix D.

Relational Condition

The relational condition specifies a comparison to be performed. Its format is described below. Relational conditions can be logically combined, with AND or OR, to form a logical expression. If they are combined, the following rules apply:

1. "AND" statements are evaluated before "OR" statements unless parentheses are used to change the order of evaluation; expressions inside parentheses are

always evaluated first. (Nesting of parentheses is limited only by the amount of storage available.)

2. The signs & (AND) and | (OR) may be used instead of the words.

Relational Condition Format

The format of the relational condition is:

p1,m1[*,f1*] ,{**EQ** | **NE** | **GT** | **GE** | **LT** | **LE**} ,{*p2,m2*[*,f2*] | *constant*}	Comparison operators: **EQ** - Equal to **NE** - Not equal to **GT** - Greater than **GE** - Greater than or equal to **LT** - Less than **LE** - Less than or equal to

Fields

p1,m1,f1: The variables p1, m1, and f1 specify a field in the input record to be compared to either another field in the input record, or to a constant.

- p1 specifies the first byte of the field relative to the beginning of the input record.[1] The first data byte of a fixed-length record (FLR) has relative position 1. The first data byte of a variable-length (VLR) record has relative position 5 (because the first 4 bytes contain the RDW). All fields must start on a byte boundary, and no field may extend beyond byte 4092.

- m1 specifies the length of the field. Acceptable lengths for different formats are given below.

- f1 specifies the format of the data in the field. Permissible formats are given below.

  If all the data fields contain the same type of data, this value may be omitted, in which case you must use the FORMAT=f operand.

Format	Length	Description
CH	1-256	Character EBCDIC, unsigned.[2]
ZD	1-256	Zoned decimal, signed.
PD	1-255	Packed decimal, signed.
FI	1-256	Fixed-point, signed.

---

1 If your E15 exit routine formats the record, p1 must refer to the record as reformatted by the exit.

2 If CHALT is in effect, CH is treated as AQ.

Format	Length	Description
BI	1-256	Binary, unsigned.
AC	1-256	ISCII/ASCII character, unsigned.
CSL	2-256	EBCDIC numeric, leading separate sign.
CST	2-256	EBCDIC numeric, trailing separate sign.
CLO	1-256	EBCDIC numeric, leading overpunch sign.
CTO	1-256	EBCDIC numeric, trailing overpunch sign.
ASL	2-256	ISCII/ASCII numeric, leading separate sign.
AST	2-256	ISCII/ASCII numeric, trailing separate sign.
AQ	1-256	EBCDIC character, alternate collating sequence.

p2,m2,f2: These parameters specify another field in the input record with which the p1, m1, and f1 input field will be compared. Permissible comparisons between input fields with different formats are shown in Figure 8.

Note that, for maximum performance, all comparisons in a complex expression are checked in a single pass for each record. For this reason, if *all* records do not contain *all* INCLUDE/OMIT fields, message ICE015A is issued; that is, you *cannot* use a complex expression in which one of the comparisons excludes variable-length records that are too short to contain other fields in the expression.

Field FORMAT	BI	CH	ZD	PD	FI	AC	ASL	AST	CSL	CST	CLO	CTO	AQ
BI	X	X											
CH	X	X											
ZD			X	X									
PD			X	X									
FI					X								
AC						X							
ASL							X	X					
AST							X	X					
CSL									X	X			
CST									X	X			
CLO											X	X	
CTO											X	X	
AQ													X

Figure 8. Permissible Field-to-Field Comparisons for INCLUDE/OMIT

Constants: A constant can be decimal, character, or hexadecimal. The different formats are shown in detail below. Permissible comparisons between input fields and types of constants are shown in Figure 9.

Field Format	Self-Defining Term		
	Decimal Number	Character String	Hexadecimal String
BI		x	x
CH		x	x
ZD	x		
PD	x		
FI	x		
AC		x	x
ASL	x		
AST	x		
CST	x		
CSL	x		
CLO	x		
CTO	x		
AQ		x	x

Figure 9. Permissable Field-to-Constant Comparisons for INCLUDE/OMIT

Decimal Number Format: The format for coding a decimal constant is:

```
[±]n
```

When the decimal constant is compared with a field of FI format, it may not be larger than 2147483647 nor smaller than -2147483648.

Examples of valid and invalid decimal constants are:

Valid	Invalid	
15	++15	Too many sign characters
+15	15+	Sign in wrong place
-15	1.5	Contains invalid character
18000000	1,500	Contains invalid character

Character String Format: The format for coding a character string constant is:

```
C'xx...x'
```

The value x may be any EBCDIC character (the EBCDIC character string is translated appropriately for comparison to an AC or AQ field).

If you want to include a single apostrophe in the character string, you must specify it as two single apostrophes. Thus:

```
Required:  O'NEILL     Specify:  C'O''NEILL'
```

Examples of valid and invalid character string constants are shown below:

Valid	Invalid
`C'JOHN DOE INC'`	`C'''''` Apostrophes not paired
`C'$@#'`	`'ABCDEF'` C identifier missing
`C'+0.193'`	`C'ABCDEF` Apostrophe missing

Hexadecimal String Format: The format for coding a hexadecimal string constant is:

```
X'yy...yy'
```

The value yy represents any pair of hexadecimal digits.

Examples of valid and invalid hexadecimal constants are shown below.

Valid	Invalid
`X'FF'`	`X'ABGD'` Invalid hexadecimal digit
`X'BF3C'`	`X'F1F'` Incomplete pair of digits
`X'AF050505'`	`'BF3C'` Missing X indentifier
	`'BF3C'X` X identifier in wrong place

Padding and Truncation

In a field-to-field comparison, the shorter field is padded appropriately. In a field-to-constant comparison, the constant is padded or truncated to the length of the field.

Character and hexadecimal strings are truncated and padded on the right.

The padding characters are:

X'40' For character string
X'00' For hexadecimal string

Decimal constants are padded and truncated on the left. Padding is done with zeros in the proper format.

INCLUDE/OMIT Statement Notes

1. The size of the routine generated by DFSORT to handle the INCLUDE/OMIT function is dependent on how many fields are referenced, and what lengths and formats they have. The size of the routine must not exceed 4096 bytes or DFSORT will issue a message and terminate.

2. Floating point fields may not be referenced in INCLUDE or OMIT statements.

3. Any selection can be performed with either an INCLUDE or an OMIT statement. INCLUDE and OMIT are mutually exclusive.

4. Remember that if several relational conditions are joined with a combination of AND and OR logical operators, the AND statement is evaluated first. The order of evaluation may be changed by using parentheses inside the COND expression.

5. If any changes are made to record formats by exits E15 or E32, the INCLUDE or OMIT statement must apply to the newest formats.

6. DFSORT issues a message and terminates if an INCLUDE or OMIT statement is specified for a tape work data set sort or conventional merge application.

Figure 10 on page 56 shows how DFSORT reacts to the result of a relational condition comparison, depending on whether the statement is INCLUDE or OMIT and whether the relational condition is followed by an AND or an OR logical operator.

When writing complex statements, be sure the result will be what you want. The table in Figure 10 should help you.

Note that, for maximum performance, all comparisons in a complex statement are checked in a single pass for each record. For this reason, if ***all*** records do not contain ***all*** INCLUDE/OMIT fields, message ICE015A is issued; that is, you ***cannot*** use a complex statement in which one of the comparisons excludes variable-length records too short to contain other fields in the statement.

Statement	Relational Condition	Program action if next logical operator is:	
	Compare	AND	OR
OMIT	True	Check next compare, or if last compare, OMIT record	OMIT record
OMIT	False	INCLUDE record	Check next compare or if last compare, INCLUDE record
INCLUDE	True	Check next compare, or if last compare, INCLUDE record	INCLUDE record
INCLUDE	False	OMIT record	Check compare, or if last compare, OMIT. record

Figure 10. Logic Table for INCLUDE/OMIT

INCLUDE Statement Examples

INCLUDE Example 1

```
INCLUDE    COND=(5,8,GT,13,8,I,105,4,LE,1000),FORMAT=FI
```

DFSORT includes only records in which:

- The fixed-integer number in bytes 5 through 12 is greater than the fixed-integer number in bytes 13 through 20. *OR*
- The fixed-integer number in bytes 105 through 108 is less than or equal to 1000.

Note that all four fields have the same format.

INCLUDE Example 2

```
INCLUDE    COND=(1,10,CH,EQ,C'STOCKHOLM',
           AND,21,8,ZD,GT,+50000,
           OR,31,4,CH,NE,C'HERR')
```

This statement only includes records in which:

- The first 10 bytes contain STOCKHOLM (this 9 character string was padded on the right with a blank) *AND* the zoned-decimal number in bytes 21 through 28 is greater than 50000. *OR*
- Bytes 31 through 34 do not contain HERR.

Note that the AND is evaluated before the OR. (The OMIT example "OMIT Statement Example" on page 71, illustrates how parentheses may be used to change the order of evaluation.) Also note that ending a line with a comma-blank indicates continuation on the next line starting in any position from 2 through 71.

INCLUDE Example 3

```
INCLUDE    COND=((5,1,CH,EQ,8,1,CH),&,
                 ((20,1,CH,EQ,C'A',&,30,1,FI,GT,10),|,
                  (20,1,CH,EQ,C'B',&,30,1,FI,LT,100),|,
                  (20,1,CH,NE,C'A',&,20,1,CH,NE,C'B')))
```

This statement only includes records in which:

- Byte 5 equals byte 8. *AND*
- One of the following is true:
  - Byte 20 equals 'A' and byte 30 is greater than 10.
  - Byte 20 equals 'B' and byte 30 is less than 100.
  - Byte 20 is not equal to 'A' or 'B'.

INREC Control Statement

INREC FIELDS=([*s*,]*p*,*m*[,*a*]...[,*s*][,*p*[,*m*][,*a*]][,*s*])

The INREC control statement allows you to reformat the input records before they are processed; that is, to define which parts of the input record are to be included in the reformatted input record, in what order they are to appear, and how they are to be aligned.

You do this by defining one or more fields from the input record. The reformatted input record consists of those fields only, in the order in which you have specified them, and aligned on the boundaries you have indicated.

You can also pad reformatted input records with blanks and/or binary zeros before, between, and/or after the input fields, using the s parameter.

For information concerning the interaction of INREC and OUTREC, see also "Using Options That May Enhance Performance" on page 217.

FIELDS=([*s*,]*p*,*m*[,*a*]...[, *s*][,*p*[,*m*].[,*a*]][,*s*])

You can use this parameter to specify the order in which the input and separation fields are to appear in the reformatted input record.

s

indicates a separation field to be inserted into the reformatted input record in the position you code it relative to the input fields. It can be specified before or after the p,m,a parameters for any field. Permissible values are:

nX Blank separation. n bytes of EBCDIC blanks (X'40') are inserted in the reformatted input records. n may be from 1 to 256.

nZ Binary zero separation. n bytes of binary zeros (X'00') are inserted in the reformatted input records. n may be from 1 to 256.

Consecutive separation fields may be specified.

For variable-length records:

- Separation field(s) must not be specified before the first input field (the RDW).
- Separation field(s) must not be specified after the variable part of the input record.

p

specifies the first byte of the input field relative to the beginning of the input record.[3] The first data byte of a fixed-length record has relative position 1. The first data byte of a variable-length record has relative position 5 (because the first 4 bytes contain the RDW). All fields must start on a byte boundary, and no field may extend beyond byte 32000. For special rules concerning variable-length records, see "INREC Statement Notes."

m

specifies the length of the input field. It must include the sign if the data is signed, and must be a whole number of bytes. See note 5 on page 60 for more information.

a

specifies the alignment (displacement) of the input field in the reformatted input record, relative to the start of the reformatted input record.

The permissible values are:

H Halfword aligned. This means that the displacement (p-1) of the field from the beginning of the reformatted input record, in bytes, is a multiple of 2 (that is, position 1, 3, 5, and so forth).

F Fullword aligned. The displacement is a multiple of 4 (that is, position 1, 5, 9, and so forth).

D Doubleword aligned. The displacement is a multiple of 8 (that is, position 1, 9, 17, and so forth).

Alignment can be necessary if, for example, the data is to be used in a COBOL application program where COMPUTATIONAL items are aligned through the SYNCHRONIZED clause. Unused space preceding aligned fields will always be padded with binary zeros.

Default: None; must be specified.

Applicable Functions: See Appendix D.

INREC Statement Notes

1. When INREC is specified, DFSORT reformats the input records after user exit E15 and/or INCLUDE/OMIT processing is finished. Thus, references to fields by your E15 exit and INCLUDE/OMIT statements are not affected, whereas your SORT, OUTREC, and SUM statements must refer to fields in the *reformatted input* record. Your E35 exit must refer to fields in the *reformatted output* record (see below).

---

[3] If your E15 exit reformats the record, p must refer to the record as reformatted by the exit.

2. When you specify INREC, you should be aware of the change in record size and layout of the resulting reformatted input records. You should also understand how reformatting of records affects sort performance, and how to use INREC and/or OUTREC to achieve the most efficient sort. (See also "OUTREC Control Statement" on page 92 and "Using Options That May Enhance Performance" on page 219 for more details.)

3. For variable-length records, the first entry in the FIELDS parameter must specify or include the 4-byte Record Descriptor Word (RDW). DFSORT sets the length of the reformatted record in the RDW.

   If the first field in the data portion of the input record is to appear in the reformatted input record immediately following the RDW, the entry in the FIELDS parameter can specify both RDW and data field in one. Otherwise, the RDW must be specifically included in the reformatted input record.

4. The length of the INREC/OUTREC record (reformatted length) is not used to determine the LRECL of SORTOUT. If not specified in the data set control block (DSCB) or DD statement, the value for SORTOUT LRECL is determined in the usual way (that is, from the L3 value or SORTIN LRECL). If the reformatted length does not match the SORTOUT LRECL, the same checks used when the SORTIN LRECL does not match the SORTOUT LRECL are made and padding/truncation is performed, if possible. When processing variable-length records, the maximum SORTIN LRECL must not exceed the maximum SORTOUT LRECL.

   For VSAM data sets, the maximum record size defined in the cluster is equivalent to the LRECL when processing fixed-length records, and is 4 less than the LRECL when processing variable-length records. See "VSAM Data Set Notes and Limitations" on page 4 for more information.

5. The variable part of the input record (that part beyond the minimum record length) may be included in the reformatted input record, and if included, must be the last part. In this case, a value should be specified for p*n* that is less than or equal to the minimum record length (see L4 of the RECORD control statement) plus 1 byte; m*n* and a*n* should be omitted.

   If INREC and OUTREC are both specified, either both must specify position-only for the last part, or neither must specify position-only for the last part.

   If the reformatted input includes only the RDW and the variable part of the input record, "null" records containing only an RDW could result.

6. The input records are reformatted before processing, as specified by INREC. The output records are in the format specified by INREC, unless OUTREC is also specified.

7. Fields referenced in INREC statements may overlap each other and/or control fields.

8. If input is variable records, the output is also variable. This means that each record is given the correct RDW by DFSORT before output, even if the records are treated as fixed internally because they are all the same length.

9. In general, INREC should be used to reduce the length of input records as much as possible to achieve the most efficient processing (and OUTREC should be used to reformat the records for output). However, in the case where overflow could occur during summation, INREC could be used to create a larger SUM field in the reformatted input record (perhaps resulting in a larger record for sorting or merging), so that overflow does not occur.

10. DFSORT issues a message and terminates if an INREC statement is specified for a tape work data set sort or conventional merge application.

INREC Statement Examples

INREC Example 1

```
INCLUDE COND=(5,1,GE,C'M'),FORMAT=CH
INREC FIELDS=(10,3,20,8,33,11,5,1)
SORT FIELDS=(4,8,CH,A,1,3,FI,A)
SUM FIELDS=(17,4,BI)
```

OUTREC Example 2

```
INCLUDE COND=(5,1,GE,C'M'),FORMAT=CH
OUTREC FIELDS=(10,3,20,8,33,11,5,1)
SORT FIELDS=(20,8,CH,A,10,3,FI,A)
SUM FIELDS=(38,4,BI)
```

The above examples illustrate how a fixed-length input data set is sorted and reformatted for output. A more efficient sort is achieved by eliminating unnecessary fields, before sorting, using INREC. The SORTIN LRECL is 80.

Records are also included or excluded by means of the INCLUDE statement, and summed by means of the SUM statement.

The reformatted input records are fixed length, with a record size of 23 bytes (a significant reduction from the original size of 80 bytes). The SORTOUT LRECL should be specified as 23. They look as follows:

Position	**Contents**
1-3	Input positions 10 through 12
4-11	Input positions 20 through 27
12-22	Input positions 33 through 43
23	Input position 5

Identical results are achieved with INREC or OUTREC. However, use of INREC can result in better performance. In either case, the INCLUDE COND parameters must refer to the fields of the original input records. However, with INREC, the SUM and SORT FIELDS parameters must refer to the fields of the *reformatted* input records, while with OUTREC, the SUM and SORT FIELDS parameters must refer to the fields of the *original* input records.

INREC Example 3

```
INREC FIELDS=(1,35,2Z,36,45)
MERGE FIELDS=(20,4,CH,D,10,3,CH,D),FILES=3
SUM FIELDS=(36,4,BI,40,8,PD)
RECORD TYPE=F,LENGTH=(80,,82)
```

This example illustrates how overflow of a summary field can be prevented when three fixed-length data sets are merged and reformatted for output. The input record size is 80 bytes. To illustrate the use of the RECORD statement, assume that SORTIN and SORTOUT are not present (that is, all input/output is handled by user exits).

The reformatted input records are fixed length, with a record size of 82 bytes (an insignificant increase from the original size of 80 bytes). They look as follows:

Position	**Contents**
1-35	Input positions 1 through 35
36-37	Binary zeros (to prevent overflow)
38-82	Input positions 36 through 80

The MERGE and SUM statements must refer to the fields of the reformatted input records.

The reformatted output records are identical to the reformatted input records.

Thus, the 2-byte summary field at positions 36 and 37 in the original input records expand to a 4-byte summary field in positions 36 through 39 of the reformatted input/output record *before* merging. This prevents overflow of this summary field. Note that, if OUTREC were used instead of INREC, the records would be reformatted *after* merging, and the 2-byte summary field could overflow.

Note: This method of preventing overflow *cannot* be used for negative FI summary fields because padding with zeros rather than ones would change the sign.

INREC Example 4

```
INREC FIELDS=(1,4,15,15,47,50,101)
SORT FIELDS=(32,4,CH,A)
RECORD TYPE=V,LENGTH=(,,,100,130)
OUTREC FIELDS=(1,4,10Z,5,15,17Z,20,50,4X,70)
```

This example illustrates how a variable-length input data set can be sorted more efficiently by eliminating padding fields before sorting, and reinserting them after sorting. The resulting output records are *not* actually reformatted. The variable part of the input records is included in the output records. The minimum input record size is 100 bytes, and the maximum input record size (SORTIN LRECL) is 200 bytes.

The reformatted input records are variable length, with a minimum record size of 69 bytes and a maximum record size of 169 bytes (a significant reduction from the original sizes of 100 and 200 bytes, respectively). They look as follows:

Position	Contents
1-4	RDW (input positions 1 through 4)
5-19	Input positions 15 through 29
20-69	Input positions 47 through 96
70-n	Input positions 101 through n (variable part of input records)

The SORT and OUTREC statements must refer to the fields of the reformatted input records.

Because padding fields are removed by INREC and reinserted by OUTREC, the output records are identical to the original input records; that is, variable length, with a minimum record size of 100 bytes and a maximum record size (SORTOUT LRECL) of 200 bytes. They look as follows:

Position	Contents
1-4	RDW
5-14	Binary zeros
15-29	Input positions 15 through 29
30-46	Binary zeros
47-96	Input positions 47 through 96
97-100	EBCDIC blanks
101-n	Input positions 101 through n (variable part of input records)

Thus, the use of INREC and OUTREC allows sorting of smaller records, although the output records are not actually reformatted.

INREC Example 5

```
INREC FIELDS=(20,4,12,3)
SORT FIELDS=(1,4,D,5,3,D),FORMAT=CH
OUTREC FIELDS=(5X,1,4,H,8X,1,2,5,3,80Z)
```

This example illustrates how a fixed-length input data set can be sorted and reformatted for output. A more efficient sort is achieved by using INREC to reduce the input records as much as possible before sorting, and using OUTREC to repeat fields and insert padding after sorting. The SORTIN LRECL is 80 bytes.

Note: Contrast this example with OUTREC Example 4, where INREC does not achieve a more efficient sort because no fields can be eliminated before sorting.

The reformatted input records are fixed length, with a record size of 7 bytes (a significant reduction from the original size of 80 bytes). They look as follows:

Position	Contents
1-4	Input positions 20 through 23
5-7	Input positions 12 through 14

The SORT and OUTREC statements must refer to the fields of the reformatted input records.

The reformatted output records are fixed length, with a record size of 103 bytes; the SORTOUT LRECL is specified as 103. They look as follows:

Position	Contents
1-5	EBCDIC blanks
6	Binary zero (for H alignment)
7-10	Input positions 20 through 23
11-18	EBCDIC blanks
19-20	Input positions 20 through 21
21-23	Input positions 12 through 14
24-103	Binary zeros

Thus, the use of INREC and OUTREC allows sorting of 7-byte records rather than 80-byte records, even though the output records are 103 bytes long.

MERGE Control Statement

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

The MERGE control statement must be used when a merge operation is to be performed. It may also be used to specify a copy application. It provides essentially the same information to DFSORT 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 DYNALLOC option is not used (ignored if specified).
- The FILSZ/SIZE value takes *all* the input data sets into account.

When an option can be specified on either the MERGE or OPTION statement, it is preferable to specify it on the OPTION statement.

A table showing other possible sources for specifying options available on the MERGE statement and the rules of override are in Appendix D.

FIELDS=(*p,m,f,s...p,m,f,s*)
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 3 on page 20.

FIELDS=COPY
See the discussion of this operand on the OPTION statement.

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

CKPT
See the discussion of this operand on the OPTION statement.

FILES=*n*
specifies the number of input files to a merge when input is supplied through the E32 exit.

Default: None.

Applicable Functions: See Appendix D

EQUALS | NOEQUALS
See the discussion of this operand on the OPTION statement.

FILSZ=*x* | SIZE=*y*
See the discussion of this operand on the OPTION statement.

MERGE Statement Examples

MERGE Example 1. One Control Field, Size Option

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

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 29483 records.

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

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

FIELDS
The major control field begins on byte 3 of each record, is 8 bytes long, and contains zoned decimal data that is 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 30000 records.

MERGE Example 3. Two Control Fields, Format Option

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

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.

MERGE Example 4. COPY Option

```
MERGE    FIELDS=COPY
```

FIELDS
The input data set is copied to output. No merge takes place.

MODS Control Statement

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

The MODS statement is needed only if you want DFSORT to pass control to your routines at user exits. The MODS statement associates the user routine(s) with specific DFSORT exits and provides DFSORT with descriptions of these routines. For details about DFSORT exits and how user routines can be used, see Chapter 4, "User Exit Routines" on page 135.

To use one of the exits, you substitute its 3-character name (for example, E31) for the word *exit* in the MODS statement format above. You may specify any valid exit, *except* E32. (E32 can only be used in a merge operation that is invoked from a program; its address must be passed in a parameter list.)

exit=(n,m,s[,e])
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 your routine uses. Include storage obtained (via GETMAIN) by your routine (or, for example, by OPEN), and the storage required to load the COBOL library subroutines.

s
either the name of the DD statement in your DFSORT job step that defines the library 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, or indicates your routine is written in COBOL.

N

means that your routine has already been link-edited and can be used in the DFSORT run without further link-editing. This is the default for e. N (specified or defaulted) may be overridden by the EXEC PARM parameters 'E15=COB' and 'E35=COB'.

C

means that your E15 or E35 routine is written in COBOL. If you code C for any other exit, it is ignored, and N is assumed. Your COBOL-written routine must already have been link-edited.

T

means that your routine must be link-edited ***together*** with other routines to be used in the same phase (for example, E1n routines) of DFSORT.

This value is not valid for copy processing.

S

means that your routine requires link-editing but that it must be link-edited *separately* from the other routines (for example, E3n routines) to be used in a particular phase of DFSORT. E11 and E31 exit routines are the only routines eligible for separate link-editing.

This value is not valid for copy processing.

If you do not specify a value for e, N is assumed.

Notes:

1. *All the routines for which N or C is specified for the e parameter must be in the same library, or in libraries defined as a concatenated data set. These routines may **not** be placed in SYSIN. Each such routine must be a load module.*

2. *Each routine for which T or S is specified for the e parameter may be placed in any library or in SYSIN; they do not all have to be in the same library or SYSIN (but can be). Some routines can even be in different libraries (or the same library) and the rest can be in SYSIN. Each such routine, if in a library, can be either an object deck or a load module; if in SYSIN, it must be an object deck.*

3. *If the same routine is used in both input (that is, E1n routines) and output (that is, E3n routines) DFSORT program phases, a separate copy of the routine must be provided for each exit.*

4. *COBOL E15 and E35 exit routines can also be specified in the EXEC statement parameters (E15=COB or E35=COB). In this case, the e parameter of the MODS statement must not be "T". If "T" is specified, the program terminates with error message ICE034A. COBOL exits must already have been link-edited*

and, if the EXEC parm was specified for the exit, the e parameter must be "C", or "N", or defaulted to "N". ("S" is not valid for any E15 or E35 exit.)

5. *If you code C for a conventional merge or a tape work data set sort, DFSORT issues message ICE153A and terminates.*

For information on user exit routines in SYSIN, see "System DD Statements" on page 124.

For details on how to design your routines, refer to "Summary of Rules for User Exit Routines" on page 183.

When you are preparing your MODS statement, remember that DFSORT 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 way whether it is an exact figure or an estimate: You do not precede the value by E for an estimate.

Default: None; optional. N is the default for the fourth parameter.

Applicable Functions: See Appendix D

MODS Statement Examples

MODS Example 1. Two Routines in a Library

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

E15

At exit E15, DFSORT transfers 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 and uses 552 bytes.

E35

At exit E35, DFSORT transfers 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 and will use 11032 bytes.

MODS Example 2. E15 and E35 written in COBOL

```
MODS    E15=(COBOLE15,7000,EXITC,C),
        E35=(COBOLE35,7000,EXITC,C)
```

E15

At exit E15, DFSORT transfers control to your own routine. Your routine is written in COBOL and is in the library defined by a job control statement

with the ddname EXITC. Its member name is COBOLE15 and uses 7000 bytes.

E35

At exit E35, DFSORT transfers control to your routine. Your routine is written in COBOL and is in the library defined by the job control statement with the ddname EXITC. Its member name is COBOLE35 and uses 7000 bytes.

OMIT Control Statement

```
OMIT  {COND=(p1,m1,f1,{EQ | NE | GT | GE | LT | LE}
      ,{p2,m2,f2 | constant}[{,AND | ,OR},...]) |
      COND=(p1,m1,{EQ | NE | GT | GE | LT | LE},
      {p2,m2 | constant}[{,AND | ,OR},...]),FORMAT=f}
```

An OMIT statement is used if you do not want all the input records to appear in the output data set. By using the OMIT statement, you select the records that do *not* qualify for inclusion.

The OMIT statement defines a logical expression (that is, one or more comparisons logically combined) based on fields in the input record. Each comparison may be between two input fields or between an input field and a constant. If the logical expression is true for a given record, that record is omitted from the output data set. For example, you could compare the first 6 bytes of each record with its last 6 bytes, and omit those records in which those fields are not identical. Or you could compare a field with a specified date, and omit those records with earlier dates.

For further details on this statement, see "INCLUDE Control Statement" on page 50.

OMIT Statement Example

OMIT Example.

```
 OMIT COND=(1,10,CH,EQ,C'STOCKHOLM',&,(21,8,ZD,GT,+50000,| *
,31,4,CH,NE,C'HERR'))
```

This statement omits records in which:

- The first 10 bytes contain STOCKHOLM (the string was padded on the right with a blank). *AND,*
- The zoned-decimal number in bytes 21 through 28 is greater than 50000, *OR* bytes 31 through 34 do not contain HERR.

Note that the AND and OR operators can be written with the AND and OR signs, and that parentheses are used to change the order in which AND and OR are evaluated. Also note that the asterisk in column 72 indicates continuation of the parameters to the next line (starting in position 16).

OPTION Control Statement

```
OPTION [ARESALL={n | nK}]
       [,ARESINV={n | nK}]
       [,CHALT | ,NOCHALT]
       [,CHECK | ,NOCHECK]
       [,CKPT]
       [,COBEXIT={COB1 | COB2}]
       [,COPY]
       [,DYNALLOC[={d | (d) | (,n) | (d,n)}]]
       [,EQUALS | ,NOEQUALS]
       [,FILSZ=x | ,SIZE=y | ,FILSZ=En | ,SIZE=En]
       [,LIST | ,NOLIST]
       [,MAINSIZE={n | nK | MAX}]
       [,MSGDDN=ddname]
       [,MSGPRT={ALL | NONE | CRITICAL}]
       [,NOBLKSET]
       [,NOOUTREL]
       [,NOOUTSEC]
       [,NOSTIMER]
       [,NOWRKREL]
       [,NOWRKSEC]
       [,RESALL={n | nK}]
       [,RESINV={n | nK}]
       [,SKIPREC=z]
       [,SORTDD=cccc]
       [,SORTIN=ddname]
       [,SORTOUT=ddname]
       [,STOPAFT=n]
       [,VERIFY | ,NOVERIFY]
       [,VLSHRT | ,NOVLSHRT]
```

The OPTION control statement allows you to override some of the options available at installation time (such as EQUALS and CHECK), and to supply other optional information (such as DYNALLOC, COPY, and SKIPREC).

Some of the options available on the OPTION statement are also available on the SORT or MERGE statement (such as FILSZ and SIZE). It is preferable to specify these options on the OPTION statement. For override rules, see Appendix D.

OPTION parameters used by other IBM sort programs cause DFSORT to terminate unless they conform to the following parameters. For a description of the OPTION control statement and its parameters, see also Figure 3 on page 20.

The keywords LIST, NOLIST, MSGPRT, MSGDDN, SORTDD, SORTIN, and SORTOUT are used only when they are specified on the OPTION control statement passed by an extended parameter list. If they are specified on an OPTION statement read from the SYSIN data set or the SORTCNTL data set, the keyword is recognized, but the parameter is ignored.

The BLKSET option that was available in previous releases of DFSORT is ignored. Existing programs that use this option need not be changed.

ARESALL={*n* | *n*K}

For MVS/XA, you can use this parameter to temporarily override the installation option ARESALL=n. It indicates the number of bytes to be reserved *above* 16-megabyte virtual for system use.

ARESALL applies only to the amount of main storage above 16-megabyte virtual. This option is normally not needed because of the large amount of storage available above 16-megabyte virtual (the default for ARESALL is 0 bytes). The RESALL option applies to the amount of main storage below 16-megabyte virtual.

n

n is a decimal value that specifies the number of bytes of main storage to be reserved.

Limit: 8 digits.

***n*K**

nK specifies n times 1024 bytes of main storage are to be reserved.

Limit: 5 digits.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

ARESINV={*n* | *n*K}

For MVS/XA, you can use this parameter to temporarily override the installation option ARESINV=n. *ARESINV is used only when DFSORT is dynamically invoked.* It indicates the number of bytes to be reserved for an invoking program or for exits that reside or use space *above* 16-megabyte virtual. The reserved space is not meant for the executable code itself.

ARESINV applies only to the amount of main storage above 16-megabyte virtual. The RESINV option applies to the amount of main storage below 16-megabyte virtual.

n

n is a decimal value that specifies the number of bytes of main storage to be reserved.

Limit: 8 digits.

***n*K**

nK specifies n times 1024 bytes of main storage are to be reserved.

Limit: 5 digits.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

CHALT | NOCHALT

You can use this parameter to temporarily override the installation option CHALT={YES | NO}, that specifies whether format CH fields are translated by the alternate collating sequence as well as format AQ, or just the latter.

CHALT
: means that DFSORT translates character control fields with formats CH and AQ using the alternate collating sequence.

NOCHALT
: means that format CH fields is not translated.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D

CHECK | NOCHECK

You can use this parameter to temporarily override the installation option CHECK={YES | NO}, that specifies whether record count should be checked for applications that use the E35 user exit routine without a SORTOUT data set.

CHECK
: means that record counter checking is done at the end of program execution.

NOCHECK
: means that record counter checking is not done.

Default: Usually the installation default, but refer to Appendix E for full override details.

Applicable Functions: See Appendix D

CKPT

CKPT (the spelling CHKPT is also accepted) causes DFSORT to activate the checkpoint/restart facility of the operating system.

See "Checkpoint/Restart" on page 13 for further details.

If necessary, the Blockset technique can be bypassed so the checkpoint/restart facility can be used, by specifying either IGNCKPT=NO on the ICEMAC installation macro or NOBLKSET on the OPTION statement.

Checkpoint/restart 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 "SORTCKPT DD Statement" on page 133. In addition, you may need to specify more intermediate storage for a sort application. See "Intermediate Storage" on page 277.

Default: None; optional.

Applicable Functions: See Appendix D

COBEXIT={COB1 | COB2}
indicates whether the E15 and E35 routines written in COBOL are executed with the VS COBOL II library.

COB1
specifies that E15 and E35 routines written in COBOL are executed with the OS/VS COBOL library or, in some cases, with no COBOL library.

COB2
specifies that E15 and E35 routines written in COBOL are executed with the VS COBOL II library.

Default: Usually the installation default, but see Appendix D for full override details.

Applicable Functions: See Appendix D.

COPY
COPY causes DFSORT to copy a SORTIN data set and/or inserted records to a SORTOUT data set unless all records are disposed of by an E35 exit. Records can be edited by SKIPREC, E15, INCLUDE/OMIT, STOPAFT, INREC/OUTREC, and/or E35. E35 is entered after each SORTIN or E15 record is copied.

The following must not be used with copy applications:

- FORMAT=f
- BDAM data sets
- Dynamic link-editing

Default: None; optional.

Applicable Functions: See Appendix D.

DYNALLOC[={d | (d) | (,n) | (d,n)}]

This parameter is for MVS. This parameter assigns DFSORT the task of dynamically allocating needed work space. With DYNALLOC you do not need to calculate and specify through JCL, the amount of intermediate work space needed by the program. DFSORT, by use of the dynamic allocation facility of the operating system, allocates work space to get the best possible performance for the newest application.

d

specifies the device type. You may specify any of the following IBM devices: 2314, 3330, 3330-1, 3340, 3350, 3375, 3380, 2400, 2400-3, 2400-4, 3400-3, 3400-4, 3480, 3850, or their user-assigned group name, such as SYSDA.

n

specifies the number of requested work data sets. The maximum value of n is 16; if you specify more than 16, 16 is used.

For disk work data sets, an estimate of the number of input records is used as the basis for determining the total work space to allocate. If DFSORT cannot reasonably estimate the number of input records and FILSZ/SIZE is not specified (see FILSZ/SIZE on the OPTION statement for details), 6000 blocks are dynamically allocated.

Dynamically allocated work data sets are not deallocated until the job or step 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.

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

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

With VIO=NO: If your DFSORT program was installed with the VIO=NO option ("no virtual I/O"):

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

Default: None; optional. If DYNALLOC is specified without d, the default for d is that specified (or defaulted) by the ICEMAC DYNALOC option at installation time. If DYNALLOC is specified without n, the default for n is that specified (or defaulted) by the ICEMAC DYNALOC option at installation time.

Note: Diagnostic messages ICE806I and ICE803I give information about intermediate storage allocation/use.

Applicable Functions: See Appendix D.

EQUALS | NOEQUALS

You can use this parameter to temporarily override the installation option EQUALS={YES | NO}, that specifies whether the sequence of identical collating records for a sort or a merge should be preserved from input to output.

EQUALS
: means the sequence must be preserved.

NOEQUALS
: means the sequence need not be preserved.

When you specify EQUALS, you preserve the original sequence of the identically collating records.

When sorting, the sequence of the output depends upon the order of:

- The records from the SORTIN file
- The records inserted by an E15 user exit routine
- The E15 records inserted within input from SORTIN

When merging, the sequence of the output depends upon the order of:

- The records from a SORTINnn file.

  If two equal collating records are from two different files, for example SORTINn1 and SORTINn2, and n1 is less than n2, the record from SORTINn1 is placed before the one from SORTINn2.

- The records from an E32 user exit routine for the same file increment number.

  If two equal collating records from an E32 exit have different file increment numbers, for example m1 and m2, and m1 is less than m2, the record associated with m1 is placed before the one associated with m2.

Notes:

1. *The total number of bytes occupied by all control fields must not exceed 4088 when the EQUALS option is in effect.*
2. *Use of EQUALS can degrade performance, except when using the Blockset sort technique for variable-length records. EQUALS is always used with this technique.*
3. *EQUALS is not used if SUM is specified and a technique other than Blockset is selected.*
4. *Do not specify EQUALS if variable-length records are sorted using tape work files and the RDW is part of the control field.*

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

FILSZ=*x* | SIZE=*y* | FILSZ=*En* | SIZE=*En*

This parameter specifies an exact or estimated number of records for the sort or merge. An *EXACT* value can be used to force DFSORT to terminate with an error message if the number of records sorted or merged is not as expected.

If DFSORT cannot reasonably estimate the number of records to be sorted or merged, it uses the FILSZ or SIZE value to aid optimization of intermediate storage; otherwise, it does not use the value for this purpose. Following are the circumstances under which DFSORT uses FILSZ or SIZE (if specified) for optimization:

- An E15 exit routine is used
- Work data sets are dynamically allocated
- Input data sets are VSAM, multi-volume or on tape
- Input data sets are concatenated and Blockset is not selected

Note that Blockset never uses the *ESTIMATED* value.

FILSZ=x

x is the exact number of records to be sorted or merged; it must take into account the number of records in the input data set(s), records to be inserted or deleted by exit E15/E32, and records to be deleted by INCLUDE/OMIT, SKIPREC, and STOPAFT.

SIZE=y

y is the exact number of records in the input data set(s) (that is, the number of records in the SORTIN data set or SORTINnn data sets). It must take into account the number of records to be deleted by STOPAFT.

If the actual number of records is not the same as the specified value, the program terminates 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 or merged (Blockset will not use this estimate); it must be immediately preceded by the letter E; in either case, it should be large enough to include both the SORTIN data set or SORTINnn data sets and any records you may add at exit E15/E32.

For example, if you estimate your total data set size to be 5000 records, specify FILSZ=E5000. The program accepts either FILSZ or SIZE, but FILSZ is always preferable when its use is necessary.

If you omit the FILSZ or SIZE operand, DFSORT estimates the number of input records.

Default: None; optional.

Applicable Functions: See Appendix D.

LIST | NOLIST

You can use this parameter to temporarily override the installation option LIST={YES | NO}, which specifies whether program control statements should be listed. See Appendix H for details on use of the message data set.

LIST

means that control statements are printed.

NOLIST

means that control statements are not printed.

Note: LIST | NOLIST is processed only if it is passed on the OPTION control statement in an extended parameter list.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

MAINSIZE={*n* | *n*K | MAX}

You can use this parameter to temporarily override the installation option SIZE={MAX | n}, which specifies the amount of main storage available to DFSORT, provided the value you specify is greater than the MINLIM value set at DFSORT installation time.

For MVS/XA systems, MAINSIZE applies to the total amount of main storage above and below 16-megabyte virtual. DFSORT determines how much storage to allocate above and below 16-megabyte virtual but the total amount of storage can not exceed MAINSIZE.

For details on main storage allocation, see "Tuning Main Storage" on page 207 and "Main Storage" on page 275.

n

n is a decimal value representing the number of bytes of main storage to be allocated. You may specify a value greater than MAXLIM or TMAXLIM.

Limit: 8 digits.

*n*K

specifies n times 1024 bytes of main storage to be allocated. You may specify a value greater than MAXLIM or TMAXLIM.

Limit: 5 digits.

MAX
: instructs DFSORT to calculate the amount of main storage available and allocate this maximum amount, up to the MAXLIM (or TMAXLIM for MVS/XA systems) value set when DFSORT was installed.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

MSGDDN=*ddname*
: You can use this parameter to temporarily override the installation option MSGDDN=ddname, which specifies an alternate ddname for the message data set. MSGDDN must be in effect if:

  - A program that invokes DFSORT uses SYSOUT (for instance, COBOL uses SYSOUT) and you do not want DFSORT messages intermixed with the program messages.
  - Your E15 and/or E35 routines are written in COBOL and you do not want DFSORT messages intermixed with the program messages.
  - A program invokes DFSORT more than once and you want separate messages for each invocation of DFSORT.

  The ddname can be any 1- through 8-character name, but must be unique within the job step; do not use a name that is used by DFSORT (for example, SORTIN). If the ddname specified is not available at execution time, SYSOUT is used instead. For details on use of the message data set, see Appendix H.

Note: MSGDDN is processed only if it is passed on the OPTION control statement in an extended parameter list.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

MSGPRT={ALL | CRITICAL | NONE}
: You can use this parameter to temporarily override the installation option MSGPRT={ALL | CRITICAL | NONE}, which specifies the class of messages to be written to the message data set. For details on use of the message data set, see Appendix H.

  ALL
  : specifies that all messages except diagnostic messages (ICE800I to ICE999I) are to be printed. Control statements print only if LIST is in effect.

  CRITICAL
  : specifies that only critical messages will be printed. Control statements print only if LIST is in effect.

NONE
specifies that no messages and control statements will be printed.

Note: MSGPRT is processed only if it is passed on the OPTION control statement in an extended parameter list.

Default: Usually the installation default, but refer to Appendix D for full override details.

NOBLKSET
DFSORT uses the Blockset technique whenever possible. By use of this parameter, you can cause DFSORT to bypass the Blockset technique for a sort or merge application. However, this generally degrades performance.

Default: None; optional.

Applicable Functions: See Appendix D.

NOOUTREL
You can use this parameter to temporarily override the installation option, OUTREL=YES, which specifies that unused temporary SORTOUT data set space is to be released. NOOUTREL means that unused temporary SORTOUT data set space is *not* to be released.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

NOOUTSEC
You can use this parameter to temporarily override the installation option, OUTSEC=YES, which specifies that automatic secondary allocation should be used for SORTOUT data sets. NOOUTSEC means that automatic secondary allocation for SORTOUT data sets is *not* used.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

NOSTIMER
You can use this parameter to temporarily override the installation option, STIMER=YES, which specifies that DFSORT uses the STIMER macro. NOSTIMER means that DFSORT does not use the STIMER macro; processor time data does not appear in SMF records.

If your exit(s) take checkpoints, and STIMER=YES is the installation default, you should specify this parameter.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

NOWRKREL

You can use this parameter to temporarily override the installation option WRKREL=YES, which specifies that unused temporary SORTWKnn data set space is to be released. NOWRKREL means that no unused temporary SORTWKnn data set space will be released.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

NOWRKSEC

You can use this parameter to temporarily override the installation option, WRKSEC=YES, which specifies that automatic secondary allocation should be used for SORTWKnn data sets. NOWRKSEC means that automatic secondary allocation is not used for SORTWKnn data sets.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

RESALL={*n* | nK}

You can use this parameter to temporarily override the corresponding installation option RESALL=n. *RESALL is used only when MAINSIZE/SIZE=MAX is in effect.* It indicates the number of bytes to be reserved in a partition or REGION when the maximum amount for sorting is calculated. Usually, only 4K bytes (the standard default) of main storage has to be available in a partition or region for system use. However, in a few cases, this may not be enough; for example, if your installation does not have BSAM/QSAM modules resident, you have exits that open data set(s), or you have COBOL exits.

For MVS/XA systems, RESALL applies only to the amount of main storage below 16-megabyte virtual. The ARESALL option applies to the amount of main storage above 16-megabyte virtual.

n

n is a decimal value that specifies the number of bytes of storage to be reserved. If you specify less than 4096, 4096 is used.

Limit: 8 digits.

*n*K

nK specifies n times 1024 bytes of storage are to be reserved. If you specify less than 4K, 4K is used.

Limit: 5 digits.

Note: A better way to release the required storage for user exits is the *m* parameter on the MODS statement.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

RESINV={*n* | *n*K}

You can use this parameter to temporarily override the corresponding installation option RESINV=n. *RESINV is used only when DFSORT is dynamically invoked and MAINSIZE/SIZE=MAX is in effect.* It indicates the number of bytes to be reserved in a partition or REGION for the invoking program when the maximum amount available for processing is being calculated.

For MVS/XA systems, RESINV applies only to the amount of main storage below 16-megabyte virtual. The ARESINV option applies to the amount of main storage above 16-megabyte virtual.

This extra space is usually required for data handling by the invoking program or exits while DFSORT is executing (as is the case with some PL/I and COBOL invoked sort applications).

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.

If your invoking program and its associated exits do not perform data set handling, you do not need to specify this parameter.

n

n is a decimal value that specifies the number of bytes of main storage to be reserved.

Limit: 8 digits

*n*K

nK specifies n times 1024 bytes of main storage are to be reserved.

Limit: 5 digits

Note: A better way to release the required storage for user exits is the *m* parameter on the MODS statement.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

SKIPREC=*z*

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

A program with an input data set that exceeds intermediate storage capacity usually terminates unsuccessfully. However, for a tape sort, you can use a routine at E16 (as described in Chapter 4, "User Exit Routines" on page 135) to instruct the program to sort only those records already read in. It then prints 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.

Notes:

1. *SKIPREC applies only to records read from SORTIN (not from E15 routines). (See Figure 2 on page 10.)*

2. *If SKIPREC=0 is in effect, SKIPREC is not used.*

Default: None; optional.

Applicable Functions: See Appendix D.

SORTDD=*cccc*

You should use this parameter to specify a 4-character prefix for ddnames when you dynamically invoke DFSORT more than once in a program step. The four characters replace "SORT" in the following ddnames: SORTIN, SORTOUT, SORTINnn, SORTWKnn, and SORTCNTL.

cccc specifies a 4-character prefix. The four characters must all be alphanumeric or national ($, #, or @). The first character must be alphabetic. The first three characters must not be SYS.

Example: If you use ABC# as replacement characters, DFSORT will use DD statements ABC#IN, ABC#CNTL, ABC#WKnn, and ABC#OUT instead of SORTIN, SORTCNTL, SORTWKnn, and SORTOUT.

Notes:

1. *SORTDD is processed only if it is passed on the OPTION control statement in an extended parameter list.*

2. *If SORTIN=ddname and SORTDD=cccc are both specified, ddname is used for DFSORT input.*

3. *If SORTOUT=ddname and SORTDD=cccc are both specified, ddname is used for DFSORT output.*

Default: If this parameter is not specified, DFSORT defaults to SORT.

Applicable Functions: See Appendix D.

SORTIN=*ddname*

You can use this parameter to specify a ddname to be associated with the SORTIN data set. This allows you to dynamically invoke DFSORT more than once in a program step, passing a different ddname for each input file.

Notes:

1. *SORTIN is processed only if it is passed on the OPTION control statement in an extended parameter list.*

2. *If SORTIN=ddname and SORTDD=cccc are both specified, ddname is used for the input file. The same ddname cannot be specified for SORTIN and SORTOUT.*

3. *If SORTIN is used for a tape work data set sort, DFSORT terminates.*

Default: If this parameter is not specified, DFSORT defaults to SORTIN, unless SORTDD=cccc is specified, in which case ccccIN will be the default.

Applicable Functions: See Appendix D.

SORTOUT=*ddname*

You can use this parameter to specify a ddname to be associated with the SORTOUT data sets. This allows you to dynamically invoke DFSORT more than once in a program step, passing a different ddname for each output file.

Notes:

1. *SORTOUT is processed only if it is passed on the OPTION control statement in an extended parameter list.*

2. *If SORTOUT=ddname and SORTDD=cccc are both specified, ddname is used for the output file. The same ddname cannot be specified for SORTIN and SORTOUT.*

3. *If SORTOUT is specified for a conventional merge or for a tape work data set sort, DFSORT terminates.*

Default: If this parameter is not specified, DFSORT defaults to SORTOUT, unless SORTDD=cccc is specified, in which case ccccOUT is the default.

Applicable Functions: See Appendix D.

STOPAFT=*n*

n is the maximum number of records you want accepted for sorting or copying (that is, read from SORTIN or inserted by E15 and not deleted by SKIPREC, E15 or INCLUDE/OMIT). When n records have been accepted, no more records are read from SORTIN; E15 continues to be entered as if EOF were encountered until a return code of 8 is sent, but no more records are inserted. If end-of-file is encountered before n records are accepted, only those records accepted up to that point are sorted or copied.

Notes:

1. *STOPAFT is not used for a tape work data set sort.*

2. *If you specify FILSZ=x or SIZE=x and the number of records accepted for processing does not equal x, DFSORT issues message ICE047A and terminates.*

3. *If STOPAFT=0 is in effect, it will not be used.*

Default: None; optional.

Applicable Functions: See Appendix D.

VERIFY | NOVERIFY

You can use this parameter to temporarily override the installation option VERIFY={YES | NO}, that specifies whether sequence checking of the final output records should be performed.

VERIFY
means that sequence checking is to be performed.

NOVERIFY
means that sequence checking is not to be performed.

Note: Use of VERIFY can degrade performance.

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

VLSHRT | NOVLSHRT

You can use this parameter to temporarily override the installation option {VLSHRT=YES | NO} that specifies whether DFSORT is to continue sorting or merging if a variable-length input record is found that is too short to contain all specified control fields. VLSHRT is not meaningful for fixed-length record processing.

VLSHRT
means that sorting or merging continues if a "short" record is found.

NOVLSHRT
means that sorting or merging terminates if a "short" record is found.

Notes:

1. *VLSHRT is not used if INCLUDE/OMIT, INREC, OUTREC, and/or SUM are specified.*

2. *If Blockset is selected:*

   - *DFSORT pads "short" control fields with binary zeroes, thus making the order predictable for records with equal control fields of different lengths.*

   - *VLSHRT is not used for a merge application. To use VLSHRT for a merge application, you must specify the NOBLKSET option on the OPTION control statement.*

3. *If Blockset is not selected:*

   - *DFSORT terminates if the first byte of the first (major) control field is not included in the record.*
   - *DFSORT does not pad "short" control fields, thus making the order unpredictable for records with equal control fields of different lengths.*
   - *In certain cases, VLSHRT is not used due to the number and position of the control fields.*
   - *EQUALS is not used if VLSHRT is in effect.*

Default: Usually the installation default, but refer to Appendix D for full override details.

Applicable Functions: See Appendix D.

OPTION Statement Examples

OPTION Statement Example 1. One Control Field and Related Options

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

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 50000 records.

SKIPREC
: Five records are skipped before starting to process the input data set.

CKPT
: DFSORT takes checkpoints during this run.

  Note: CKPT is ignored if one of the Blockset techniques is chosen. If checkpoints are required, you must bypass the Blockset technique by specifying the NOBLKSET option, or by specifying IGNCKPT=NO on the ICEMAC installation macro.

EQUALS
: The sequence of equal collating records is preserved from input to output.

DYNALLOC
: One data set (by default) is allocated on SYSDA (by default). The space on the data set is calculated using the MAINSIZE/SIZE value in effect.

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

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

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
: DFSORT takes checkpoints during this run.

  Note: CKPT is ignored if one of the Blockset techniques is chosen. If checkpoints are required, you must bypass the Blockset technique by specifying the NOBLKSET option, or by specifying IGNCKPT=NO on the ICEMAC installation macro.

EQUALS
: The sequence of equal collating records is preserved from input to output.

NOCHALT
: Only AQ fields are translated through the ALTSEQ translate table. (This overrides CHALT=YES, had it 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 Example 3. Using OPTION to Override SORT

```
OPTION FILSZ=50,SKIPREC=5,DYNALLOC=3380
SORT   FIELDS=(1,2,CH,A),SKIPREC=1,SIZE=200,DYNALLOC=(3350,5)
```

This example shows how parameters specified on the OPTION control statement override those specified on the SORT control statement, regardless of the order of the two statements.

FILSZ
: DFSORT 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 is used in place of SIZE.)

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

DYNALLOC
DFSORT allocates one work data set (by default) on an IBM 3380.

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 Example 4. Bypassing the Blockset Technique

```
OPTION NOBLKSET
```

NOBLKSET
DFSORT bypasses FLR-Blockset or VLR-Blockset regardless of whether the Blockset technique can be used.

OPTION Example 5. Using STOPAFT and COBEXIT

```
OPTION STOPAFT=100,COBEXIT=COB2
```

STOPAFT
DFSORT accepts 100 records before sorting.

COBEXIT
E15 and/or E35 routines can be executed with the VS COBOL II library.

OPTION Example 6. Passing an OPTION Control Statement through a SORTCNTL or SYSIN Data Stream

```
OPTION RESINV=32000,MSGPRT=NONE,
       MSGDDN=SORTMSGS,SORTDD=ABCD,SORTIN=MYINPUT,
       SORTOUT=MYOUTPUT,NOLIST
```

This example illustrates the parameters RESINV, MSGPRT, MSGDDN, SORTDD, SORTIN, SORTOUT, and NOLIST, and the actions taken when these parameters are supplied on an OPTION statement read from the SYSIN data set or the SORTCNTL data set. The parameters are recognized, but not used.

RESINV
32000 bytes of storage are reserved for the user.

MSGPRT=NONE
The keyword is ignored, and messages are printed according to the installation-supplied default.

MSGDDN=SORTMSGS
The keyword is ignored, and all messages are written to the SYSOUT data set.

SORTDD=ABCD
The keyword is ignored, and the standard prefix SORT is used.

SORTIN=MYINPUT
The keyword is ignored, and the ddname SORTIN is used to reference the input data set.

SORTOUT=MYOUTPUT
The keyword is ignored, and the ddname SORTOUT is used to reference the output data set.

NOLIST
The keyword is ignored, and control statements are printed according to the installation-supplied defaults.

OPTION Example 7. Passing an OPTION Control Statement in the Extended Parameter List

```
OPTION RESINV=32000,MSGPRT=CRITICAL,
       MSGDDN=SORTMSGS,SORTDD=ABCD,SORTIN=MYINPUT,
       SORTOUT=MYOUTPUT,NOLIST
```

This example illustrates keywords RESINV, MSGPRT, MSGDDN, SORTDD, SORTIN, SORTOUT, and NOLIST and the actions taken when these keywords are supplied on the OPTION control statement passed by an extended parameter list.

RESINV
32000 bytes of storage are reserved for the user.

MSGPRT=CRITICAL
Only critical messages are printed on the message data set.

MSGDDN=SORTMSGS
Messages are written to the SORTMSGS data set.

SORTDD=ABCD
SORT uses ABCD as a prefix for all sort names.

SORTIN=MYINPUT
The ddname MYINPUT is used to reference the input data set.

SORTOUT=MYOUTPUT
The ddname MYOUTPUT is used to reference the output data set.

NOLIST
Control statements are not printed.

OPTION Example 8. Using COPY with the OPTION statement

```
SORT   FIELDS=(3,4,CH,A)
OPTION COPY,SKIPREC=10,CKPT
MODS E15=(E15,1024,MODLIB),E35=(E35,1024,MODLIB)
```

SORT
The sort statement is ignored because the COPY option has been specified.

COPY
The copy processing is done on a record-by-record basis. So, each record is read from SORTIN, passed to the E15 exit, passed to the E35 exit, and written to SORTOUT. (Contrast this with a sort, where all the records are read from SORTIN and passed to the E15 exit before any records are passed to the E35 exit and written to SORTOUT.)

SKIPREC
Ten records are skipped before copying starts.

CKPT
The checkpoint option is not used for copy applications.

OUTREC Control Statement

OUTREC FIELDS=([*s*,]*p*,*m*[,*a*]...[,*s*][,*p*,[*m*][,*a*]][,*s*])

The OUTREC control statement allows you to reformat the input records before they are output; that is, to define which parts of the input record are to be included in the reformatted output record, in what order they are to appear, and how they are to be aligned.

You do this by defining one or more fields from the input record. The reformatted output record consists of those fields only, in the order in which you have specified them, and aligned on the boundaries you have indicated. You can also pad reformatted output records with blanks and/or binary zeros before, between, and/or after the input fields, using the s parameter.

For information concerning the interaction of INREC and OUTREC, see "Using Options That May Enhance Performance" on page 217.

FIELDS=([*s*,]*p*,*m*[,*a*][,*s*][,*p*,[*m*][,*a*]][,*s*])
specifies the order in which the input and separation fields are to appear in the reformatted output record.

s
indicates a separation field to be inserted into the reformatted output record in the position you code it relative to the input fields. It can be specified before or after the p,m,a parameters for any field. Permissible values are:

nX Blank separation. n bytes of EBCDIC blanks (X'40') are inserted in the reformatted output records. n may be from 1 to 256.

nZ Binary zero separation. n bytes of binary zeros (X'00') are inserted in the reformatted output records. n may be from 1 to 256.

Consecutive separation fields may be specified.

For variable-length records:

- Separation field(s) must not be specified before the first input field (the RDW).
- Separation field(s) must not be specified after the variable part of the input record.

p

specifies the first byte of the input field relative to the beginning of the input record.[4] The first data byte of a fixed-length record has relative position 1. The first data byte of a variable-length record has relative position 5, because the first four bytes are occupied by the RDW. All fields must start on a byte boundary, and no field may extend beyond byte 32000. See "OUTREC Statement Notes" below for special rules concerning variable-length records.

m

specifies the length of the input field. It must include the sign if the data is signed, and must be a whole number of bytes. See note 5 on page 60 for more information.

a

specifies the alignment (displacement) of the input field in the reformatted output record, relative to the start of the reformatted output record.

The permissible values are:

H Halfword aligned. This means that the displacement (p-1) of the field from the beginning of the reformatted input record, in bytes, is a multiple of 2 (that is, position 1, 3, 5, and so forth).

F Fullword aligned. The displacement is a multiple of 4 (that is, position 1, 5, 9, and so forth).

D Doubleword aligned. The displacement is a multiple of 8 (that is, position 1, 9, 17, and so forth).

Alignment can be necessary if, for example, the data is to be used in a COBOL application program where COMPUTATIONAL items are aligned through the SYNCHRONIZED clause. Unused space preceding aligned fields are always padded with binary zeros.

Default: None; must be specified.

Applicable Functions: See Appendix D.

OUTREC Statement Notes

1. If input records are reformatted by INREC or E15, OUTREC must refer to fields in the appropriate reformatted record (see *p* above).

2. When you specify OUTREC, you should be aware of the change in record size and layout of the resulting reformatted output records. You should also understand how reformatting of records affects processing performance, and how to use INREC and/or OUTREC to achieve the most efficient processing.

---

[4] If INREC is specified, p must refer to the record as reformatted by INREC. If your E15 exit reformats the record, and INREC is not specified, p must refer to the record as reformatted by your E15 exit.

(See also "INREC Control Statement" on page 58 and "Using Options That May Enhance Performance" on page 219 for more details).

3. The length of the INREC/OUTREC record (reformatted length) is not used to determine the LRECL of SORTOUT. If not specified in the DSCB or DD statement, the value for SORTOUT LRECL will be determined in the usual way (that is, from the L3 value or SORTIN LRECL). If the reformatted length does not match the SORTOUT LRECL, the same checks used when the SORTIN LRECL does not match the SORTOUT LRECL are made and padding/truncation is performed, if possible. When processing variable-length records, the maximum SORTIN LRECL must not exceed the maximum SORTOUT LRECL.

   For VSAM data sets, the maximum record size defined in the cluster is equivalent to the LRECL when processing fixed-length records, and is four more than the LRECL when processing variable-length records. See "VSAM Data Set Notes and Limitations" on page 4 for more information.

4. For variable-length records, the first entry in the FIELDS parameter must specify or include the 4-byte RDW. DFSORT sets the length of the reformatted record in the RDW.

   If the first field in the data portion of the input record is to appear in the reformatted output record immediately following the RDW, the entry in the FIELDS parameter can specify both RDW and data field in one. Otherwise, the RDW must be specifically included in the reformatted output record.

5. The variable part of the input record (that part beyond the minimum record length) may be included in the reformatted output record as the last part. In this case, a value should be specified for p*n* that is less than or equal to the minimum record length (L4) plus 1 byte, and m*n* and a*n* should be omitted. If INREC and OUTREC are both specified, either both must specify position-only for the last part, or neither must specify position-only for the last part.

   Note that, if the reformatted input includes only the RDW and the variable part of the input record, "null" records containing only an RDW could result.

6. The reformatted output records are in the format specified by OUTREC regardless of whether INREC was specified.

7. Fields referenced in OUTREC statements may overlap each other and/or control fields.

8. If input is variable records, the output is also variable. This means that each record is given the correct RDW by DFSORT before output even if the records are treated as fixed internally because they are all the same length.

9. When OUTREC is specified, your E35 exit routine must refer to fields in the reformatted output record.

10. DFSORT issues a message and terminates if an OUTREC statement is specified for a tape work data set sort or conventional merge application.

11. When you specify OUTREC, VLSHRT is not used. If it is specified, it is ignored.

OUTREC Statement Examples

See INREC Examples 1, 3, and 4 for applications in which both INREC and OUTREC statements are used in the same job stream to improve performance.

OUTREC Example 1.

```
OUTREC FIELDS=(11,32)
```

This statement specifies that the output record should contain 32 bytes beginning with byte 11 of the input record. This statement can only be used with fixed-length input records, because it does not include the first 4 bytes.

OUTREC Example 2.

```
OUTREC FIELDS=(1,4,11,32,D,101)
```

This statement is for variable-length records of minimum length 100 bytes, and specifies that the output record should contain an RDW plus 32 bytes of the input record starting at byte 11 (aligned on a doubleword boundary, relative to the start of the record) plus the entire variable portion of the input record.

Note that no extra comma is coded to indicate the omission of the first alignment parameter. If you do include an extra comma, you get a syntax error message and the program terminates.

OUTREC Example 3.

```
OUTREC FIELDS=(1,42,D,101)
```

This statement is for variable-length records of minimum length 100 bytes, and specifies that the output record should contain an RDW plus the first 38 data bytes of the input record plus the entire variable portion of the input record.

The 'D' parameter has no effect, because the first field is always placed at the beginning of the output record.

OUTREC Example 4.

```
SORT FIELDS=(20,4,CH,D,10,3,CH,D)
OUTREC FIELDS=(5X,20,4,H,8X,20,2,10,3,1Z,1,9,13,7,24,57,6Z)
```

This example illustrates how a fixed-length input data set could be sorted and reformatted for output. The SORTIN LRECL is 80 bytes.

The reformatted output records are fixed length with a record size of 103 bytes and look as follows. The SORTOUT LRECL should be specified as 103.

Position	Contents
1-5	EBCDIC blanks
6	Binary zero (for H alignment)
7-10	Input positions 20 through 23
11-18	EBCDIC blanks
19-20	Input positions 20 through 21
21-23	Input positions 10 through 12
24	Binary zero
25-33	Input positions 1 through 9
34-40	Input positions 13 through 19
41-97	Input positions 24 through 80
98-103	Binary zeros

OUTREC Example 5.

```
SORT FIELDS=(12,4,PD,D)
RECORD TYPE=V,LENGTH=(,,,100)
OUTREC FIELDS=(1,7,5Z,5X,28,8,6X,101)
```

This example illustrates how a variable-length input data set could be sorted and reformatted for output. The variable part of the input records is included in the output records. The minimum input record size is 100 bytes and the maximum input record size (SORTIN LRECL or maximum record size for VSAM) is 200 bytes.

The reformatted output records are variable length, with a maximum record size of 131 bytes. For variable records, the maximum output record size (SORTOUT LRECL) must be equal to or greater than the maximum input record size (SORTIN LRECL), which in this case is 200. The reformatted records look as follows:

Position	Contents
1-4	RDW (input positions 1 through 4)
5-7	Input positions 5 through 7
8-12	Binary zeros
13-17	EBCDIC blanks
18-25	Input positions 28 through 35
26-31	EBCDIC blanks
32-n	Input positions 101 through n (variable part of input records)

OUTREC Example 6.

```
MERGE FIELDS=(28,4,BI,A)
OUTREC FIELDS=(1,4,5Z,5X,5,3,28,8,6Z)
```

This example illustrates how input files can be merged and reformatted for output. The variable part of the input records is not to be included in the output records. The SORTINnn LRECL is 50 bytes.

The reformatted output records are variable length, with a maximum record size of 31 bytes and look as follows. The SORTOUT LRECL must be 50 bytes.

Position	**Contents**
1-4	RDW (input positions 1 through 4)
5-9	Binary zeros
10-14	EBCDIC blanks
15-17	Input positions 5 through 7
18-25	Input positions 28 through 35
26-31	Binary zeros

RECORD Control Statement

RECORD [TYPE=*x*][,LENGTH=(L1,L2,L3,L4,L5,L6,L7)]

The RECORD control statement describes the format and lengths of the records being processed. It is required when:

- You include user exit routines that change record lengths during a DFSORT program run.
- Input is from a user data set.
- SORTIN and/or SORTOUT record length information is unavailable.
- A sort is invoked from a program written in PL/I, or
- Input is from a VSAM data set.

The RECORD control statement can also be used when sorting variable-length records to supply the minimum and average record lengths to the program.

For details of the RECORD control statement and its parameters, see also Figure 3 on page 20.

TYPE=*x*

F
indicates that the records to be processed are fixed-length records.

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

D
indicates that the records are ISCII/ASCII variable-length records.

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 Chapter 3, "Job Control Statements" on page 113), or that given on the data set label. If the formats are not the same or TYPE is not specified, the program uses the format given in the data set label/DD statement.

The TYPE operand is always required for VSAM SORTIN or SORTINnn data sets.

Default: Required for E15 or E32 input if SORTIN or SORTINnn record format is unavailable; otherwise, defaults to SORTIN or SORTINnn record format.

Applicable Functions: See Appendix D.

LENGTH=(L1,L2,L3,L4,L5,L6,L7)

This parameter is required when you change record lengths at one or more exits, or when no SORTIN DD statement is supplied.

L1

Input record length, L1, is required only when no SORTIN or SORTINnn 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 be degraded.

For VSAM input data sets, if the L1 value is not equal to the maximum record size defined in the cluster for fixed-length records, it is overridden by the cluster value. If processing variable-length records with VSAM input and non-VSAM output, the L1 value must be four more than the value defined in the cluster. See "VSAM Data Set Notes and Limitations" on page 4 for more information.

L2

L2 is the record length after E15. It is extremely important to specify an accurate value for L2 if you change record lengths at E15. Note that, except for Blockset, if you have specified a value for L1 but not for L2, the value you specified acts as a 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.

L3

Output record length, L3, can usually be supplied by default: you need to specify L3 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 inappropriate.

For VSAM SORTOUT data sets, if the L3 value is not equal to the maximum output record size defined in the VSAM cluster, it is overridden by the cluster value.

L4

Specifying the minimum record length (L4) may help performance. However, if you specify too large a value, the program fails and issues 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. L4 is not used for Blockset.

L5

L5 is the average record length for variable-length records.

L6, L7

L6 and L7 are accepted, but not used; they are reserved for future use.

Default: For defaults, see RECORD in Figure 3 on page 20.

Usual 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, provided you indicate their omission by a comma, that is, LENGTH=(,,,30,80).

Applicable Functions: See Appendix D.

RECORD Statement Examples

RECORD Example 1. Fixed-Length, Three Length Values

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

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 Example 2. Variable-Length, Five Length Values

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

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 Example 3. Variable Length, Two Length Values

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

TYPE

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

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.

SORT Control Statement

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

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 sorts.

A SORT statement can also be used to specify a copy application.

The options available on the SORT statement can be specified in other sources, as well. A table showing all possible sources for these options and the order of override is given in Appendix D.

When an option can be specified on either the SORT or OPTION statement, it is preferable to specify it on the OPTION statement.

FIELDS=(*p,m,f,s...,p,m,f,s*)
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 DFSORT by the values of the FIELDS operand, represented by p, m, f, and s in Figure 3 on page 20.

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) must not exceed 4092 bytes long (or, when the EQUALS option is in operation, 4088 bytes). As shown in Figure 3 on page 20, 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; if you prefer, you can always use the first format.

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 3 on page 20, e, m, f, and s describe the control fields. The specifications for the parameters in the SORT control statement are summarized in Figure 3. The text that follows gives these specifications in detail.

p

specifies the first byte of a control field relative to the beginning of the input record.[5]

The first data byte of a fixed-length record has relative position 1. The first data byte of a variable-length record has relative position 5 (because the first 4 bytes contain the RDW). 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.

Note that the beginning of a variable-length record must include a 4-byte record descriptor word (RDW) that precedes the actual record. This is also true for VSAM input records, for which DFSORT supplies the necessary RDW on input to the program and removes 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.

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

1. First, specify the byte location relative to the beginning of the record and follow it with a period.

2. 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.

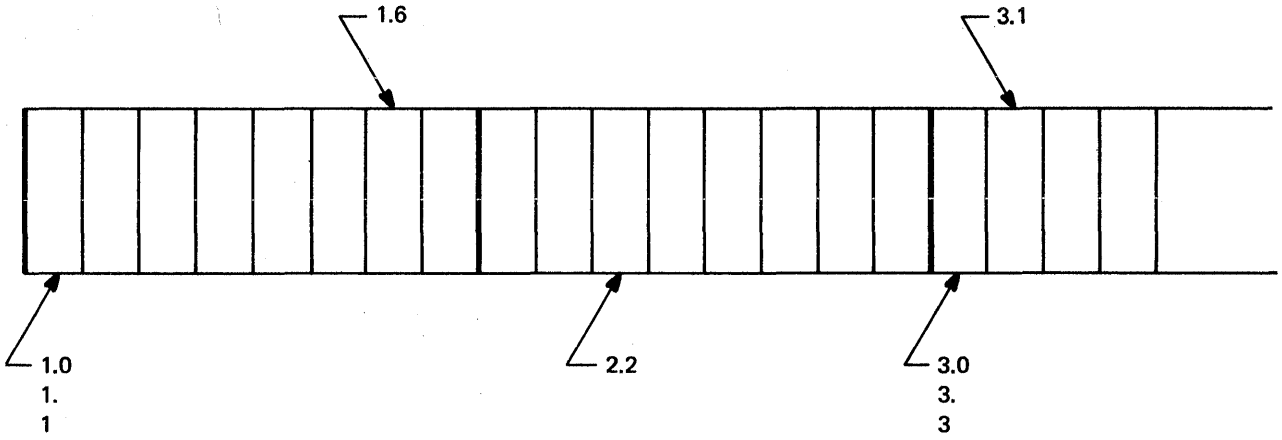
Thus, 1.0 represents the beginning of a record. A binary field beginning on the third bit of the third byte of a record is represented as 3.2. When the beginning of a binary field falls on a byte boundary (say, for example, on the fourth byte), you can write it in one of three ways:

```
4.0
4.
4
```

---

5 If INREC is specified, p must refer to the record as reformatted by INREC. If your E15 exit reformats the record, and INREC is not specified, p must refer to the record as reformatted by your E15 exit.

Other examples of this notation are:



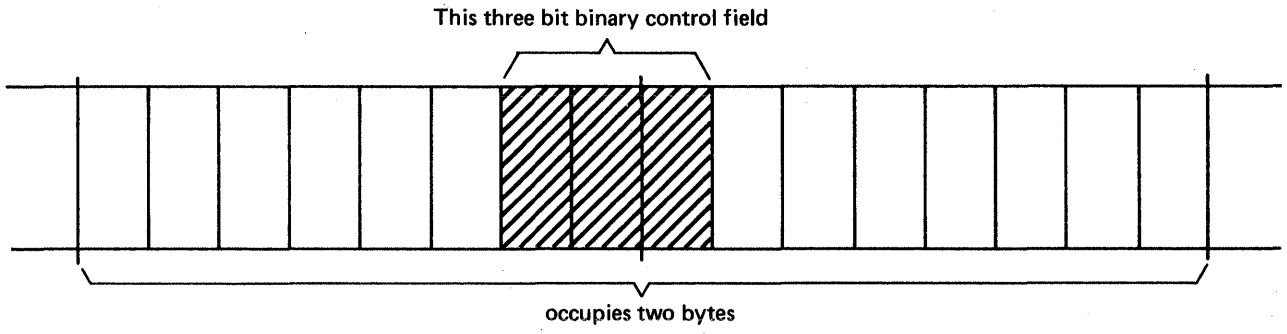
m

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.



f

specifies the format of the data in the control field. Acceptable control field lengths (in bytes) and available formats are shown in Figure 11 on page 105.

Format	Length	Description
CH	1-4092 1-256	Character EBCDIC, unsigned. If CHALT=YES is in effect, CH is treated the same as AQ.
ZD	1-32	Zoned decimal, signed.
PD	1-32	Packed decimal, signed.
FI	1-256	Fixed-point, signed.
BI	1 bit- 4092 bytes	Binary, unsigned.
FL	1-256	Floating-point, signed.
AC	1-256	Character ISCII/ASCII, unsigned.
CSL	2-256	Signed numeric, leading separate sign.
CST	2-256	Signed numeric, trailing separate sign.
CLO	1-256	Signed numeric, leading overpunch sign.
CTO	1-256	Signed numeric, trailing overpunch sign.
ASL	2-256	Signed numeric, ISCII/ASCII, leading separate sign.
AST	2-256	Signed numeric, ISCII/ASCII, trailing separate sign.
AQ	1-256	Character EBCDIC, alternate collating sequence.

Figure 11. Control Field Formats/Lengths

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 F, "EBCDIC and ISCII/ASCII Collating Sequences" on page 301, 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, DFSORT collates them using the format(s) specified [6] and ascending order.

For information on how to add a user routine to modify a control field, see Chapter 4, "User Exit Routines" on page 135.

Default: None; a parameter must be specified.

Applicable Functions: See Appendix D.

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 3 on page 20.

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.

If you have specified the COPY operand, FORMAT=*f* cannot be specified.

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

Applicable Functions: See Appendix D.

FIELDS=COPY

See the discussion of the COPY parameter on the OPTION statement.

CKPT

See the discussion of this operand on the OPTION statement.

DYNALLOC[={*d* | (*d*) | (,*n*) | (*d*,*n*)}**] (MVS Only)**

See the discussion of this parameter on the OPTION statement.

EQUALS | NOEQUALS

See the discussion of this parameter on the OPTION statement.

FILSZ=*x* | SIZE=*y* | FILSZ=*En* | SIZE=*En*

See the discussion of this parameter on the OPTION statement.

---

6 With a conventional merge or a tape work data set sort, control fields for which E is specified are treated as binary byte format regardless of the actual format(s) specified.

SKIPREC=*z*

See the discussion of this parameter on the OPTION statement.

SORT Statement Note

If the records are reformatted by INREC or E15, SORT must refer to fields in the appropriate reformatted record (see the description for "p" following "FIELDS" above).

SORT Statement Examples

SORT Example 1. One Control Field and File Size Option

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

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 29483 records.

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

```
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)
```

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 were not normalized, you would specify E instead of A

and include your own routine to normalize the field before the program examined it.

FILSZ
The data set to be sorted contains exactly 10693 records.

CKPT
Instructs the program to take checkpoints during this run.

Note: If the Blockset technique is chosen, the CKPT option is ignored. If checkpoints are required, the Blockset technique can be bypassed by specifying either IGNCKPT=NO on the ICEMAC installation macro or NOBLKSET on the OPTION statement.

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

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

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

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 is 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 is sorted into descending sequence.

FILSZ
The input data set contains approximately 30000 records.

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

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

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 sequence of equal collating records is to be preserved from input to output.

SORT Example 5. COPY Option

```
SORT    FIELDS=COPY
```

FIELDS

The input data set is copied to the output data set without sorting or merging.

SUM Control Statement

```
SUM   {FIELDS=(p,m,f...,p,m,f) |
       FIELDS=(p,m...,p,m),FORMAT=f |
       FIELDS=NONE}
```

The SUM control statement specifies that, whenever two records are found with equal control fields, the contents of their summary fields are to be added, the sum is to be placed in one of the records, and the other record is to be deleted.

FIELDS=(*p,m,f...p,m,f*)
designates numeric fields in the input record as summary fields.

p
specifies the first byte of the field relative to the beginning of the input record.[7] The first data byte of a fixed-length record has relative position 1. The first data byte of a variable-length record has relative position 5, as the first four bytes are occupied by the RDW. All fields must start on a byte boundary, and no field may extend beyond byte 4092.

m
specifies the length in bytes of the summary fields to be added. The value must include the sign, if signed data. See below for permissible length values.

f
specifies the format of the data in the summary field, which can only be of the following types:

Code	Length	Description
BI	2, 4, or 8 bytes	Binary, unsigned
FI	2, 4, or 8 bytes	Fixed-point, signed
PD	1-16 bytes	Packed decimal, signed
ZD	1-18 bytes	Zoned decimal, signed

NONE
eliminates records with duplicate keys. Only one record with each key is kept and no summarization is performed.

Default: None; must be specified.

---

7 If INREC is specified, p must refer to the record as reformatted by INREC. If your E15 exit reformats the record, and INREC is not specified, p must refer to the record as reformatted by your E15 exit.

Applicable Functions: See Appendix D.

FORMAT=*f*
can be used when all the summary fields contain the same type of data. The values for f are listed above.

Default: None; optional.

Applicable Functions: See Appendix D.

SUM Statement Notes

1. If input records are reformatted by INREC or E15, SUM must refer to fields in the appropriate reformatted record (see the description of *p* above).

2. The size of the routine generated by DFSORT to handle the SUM function is dependent on how many fields are referenced, and what lengths and formats they have. The size of the routine must not exceed 4096 bytes or DFSORT issues a message and terminates.

3. Summary fields must not be control fields. They must not overlap control fields, or each other, and must not overlap the RDW.

4. Floating-point fields must not be summarized.

5. When records are summarized, the choice of which record is to receive the sum (and be retained) and which record is to be deleted is unpredictable unless EQUALS is in effect and the BLOCKSET technique is being used. In this case, the first record (based on the sequence described under EQUALS | NOEQUALS on page 77) is chosen.

6. Fields other than summary fields remain unchanged and are taken from the record that receives the sum.

7. If overflow occurs, the two records involved are left unsummarized (that is, the contents of the records are left undisturbed and neither record is deleted). Overflow does not prevent further summary.

8. DFSORT issues a message and terminates if a SUM statement is specified for a tape work data set sort or conventional merge.

9. Summation of data with invalid sign or digit codes results in a data exception (0C7 ABEND).

SUM Statement Examples

SUM Example 1.

```
SUM FIELDS=(41,8,ZD,49,4,FI)
```

This statement designates an 8-byte zoned decimal field at byte 41, and a 4-byte fixed-integer field at byte 49, as summary fields.

SUM Example 2.

```
SUM FIELDS=(41,8,49,4),FORMAT=FI
```

This statement illustrates the use of the FORMAT operand. The statement designates two fixed-integer fields, one 8 bytes long starting at byte 41, and the other 4 bytes long starting at byte 49.

Chapter 3. Job Control Statements

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

The job control statements required for a program application include a JOB statement, an EXEC statement, and several DD statements. 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 want to use program exits for routines of your own. If you intend to use system macro instructions or program exits, or both, you should be familiar with the material in Chapters 4 and 5. These statements, their functions, and the order in which they are arranged in the system input stream are shown in Figure 12.

While reading this chapter, you may need the appropriate JCL reference manual for supplementary information; you should have it available for reference.

//jobname	JOB	Always needed.
Preceding job steps, if any.		
//stepname	EXEC	Always needed.
		The following DD statements can be in any order:
//STEPLIB	DD	(Or JOBLIB) Omit when using a supplied cataloged procedure.
//SORTLIB	DD	Only needed for a sort using tape work files or a merge using the conventional technique. Omit when using a supplied cataloged procedure.
//SYSOUT	DD	Usually needed for messages. Omit when using a supplied cataloged procedure.
//DDname	DD	Library definition if you use routines from a library.
//SORTIN	DD	Usually needed for sort or copy processing. For a merge, the SORTINnn DD statements must be used instead.
//SORTOUT	DD	Usually needed.

Figure 12 (Part 1 of 2). Input Job Stream

```
//SORTWKnn   DD      Not needed for a sort in main storage, a merge, or a copy.
                     Must not be included if you want dynamic allocation.
                     (The ddname SORTDKnn is used by the program instead of
                     SORTWKnn if it carries out dynamic reallocation.)

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

//SORTDIAG   DD      Only needed for debugging.

//SYSUDUMP           (Or SYSABEND or SYSMDUMP) Required only if a
                     dump is needed.

//SORTMODS   DD      Needed if you have user exit routines in SYSIN.

//SYSPRINT   DD      Needed if you are using the linkage editor.
                     Omit when using the supplied SORT cataloged procedure.

//SYSUT1     DD      Needed if you are using the linkage editor.
                     Omit when using the supplied SORT cataloged procedure.

//SYSLIN     DD      Needed if you are using the linkage editor.
                     Omit when using the supplied SORT cataloged procedure.

//SYSLMOD    DD      Needed if you are using the linkage editor.
                     Omit when using the supplied SORT cataloged procedure.

//SORTCNTL   DD *    Needed if DFSORT is dynamically invoked and
                     you want to define a data set from which additional or changed
                     DFSORT control statements can be read.

//SYSIN      DD *    Needed if DFSORT is JCL invoked or needed to
                     contain user exit routines to be link-edited by
                     DFSORT, in object deck format.
  SORT or MERGE
  OPTION
  RECORD
  MODS
  INREC
  OUTREC
  INCLUDE or OMIT
  SUM
  ALTSEQ
  DEBUG
  END

/*
```

Figure 12 (Part 2 of 2). Input Job Stream

Notes to Figure:

1. *All DFSORT control statements, except for the END statement, can be specified in any order.*

2. *Either a SORT or MERGE control statement is always required unless COPY is specified on the OPTION statement. See Chapter 2 for an explanation of when the other statements are needed.*

JOB Statement

The JOB statement is the first JCL statement of your job. It must contain a valid job name 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 information, programmer's name, and so forth.

EXEC Statement

The EXEC statement is either the first JCL statement of each job step, or of each procedure step in a cataloged procedure. It identifies DFSORT to the operating system. You may also specify DFSORT options on the EXEC statement.

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.

The format of the EXEC statement is:

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

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.

If you are using a cataloged procedure, specify PROC=SORT | SORTD. You may omit PROC= and specify simply SORT or SORTD; however, PROC= serves as a reminder that a cataloged procedure is being used.

"SORT" Cataloged Procedure

You can use the supplied SORT cataloged procedure when you include user routines that require link-editing. To use this procedure without using link-edited user routines is inefficient because the SORT cataloged procedure allocates linkage editor data sets, whether or not you include user 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
```

Line	Explanation
00	The stepname of the procedure is SORT. This EXEC statement initiates the program, which is named ICEMAN.
10	The STEPLIB DD statement defines the data set containing the sort/merge program modules that reside in a link library. The data 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 modules needed for a sort using tape work files or a merge that uses the conventional technique. The data set is cataloged, and the data set name represented by xxx was specified at operation 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	Defines SYSPRINT 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 3600 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

You can use the supplied SORTD cataloged procedure when you:

- Do not include user routines, or
- 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
```

Line **Explanation**

00 The stepname of the SORTD procedure is SORT.

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 modules needed for a sort using tape work files or a merge that uses the conventional technique. 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='options'

Certain DFSORT options can be specified on the PARM parameter when DFSORT is JCL invoked. They can be specified in any order. For full override details and applicability, see Appendix D.

PARM='[ARESALL={*n* | *n*K}]
[,BSAM]
[,LIST | ,NOLIST]
[,E15=COB]
[,E35=COB]
[,MSGDDN=*ddname*]
[,MSGPRT={ALL | CRITICAL | NONE}]
[,RESALL={*n* | *n*K}]
[,SIZE={*n* | *n*K | MAX | MAX-*m* | MAX-*m*K}]'

ARESALL={*n* | *n*K}
You can use this parameter to temporarily override the installation option, ARESALL=n | nK. For an explanation of this parameter, see the ARESALL parameter of the OPTION statement.

- ARESALL=n, where n is a decimal value representing the number of bytes of storage to be reserved.

  Limit: 8 digits.

- ARESALL=nK, where nK specifies n times 1024 bytes of storage are to be reserved.

  Limit: 5 digits.

BSAM
For disk processing the EXCP access method is normally used for SORTIN and SORTOUT. If you encounter a problem related to this with I/O activity, you can temporarily bypass it by specifying BSAM.

This option is ignored if VSAM SORTIN or SORTOUT data sets are specified.

LIST | NOLIST
You can use this parameter to temporarily override the installation option, LIST={YES | NO}, that specifies whether program control statements should be listed. See Appendix H for full details on use of the message data set.

- LIST means that all DFSORT control statements are printed on the message data set.

- NOLIST specifies that control statements are not printed.

E15=COB
You can use this parameter to specify that your E15 routine is written in COBOL. It can temporarily override the MODS statement for E15. If you specify E15=COB but do not identify an E15 module by a MODS statement, the E15=COB is ignored.

E35=COB
You can use this parameter to specify that your E35 routine is written in COBOL. It can temporarily override the MODS statement for E35. If you specify E35=COB but do not identify an E35 module by a MODS statement, the E35=COB is ignored.

MSGDDN=*ddname*

You can use this parameter to temporarily override the installation option MSGDDN=ddname. For an explanation of this parameter, see the MSGDDN parameter of the OPTION statement.

The ddname can be any 1- through 8-character name, but must be unique within the job step; do not use a name that is used by DFSORT (for example, SORTIN). If the ddname specified is not available at execution time, SYSOUT is used instead. For details on using the message data set, see Appendix H.

MSGPRT={ALL | CRITICAL | NONE}

You can use this parameter to temporarily override the installation option, MSGPRT={ALL | CRITICAL | NONE}, that specifies the class of messages to be written to the message data set. See Appendix H, "DFSORT Messages and Codes" on page 311, for full details on use of the message data set

- ALL means that all messages except diagnostic messages ICE800I to ICE999I are to be printed on the message data set. Control statements are printed only if LIST is in effect.

- CRITICAL means that only critical messages are to be printed on the message data set. Control statements are printed only if LIST is in effect.

- NONE means that no messages or control statements are to be printed.

For compatibility reasons, the forms FLAG(I) | FLAG(U) | NOFLAG, and MSG={NO | AP | AC | CC | CP | PC} are also accepted.

Following is the MSGPRT/MSGCON equivalence for these options:

Option	MSGPRT	MSGCON
MSG=NO	NONE	NONE
MSG=AP	ALL	CRITICAL
MSG=AC	NONE	ALL
MSG=CC	NONE	CRITICAL
MSG=CP	CRITICAL	CRITICAL
MSG=PC	ALL	ALL
NOFLAG	NONE	CRITICAL
FLAG(I)	ALL	CRITICAL
FLAG(U)	CRITICAL	CRITICAL

RESALL={*n* | *n*K}

You can use this parameter to temporarily override the installation option RESALL=n | nK. For an explanation of this parameter, see the RESALL parameter of the OPTION statement.

- RESALL=n, where n is a decimal value representing the number of bytes of storage to be reserved, when SIZE/MAINSIZE=MAX is in effect. If you specify less than 4096, 4096 is used.

  Limit: 8 digits.

- RESALL=nK, where nK specifies n times 1024 bytes of storage are to be reserved, when SIZE/MAINSIZE=MAX. If you specify less than 4K, 4K is used.

  Limit: 5 digits.

Note: For MVS/XA systems, RESALL applies only to the amount of main storage below 16-megabyte virtual. The ARESALL option applies to the amount of main storage above 16-megabyte virtual.

SIZE={n | nk | MAX | MAX-m | MAX-mK}

You can use this parameter to temporarily override the installation option SIZE=MAX | n. For an explanation of this parameter, see the MAINSIZE parameter of the OPTION statement.

- SIZE=n, where n is a decimal value representing the number of bytes of main storage to be allocated.

- SIZE=nK, where nK specifies n times 1024 bytes of main storage are to be allocated. The value n may be 1 through 5 digits.

- SIZE=MAX, which instructs the program to calculate the amount of main storage available and allocate this maximum amount, up to the MAXLIM value (or the TMAXLIM value for an MVS/XA system) set when DFSORT was installed.

- SIZE=MAX-m, where m is the RESALL value. MAX-m instructs the program to calculate the amount of storage available and allocate this amount up to the MAXLIM value (or the TMAXLIM value for an MVS/XA system) *minus* the amount of storage reserved for system and application use (RESALL).

- SIZE=MAX-mK, where mK (m times 1024) is the RESALL value. The value m may be 1 through 5 digits. MAX-mK instructs the program to calculate the amount of storage available and allocate this amount up to the MAXLIM value (or the TMAXLIM value for an MVS/XA system) *minus* the amount of storage reserved for system and application use (RESALL).

Note: 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 also accepts the formats SIZE (option) and CORE (option).

DD Statements

A number of DD statements must be provided after the EXEC statement. Some are system DD statements, and are usually 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" on page 124 and "Program DD Statements" on page 125, respectively.

The DD statement parameters, the conditions under which they are required, and the default values, are summarized in Figure 13. The subparameters of the DCB parameter (a DD Statement parameter) are described similarly in Figure 14 on page 123. Performance is enhanced if the LRECL subparameter of the DCB is accurately specified for variable-length records. The maximum input record length you can specify for your particular configuration is given in "Input and Output Data Sets" on page 3.

When using DFSORT applications, FREE=CLOSE cannot be used on any DD statements.

Parameter	Condition Under Which Required	Summary of Parameter Values	Default Value
DSNAME or DSN	When the DD statement defines a labeled input data set (for example, SORTIN), or when the data set being created is to be kept or cataloged (for example, 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 14.)
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.	

Figure 13 (Part 1 of 2). DD Statement Parameters Used by DFSORT

Parameter	Condition Under Which Required	Summary of Parameter Values	Default Value
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 13 (Part 2 of 2). DD Statement Parameters Used by DFSORT

Shared Tape Units

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

- The input data set and the first intermediate storage data set (SORTWK01)
- The input data set and the output data set

If you want 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.

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 tape was recorded.	800 bpi
TRTCH	When the data set is located on a 7-track 2400-series tape unit.	Specifies the technique used to record 8-bit bytes on a 7-track type.	Converter not used, translator not used, odd parity.
RECFM	When the DCB parameter is required and the default value is not suitable, except on SORTWKnn statements.	Specifies the format of the records in the data set.	• For old data sets, the value in the data set label. • For a new SORT-OUT data sets, the same as for the first SORTIN or SORTINnn data set. • 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 ISCII/ASCII format.	Specifies that the tape processed is in ISCII/ASCII format.	
BUFOFF	When processing data in ISCII/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, except under some circumstances, padding (on the right with binary zeros) for sort or copy applications or truncation (from the right) for sort and merge applications occurs if the record length of the output data set is different from that of the input data set. However, if user exits modify records, they are responsible for the padding and truncation.

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

Figure 14. DCB Subparameters Used by DFSORT

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 DFSORT, such as SORTLIB.) The DD statements contained in the cataloged procedure (or provided by you) are:

//JOBLIB DD or

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

//SYSIN DD contains DFSORT control statements when DFSORT is not invoked by another program. It can also contain user exit routines to be link-edited by DFSORT, in object deck format. The control data set usually 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. SYSIN cannot be concatenated data sets.

If user exit routines are in SYSIN, make sure that:

- The END statement is the last *control* statement.
- The user exit routines are arranged in numeric order (for example, E11 before E15).
- The user exit routines are supplied immediately after the END control statement.
- Nothing follows the last object deck in SYSIN.
- A SORTMODS DD statement is included.

If you are invoking DFSORT dynamically, and you supply the DFSORT control statements through the 24-bit or extended parameter list and/or through SORTCNTL, SYSIN still remains the source of user exit routines placed in the system input stream.

//SYSOUT DD identifies the system output data set for messages. Always use this statement if a cataloged procedure is not used. If you are invoking DFSORT from another program, you can specify an alternate ddname for the message data set. (If you are invoking DFSORT from a COBOL program and using no ddname other than SYSOUT, the use of EXHIBIT or DISPLAY in your COBOL program can give uncertain printing results.) Before printing DFSORT messages, a skip to a new page is performed.

//SYSUDUMP DD or

//SYSABEND DD defines output from a system ABEND dump routine. Needed only for debugging.

If you are using the supplied SORT cataloged procedure, the four DD statements mentioned below are automatically supplied. If you are not using the SORT cataloged procedure and you are using the linkage editor, you must supply the following four DD statements:

//SYSPRINT DD	contains messages from the linkage editor.
//SYSUT1 DD	is a work area for the linkage editor.
//SYSLIN DD	defines a data set in which DFSORT places control information for the linkage editor.
//SYSLMOD DD	defines a data set that contains output from the linkage editor.

Note: If you do not include user routines or you include user routines that do ***not*** require link-editing, you can use the supplied SORTD cataloged procedure. If you include user routines that require link-editing, you can use the SORT cataloged procedure.

Program DD Statements

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

//SORTLIB DD	defines the data set that contains special load modules for DFSORT. Only needed for a tape work data set sort or a conventional merge.
//SORTIN DD	defines the input data set for a sorting or copying application. Cannot be used for merging applications.
//SORTINnn DD	defines the input data sets for a merging application. Cannot be used for sorting or copying applications.
//SORTWKnn DD	defines intermediate storage data sets. Usually needed for a sorting application unless dynamic allocation is requested. Must not be used for a merging application. Will not be used for a copying application.
//SORTOUT DD	defines the output data set for a sorting, merging, or copying application.
//SORTCKPT DD	defines a data set for checkpoint records. This statement is not required, if you are not using the checkpoint facility.
//SORTCNTL DD	defines the data set from which additional or changed DFSORT control statements can be read, when DFSORT is invoked from another program.
//SORTDKnn DD	defines the data set given to a VIO SORTWKnn allocation by DFSORT if it is dynamically reallocated (MVS only) and should never be specified in the job stream.

//SORTDIAG DD specifies that all messages and control statements will be printed. Needed only for debugging.

//SORTMODS DD defines a temporary partitioned data set. This temporary data set must be large enough to contain all your exit routines that appear in SYSIN for a given application. If none of your routines appear in SYSIN, this statement is not required. If your routines are in libraries, you must include DD statements defining the libraries.

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

SORTLIB DD Statement

The SORTLIB DD statement describes the data set that contains special DFSORT load modules.

When Required: A SORTLIB DD statement is only required for:

- Sort applications using tape work data sets
- Merge applications for which Blockset cannot be used (see message ICE800I)

Example 1. SORTLIB DD Statement

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

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 13 on page 121. The subparameters of the DCB parameter are described similarly in Figure 14 on page 123. For more detailed information, see your JCL reference manual.

SORTIN DD Statement

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

Note: FREE=CLOSE cannot be specified.

When Required: A SORTIN DD statement is required for all sort or copy applications, unless you provide an E15 exit that supplies all input to DFSORT, and you include a RECORD statement in the program control statements. The SORTIN DD statement is ignored if your program invokes DFSORT and passes the address of your E15 exit in the parameter list.

Data Set Characteristics: DFSORT accepts an empty (not a null) QSAM data set for sorting or copying (be sure to supply DCB parameters), but an empty VSAM data set causes a VSAM input error (code 160), and DFSORT terminates. Note that a null QSAM data set is a data set that has been opened for input, but no records have been written into it, and it has not been closed successfully.

See "Input and Output Data Sets" on page 3 for additional considerations.

The following rules apply to concatenated data sets:

1. RECFM must be the same for all data sets in the concatenation, except that FB and FBS can be mixed.
2. BLKSIZE may vary, but the data set with the largest block size must be specified on the first DD statement of the concatenation.
3. 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.
4. If the data sets are on unlike devices you cannot use the EXLST parameter at exit E18.

Example 2. SORTIN DD Statement

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

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.

Example 3. Volume Parameter on SORTIN DD

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

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.

When Required: 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.

Data Set Characteristics: Input data sets can be either QSAM or VSAM, but not both. Concatenated data sets are not supported. For a conventional merge, the statements must be numbered in ascending order: SORTIN01 is the name of the first, SORTIN02 the name of the second, and so forth; no numbers can be skipped.

The data set with the largest block size must be defined in the first SORTINnn 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 the first SORTINnn DD statement.

DFSORT accepts an empty (not a null) QSAM data set for merging, but an empty VSAM data set causes a VSAM input error (code 160), and DFSORT terminates.

See "Input and Output Data Sets" on page 3 for additional considerations.

Note: FREE=CLOSE cannot be specified.

Example 4. SORTIN01-03 DD Statements (Merge)

```
//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)
```

Example 5. SORTIN01-02 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
```

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.

If more than 32 SORTWKnn DD statements are specified, only the first 32 are used.

Note: FREE=CLOSE cannot be specified.

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

- Input can be contained in main storage (except for Blockset under certain conditions), or
- DYNALLOC has been specified in the SORT or OPTION statement under MVS. No SORTWKnn data sets should be provided if dynamic allocation is specified.

For information on how to calculate the amount of storage needed, see "Intermediate Storage" on page 275.

See OPTION FILSZ | SIZE for further considerations.

Diagnostic message ICE803I gives information on intermediate storage allocation/use.

Devices: SORTWKnn data sets can be on disk or on tape, but not both, as described in "Intermediate Storage" on page 275. 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 preferable for performance reasons. DFSORT reallocates temporary SORTWKnn data sets in cylinders (MVS).
  - With disk work areas, nn can be any decimal number from 00 through 99 and numbers can be in any order; however, if more than 32 are specified, only the first 32 are used.
  - Unless the input file is very large, one or two SORTWKnn data sets are usually sufficient. One or two large SORTWKnn data sets are preferable to several small ones.
  - With tape work areas, nn can be from 01 through 32; the first must be 01, and the rest must follow consecutively. No numbers can be skipped. At least three SORTWKnn data sets are required for tapes.
- DD DUMMY must not be used.
- Different SORTWKnn DD statements must not refer to the same physical data set.
- No parameters relating to ISCII/ASCII data should be included, because ISCII/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, and NOSECALL is not in effect, a default of one-fourth of primary space or one cylinder is 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.)
- If the data set is allocated to VIO, there is 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; that is, SORTWKnn must *not* be a multivolume data set.

- If primary space is fragmented, then all but the first fragment are handled as secondary space.

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 DFSORT was installed the VIO option was specified, then virtual I/O is used and performance is degraded.

Example 6. SORTWK01 DD Statement, Disk Intermediate Storage

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 is not lost:

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

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

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

Example 7. SORTWK01 DD Statement, Tape Intermediate Storage

```
//SORTWK01  DD  UNIT=3400-3,LABEL=(,NL)
//SORTWK02  DD  UNIT=3400-3,LABEL=(,NL)
//SORTWK03  DD  UNIT=3400-3,LABEL=(,NL)
```

This is an example of SORTWKnn DD statements using three tape devices.

If DFSORT terminates unsuccessfully and the above DD statements have been specified, the intermediate storage data sets remain in the system until the step has been successfully rerun or until the data sets have been deleted by some other means.

These parameters specify unlabeled data sets on three 3400 series tape units. Because the DSNAME parameters are omitted, the system assigns unique names.

SORTOUT DD Statement

The SORTOUT DD statement describes the characteristics of the data set in which the processed records are to be placed, and indicates its location.

Note: FREE=CLOSE cannot be specified.

When Required: A SORTOUT DD statement is always required, unless you provide an E35 exit that disposes of all output. The SORTOUT DD statement is ignored if your program invokes DFSORT and passes the address of your E35 exit in the parameter list.

Data Set Characteristics: See "Input and Output Data Sets" on page 3.

Notes:

1. *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.*

2. *OPTCD=W should not be specified for an IBM 3480 tape unit. If it is specified for a full function 3480 tape unit, the request is overridden. If it is specified for a 3480 operating in 3420 compatibility mode (specified as 3400-9), the request is not overridden, but performance may be degraded.*

Example 8. SORTOUT DD Statement

```
//SORTOUT DD  DSN=C905460.OUTPT,UNIT=3350,SPACE=(CYL,5),
//            DISP=(NEW,CATLG)
```

DCB The DCB parameters default to those of SORTIN.

DISP The data set is unknown to the operating system (NEW), and it is to be cataloged (CATLG) under the name C905460.OUTPT.

DSNAME The data set is to be called C905460.OUTPT.

SPACE Five cylinders of storage are requested for the data set.

UNIT Indicates that the data set is on a 3350 unit.

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.

Example 9. SORTCKPT DD Statement

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

When allocating the SORTCKPT data set, at least one intermediate storage data set is required.

If the CKPT operand is specified on the OPTION or SORT control statement, more intermediate storage may be required. See "Intermediate Storage" on page 275.

If you want to use the checkpoint/restart facility, refer to *Checkpoint/Restart.*

SORTCNTL DD Statement

The SORTCNTL data set may be used to read changed and/or additional DFSORT control statements, when DFSORT is invoked from another program (written, for example, in COBOL or PL/I). (For override rules, see Appendix D.)

Example 10. SORTCNTL DD Statement

```
//SORTCNTL DD  *
```

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

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 DDname SORTDKnn. The DD name SORTDKnn is reserved for use by DFSORT.

SORTDIAG DD Statement

The SORTDIAG DD statement specifies that all messages, including diagnostic messages (ICE800I through ICE999I), and control statements are to be written to the message data set. The statement can be used for all DFSORT techniques, and provides information on EXCP counts, intermediate storage allocation/use, and so on.

The SORTDIAG DD statement has no effect on console messages.

The statement is intended as a *diagnostic tool.*

When SORTDIAG is used for a JCL invoked DFSORT, a SYSOUT DD statement must be provided. For a dynamically invoked DFSORT, a SYSOUT DD statement or a ddname DD statement (where ddname is the alternate message data set ddname specified at installation or execution time) must be provided.

Note: If neither an alternate message data set ddname statement nor a SYSOUT ddname statement is provided, DFSORT terminates with a return code of 20. If a job using the tape sort or Conventional merge technique terminates unsuccessfully and SORTDIAG DD has been specified, a system 0C1 abend results.

Example 11. SORTDIAG DD Statement

```
//SORTDIAG DD  DUMMY
```

Chapter 4. User Exit Routines

At certain places in the executable code of DFSORT, control can be passed to your own routines. You can write routines to perform a variety of functions, such as deleting, inserting, altering, and summarizing records. The places where control is passed to your routines are called user exits.

User exit routines can be written in any language that provides the ability to:

- Pass and accept the address of the following in register 1:
  - A record
  - A fullword of zeros
  - A parameter list
- Pass a return code in register 15

Note: PL/I routines must use the special subroutine facilities of the PL/I language.

In addition, certain user exit routines can be written in COBOL using a special interface.

In this chapter we discuss only routines written in Assembler or COBOL.

DFSORT Program Phases

Because each exit is located in a particular phase of DFSORT, you should have a general understanding of the phases involved. A phase is a large component of DFSORT designed to perform a specific task (such as writing the output file). The phases containing user exits are the input and output phases.

Input Phase

The input phase is used only for a sort or copy. For a sort, the 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, DFSORT attempts to choose the most efficient. All sorting techniques use this phase. In the Peerage, Vale, and Blockset sorting techniques, indexes are created for these distributed records.

A disk sort can usually operate with no intermediate storage if the input data set can be contained in the main storage available. A copy never requires intermediate storage.

Output Phase

The output phase has two uses:

- It makes the final merge pass of a sorting application, thus creating the output data set.
- It merges the input data sets for a merging application to create the output data set.

During a copy application, this merge phase does not apply. Instead E15 is entered for *each* record, then the record is put to the output phase.

After execution of this phase, DFSORT returns control to the operating system (or invoking program).

Functions of Routines at User Exits

Figure 15 on page 138 and Figure 16 and Figure 17 on page 139 summarize the functions of user exit routines, and the exits and phases with which they may be associated.

DFSORT Input/Exit/Output Logic Examples

Figure 15 on page 138 gives examples of the logic flow for sort, copy, and merge applications as it relates to SORTIN(nn), E15, E35, and SORTOUT. The intent is to show how your E15 and E35 routines fit into the logic of an application. All possible paths are not covered. For simplicity, it is assumed that all of the applicable data sets and exits are present and that records are not inserted or deleted. (For a merge, similar logic would be used if an E32 supplied the records rather than SORTIN(nn) data sets.)

The figures illustrate the following logic:

- E15 and E35 routines continue to be entered until they pass back a return code of 8. If your exit passes a return code of 8 to DFSORT and there are still input records to be processed, the records are processed *without* being passed to your exit.
- **Sort**: Each record is read from SORTIN and passed to E15. When *all* of the records have been processed in this manner, they are sorted. Then, each sorted record is passed to E35 and written to SORTOUT.
- **Copy**: Each record is read from SORTIN, passed to E15 and E35, and written to SORTOUT.

- **Merge**: Initially, one record is read from each SORTINnn data set. The record to be output is chosen, passed to E35, and written to SORTOUT. The chosen record is then replaced by reading a record from the same SORTINnn data set and the process continues.

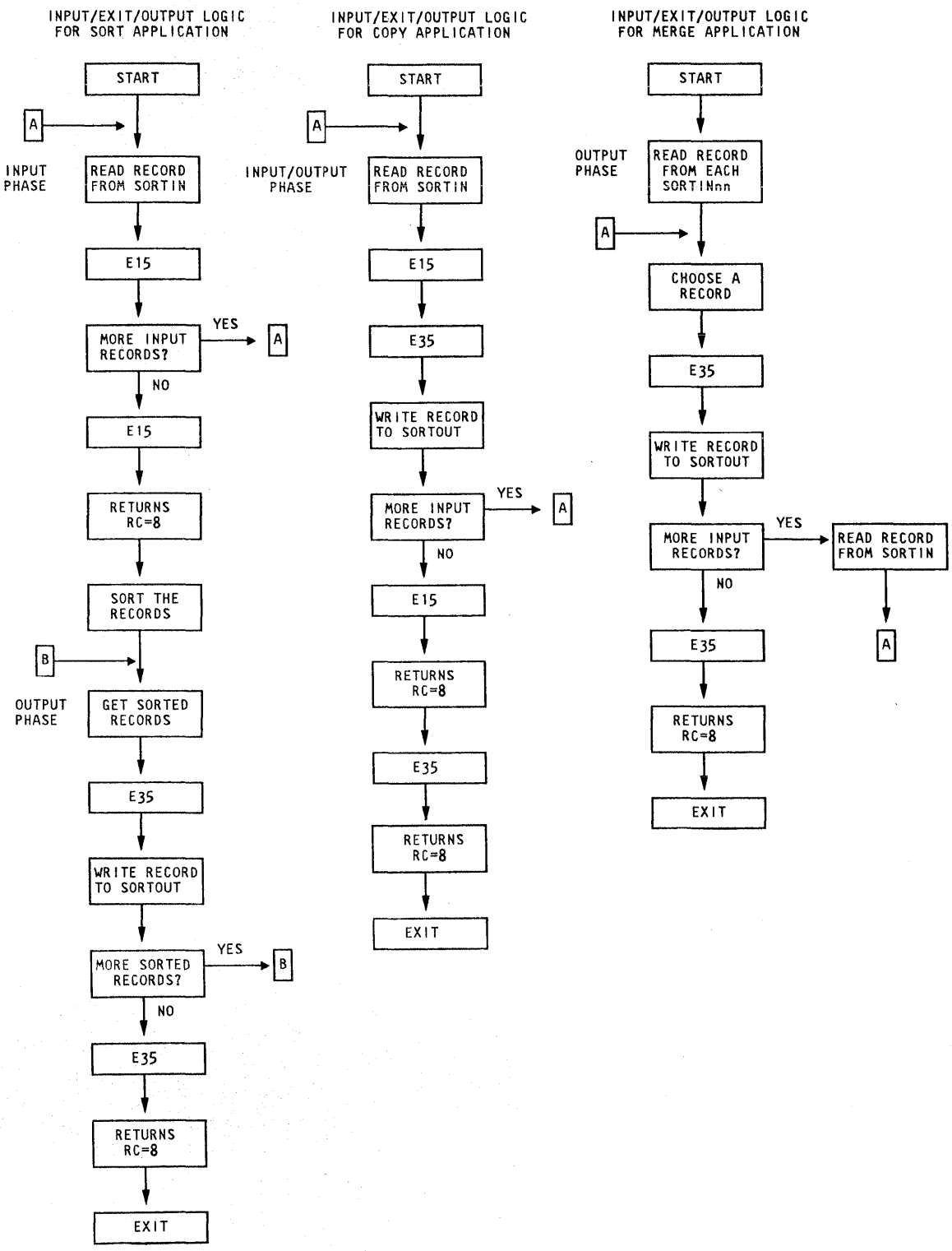


Figure 15. Examples of DFSORT Input/Exit/Output Logic

Functions	Sort Input Phase	Sort Output Phase
Open/Initialization	E11, E15	E31
Insert, Delete/Alter	E15	E35
Terminate DFSORT	E15	E35
Summarize records	E35	E35[1]
Determine action when intermediate storage is insufficient	E16[2]	N/A
Handle special I/O conditions:		
QSAM/BSAM and VSAM input	E18	E38[2]
QSAM/BSAM output	E19[2]	E39
VSAM output	N/A	E39
Modify control fields	E61	N/A
Close/housekeeping	E15, E17	E35, E37

Figure 16. Functions of Routines at Program Exits (Sort)

Functions	Copy	Merge
Open	E31, E15	E31
Insert	E15, E35	E32, E35
Delete/Alter	E15, E35	E35
Terminate DFSORT	E15, E35	E32, E35
Summarize records	E35	E35[1]
Handle special I/O conditions:		
QSAM/BSAM and VSAM input	E38	E38
QSAM/BSAM and VSAM output	E39	E39
Modify control fields	N/A	E61
Close/housekeeping	E35, E37	E35, E37

Figure 17. Functions of Routines at Program Exits (Copy and Merge)

Notes to Figure 16 and Figure 17:

1 The SUM control statement may be used instead of your own routine to summarize records.

2 Only applies to a tape work data set sort.

Opening Data Sets and Initializing

You can write your own routines to open data sets and perform other forms of initialization; you must associate these routines with the E11, E15, E31 and/or E35 exits.

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

Inserting, Deleting, and Altering Records, Terminating DFSORT

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

Note: DFSORT also provides INCLUDE and OMIT statements which automatically include or delete records based on your field criteria. For more information on these control statements, refer to Chapter 2, "Program Control Statements" on page 17.

Summarizing Records

You can summarize records in the output data set, using the E35 exit. However, you can also summarize records by using the SUM control statement described in Chapter 2.

Determining Action when Intermediate Storage Is Insufficient

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

For more details, see "Exceeding Intermediate Storage Capacity" on page 277.

Handling Special I/O

Read/Write Error Routines

DFSORT contains four exits to handle special I/O conditions: E18 and E38 for input, and E19 and E39 for output. They are particularly useful for a tape sort. With all disk sorts, E19 and E38 are ignored.

You can use these exits to incorporate your own or your installation's I/O error recovery routines into DFSORT. Your read and write error routines must reside in a partitioned data set (library). Your library routines are brought into main storage with their associated phases. When DFSORT encounters an uncorrectable I/O error, it passes the same parameters as those passed by QSAM/BSAM or VSAM. If no user routines are supplied, and an uncorrectable read or write error is encountered, DFSORT issues an error message 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 DFSORT.
- General register 15 contains the address of your error routine.

VSAM will go directly to any routine specified in the EXLST macro you passed to DFSORT via the E18, E38 or E39 exits, as appropriate. Your routine must return to VSAM via register 14. For details, see *VSAM Programmer's Guide* or *VSAM User's Guide*.

Read Errors Routines: You must associate these routines with the E18 and/or E38 exits. They must pass certain control block information back to DFSORT 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 must associate these routines with the E19 and/or E39 exit. These routines can perform any necessary abnormal end-of-task operations before DFSORT is terminated.

VSAM Exit Functions

There are three exits that can be used with VSAM files to supply passwords or an exit list to journal a VSAM data set, and carry out other VSAM exit functions (except EODAD). The exits are E18 for sort input, E38 for merge or copy input, and E39 for output.

Modifying Control Fields

You can write a routine to alter control fields before DFSORT 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 modifies 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 E15, E17, E35, and/or E37 exit.

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

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

Reserving Storage for Exits

You may have to reserve space to be used by your exits. See the options RESALL and RESINV.

MVS/XA Support of User Exits

To allow user exits called by Blockset, Peerage, or Vale (executing in an MVS/XA system) to reside above or below 16-megabyte virtual, and use either 24-bit or 31-bit addressing, DFSORT supplies these features:

- To ensure that DFSORT enters your user exit with the correct addressing mode, you must observe these rules:
  - If the exit name is specified in a MODS control statement, the exit is entered with the addressing mode indicated by the linkage editor attributes of the routine (for example, 31-bit addressing in effect if AMODE 31 is specified).
  - If the address of the exit is passed to DFSORT (preloaded exit) via the 24-bit list, the exit is entered with 24-bit addressing in effect.
  - If the address of the exit is passed to DFSORT via the extended parameter list (preloaded exit), the exit is entered with 24-bit addressing in effect if bit 0 of the exit address in the list is 0, or with 31-bit addressing in effect if bit 0 of the exit address in the list is 1.
- User exits may return to DFSORT with either 24-bit or 31-bit addressing in effect. The return address that DFSORT placed in register 14 must be used.
- Except for the user exit address constant (which is passed to either the assembler E15 or E35 exit unchanged), DFSORT handles the user exit parameter list addresses (that is, the pointer to the parameter list and the addresses in the parameter list) as follows:
  - If the exit is entered with 24-bit addressing in effect, DFSORT passes clean (zeros in the first 8 bits) 24-bit addresses to the exit. Such an exit must pass 24-bit addresses back to DFSORT. These must be clean 24-bit addresses if the exit returns to DFSORT with 31-bit addressing in effect.
  - If the exit is entered with 31-bit addressing in effect, DFSORT passes clean 24-bit addresses to the exit. Such an exit must pass 31-bit addresses or clean 24-bit addresses back to DFSORT. The only exception is when the high-order byte is used to identify an optional address being passed (for example, E18 SYNAD address). In this case DFSORT cleans the 24-bit address.

Assembler Exit Routines

To determine whether a particular exit can be used for your application, refer to Figure 16 and Figure 17 on page 139. For example, E15 cannot be used for a merge application.

Input Phase Exits

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 input phase, or to initialize your other routines. Return codes are not used.

Note: To avoid special linkage editor requirements (see "Summary of Rules for User Exit Routines" on page 183), you can include these functions in your E15 routine rather than in a separate E11 routine.

E15 Exit, Passing or Changing Records for Sort and Copy Applications

If you write your E15 routine in COBOL, see "COBOL Exit Routines" on page 161, and "COBOL E15 Exit, Passing or Changing Records for Sort" on page 164.

DFSORT enters the E15 exit routine each time a new record is brought into the input phase. DFSORT continues to enter E15 (even when there are no input records) until the exit tells DFSORT, with a return-code of 8, not to return.

See Figure 15 on page 138 for logic flow details.

Some uses for E15 are:

- Add records to an input data set.
- Pass an entire input data set to DFSORT.
- 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 variable-length records, you must be sure they contain a 4-byte record descriptor word (RDW) at the beginning of each record before the routine passes it to DFSORT. The format of an RDW is described in *Data Management Services* or *System Programming Library: Data Management.* (Alternatively, you could declare the records as fixed length, and pad them to the maximum length.)

Notes:

1. *If you use the E15 exit to pass all your records to DFSORT, the SORTIN DD statement may be omitted, in which case you must include a RECORD statement in the program control statements.*

2. *If you invoke DFSORT from an Assembler program, and pass the address of your E15 exit in the parameter list:*

   - *DFSORT ignores the SORTIN data set.*
   - *DFSORT terminates if you specify E15 in a MODS statement.*

3. *If the SORTIN DD statement is omitted or ignored, all input records are passed to DFSORT through your routine at E15: the address of each input record in turn is placed in register 1, and you return to DFSORT with a return code of 12. When DFSORT returns to the E15 exit after the last record has been passed, return to DFSORT with return code of 8 in register 15 to indicate "do not return."*

4. *DFSORT continues to re-enter your E15 exit until a return code of 8 is received. However, if STOPAFT is in effect, no additional records are inserted to DFSORT (even if you pass back a return code of 12) after the STOPAFT count is satisfied.*

5. *Remember to build an RDW for variable-length VSAM records (see* Data Management Services*).*

Information DFSORT Passes to Your Routine

The routine at E15 is entered each time a new record is brought into the input phase. DFSORT passes two words to your routine each time it is entered:

- The address of the new record. If there are no records in the input data set, this address is zero the first time your E15 is entered. When DFSORT reaches the end of the input data set, it sets this address to zero before entering your E15 exit.

  After the end of the input data set is encountered, DFSORT will continue to enter your exit routine until you pass back a return code of 8.

- The user exit address constant. If you invoked DFSORT with a user exit address constant in the parameter list, it is passed in this word to your E15 exit the first time it is entered. This word may be changed by your E15 exit any time it is entered; it is passed along on subsequent entries to your E15 exit and also on the first entry to your E35 exit. As an example, you could obtain a dynamic storage area, use it in your E15 exit, and pass its address to your E35 exit.

In general register 1, DFSORT places the address of a parameter list that contains the record address and the user address constant.

The list is two fullwords long and begins on a fullword boundary. The format of the parameter list is:

Bytes 1 through 4
Address of the new record
User exit address constant

Return Codes

Your routine must pass one of the following return codes to DFSORT, 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 DFSORT

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

0—Record Altered
If you want to change the record before passing it back to DFSORT, your routine must move the record into a work area, perform whatever modification you want, 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 DFSORT on a RECORD statement. (For details of the RECORD statement, see "RECORD Control Statement" on page 98 and *Supervisor Services and Macro Instructions* for further information about the length indicator and the record descriptor word.)

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

8—Do Not Return
DFSORT 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 DFSORT application. You need not place an address in register 1 when you return with return code 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.

If your exit passes a return code of 8 to DFSORT and there are still input records to be processed, the records are processed *without* being passed to your exit.

12—Insert Record

If you want DFSORT 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 DFSORT with a return code of 12. DFSORT will return 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 DFSORT places a zero address in the parameter list). *DFSORT keeps returning to your routine until you pass a return code of 8.*

16—Terminate DFSORT

If you want to terminate DFSORT, return with a code of 16. DFSORT then returns to its calling program or to the system with a return code of 16.

E16 Exit, Handling Intermediate Storage Miscalculation

For a tape 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 disk sort, because DFSORT uses the WRKSEC option to determine whether secondary allocation is allowed. See "SORTWKnn DD Statement" on page 129. See also "Exceeding Intermediate Storage Capacity" on page 277.

Note: When using magnetic tape, remember that the system uses an assumed tape length of 2400 feet. If you use tapes of a different length, the Nmax figure is not 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 DFSORT:

0 Sort Current Records Only
4 Try to Sort Additional Records
8 Terminate DFSORT

0—Sort Current Records Only

If you want DFSORT to continue with only that part of the input data set it estimates it can handle, return with a return code of 0. Message ICE054I contains the number of records with which sort is continuing. 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 DFSORT to continue with all of the input data set, return with a return code of 4. Enough space may be available for DFSORT to complete processing, if tapes are used. If enough space is not available, DFSORT generates a message and terminates. Refer to "Exceeding Intermediate Storage Capacity" on page 277.

8—Terminate the DFSORT

If you want DFSORT to terminate, return with a return code of 8. DFSORT 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 the input 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.

Note: To avoid special linkage editor requirements (see "Summary of Rules for User Exit Routines" on page 183), you can include these functions in your E15 routines rather than in a separate E17 routine.

E18 Exit, Handling Input Data Sets

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 DFSORT. Your E18 exit routine can also pass a fourth DCB field (EODAD) to DFSORT.

Note: If you are using a disk sorting technique, the EROPT option is ignored.

Your routines are entered first at the beginning of each phase so that DFSORT 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, DFSORT enters your E18 exit routine early in the sort (input) phase. DFSORT picks up the parameter list, including the EXLST address. Later in the phase, DFSORT enters your routine again at the EXLST address when the data set is opened.

Information Your Routine Passes to DFSORT

Before returning control to DFSORT, 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 are 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	00	00	EROPT code
04	EODAD field		
00	00	00	00

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 *not* 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 accomplish 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 through 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 through 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 *Data Management Macro Instructions*.

Use with VSAM

If input to DFSORT 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 the initialization phase when processing under Blockset and early in the input phase if Blockset is not selected.

Restrictions: If passwords are to be entered through an exit and the Blockset is not selected, the data set cannot be opened during the initialization phase. This means that MAINSIZE | SIZE=MAX must not be used, because the program cannot make the necessary calculations.

Information Your Routine Passes to DFSORT

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

Byte 1	**Bytes 2 through 4**
05	Address of VSAM exit list
06	Address of password list
00	000000

If QSAM parameters are passed instead, they are 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 a 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 is destroyed by DFSORT. This list should not be altered at any time during the program. MAINSIZE | 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 instruction giving the addresses of your routines handling VSAM exit functions. VSAM branches directly 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 DFSORT.

For details, see *VSAM Programmer's Guide* or *VSAM User's Guide*.

Below is an example of code your program can use to return control to DFSORT.

```
          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      X'000080'          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       DC      X'85',AL3(RTN)     EXLST ADDRESS LIST*
RTN       ...                        EXLST ROUTINE
QSAMEOD   ...                  QSAM END OF FILE ROUTINE
```

* X'85' = X'80' plus X'05', where:

1. X'80' means this entry is the LAST ENTRY of the list.

2. X'05' means this exit is the data control block exit

For more information, refer to *OS/MVS Data Management Services Guide.*

E19 Exit, Handling Output to Work Data Sets

This exit is used to handle write error conditions in the input phase when DFSORT is unable to correct a write error to a work data set. It cannot be used if a disk sorting technique is used; if supplied, it is ignored.

Use with QSAM/BSAM

Your routines at this exit can pass to DFSORT a parameter list containing the specifications for two DCB fields (SYNAD and EXLST).

Your routines are entered first early in the input phase so that DFSORT 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 DFSORT

Before returning control to DFSORT, 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	00	00	00

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 accomplish 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 *Data Management Macro Instructions*.

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 Chapter 2. After your routine modifies the control field, DFSORT collates the records in ascending order using the format(s) specified.[8]

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 want to merely 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 Chapter 2).

Information DFSORT Passes to Your Routine

DFSORT 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 bytes 2 through 4 of the second word; and the length of the control field (in hexadecimal) in 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	Control Field No.
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 DFSORT 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; all bytes which contain **any bits** of the control field are passed. If the control field is greater than 256 bytes in length, DFSORT splits it up into fields of 256 bytes each and passes them one at a time to your routine.

---

[8] With a conventional merge or a tape work data set sort, control fields for which E is specified are treated as binary byte format regardless of the actual format(s) specified.

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 DFSORT. The field your routine returns to DFSORT must contain the same number of bytes as when the routine was entered.

When E61 is used, records are always ordered into ascending sequence. If you need some other sequence, you can modify the fields further; for example, if after carrying out your planned modification for a binary control field, and before handing back control to DFSORT, you reverse all bits, the field is in effect collated in descending order. You have not affected the record itself, since it is only an extracted image you are modifying.

Note that if E61 is used to resolve ISCII/ASCII collating for special alphabetic characters, substituted characters must be in EBCDIC, but the sequencing result depends upon the byte value of the ISCII/ASCII translation for the substituted character.

Output Phase Exits

E31 Exit, Opening Data Sets

You might use routines at this exit to open data sets needed by your other routines in the output phase, or to initialize your other routines. Return codes are not used.

Note: To avoid special linkage editor requirements (see "Summary of Rules for User Exit Routines" on page 183), you can include these functions in your E35 routine rather than in a separate E31 routine.

E32 Exit, Handling Input to a Merge Only

This exit can only be used in a merge 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 or an OPTION statement in SORTCNTL must indicate the number of input files.

If input is variable-length records, you must be sure they contain a 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 *Data Management Services Guide*. (Alternatively, you could declare the records as fixed length, and pad them to the maximum length.)

See Figure 15 on page 138 for logic flow details.

Information DFSORT Passes to Your Routine

Your E32 exit routine is entered each time the merge program requires a new input record. DFSORT passes a 2-word parameter list to your routine. The address of the list is in register 1.

The parameter list has the format:

Bytes 1 through 4
Increment of next file to be used for input
Address of next input record

The file increment is 0,4,8,...,N−4, where N is four times the number of input files. So, the increment 0 (zero) would represent the first input file, 4 the second file, 8 the third, and so on.

A separate input buffer must be provided by your routine for each input file used. An input buffer containing the first record for a file must not be altered until you have passed the first record from each file to DFSORT.

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 DFSORT:

8 End of the Data Set Requested (No Record Returned)
12 Insert Record
16 Terminate DFSORT

E35 Exit, Changing Records

If you write your E35 routine in COBOL, see "COBOL Exit Routines" on page 161, and "COBOL E35 Exit, Changing Records" on page 172.

The E35 routine is entered each time DFSORT prepares to place a record in the output area.

See Figure 15 on page 138 for logic flow details.

Some uses are:

- Add, delete, or change records in the output data set.
- Terminate DFSORT.

Notes:

1. *If you use the E35 routine to dispose of all your output records, the SORTOUT DD statement may be omitted.*
2. *If you invoke DFSORT from an assembler program and you pass the address of your E35 routine in the parameter list:*

- *DFSORT ignores the SORTOUT data set*
- *DFSORT terminates if you specify E35 in a MODS statement.*

3. *If the SORTOUT DD statement is omitted or ignored, your E35 exit routine must dispose of each output record and return to DFSORT with a return code of 4. When DFSORT returns to your routine after you have disposed of the last record, return to DFSORT with return code of 8 to indicate "do not return."*

4. *If your E35 routine is inserting variable length records, you must be sure they contain a 4-byte record descriptor word (RDW) at the beginning of each record before the routine passes it to DFSORT. The format of an RDW is described in Data Management Services or System Programming Library: Data Management. (Alternatively, you could declare the records as fixed-length, and pad them to the maximum length.)*

5. *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).*

6. *Once records have been put into the output area, their lengths may not be increased.*

Information DFSORT passes to Your Routine

Your E35 exit routine is executed each time DFSORT prepares to place a record (including the first record) in the output area. DFSORT passes three words to your routine:

- The address of the record leaving DFSORT which usually follows the record in the output area. When DFSORT reaches the end of the input data set, it sets this address to zero before entering your E35 exit.

  After the end of the input data set is encountered, DFSORT continues to enter your exit routine until you pass back a return code of 8.

- 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 remains zero provided you pass a return code of 4 (delete record) to DFSORT.

  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.

- The user exit address constant. This word is passed to your exit exactly as it was set by your E15 exit or invoking program's parameter list.

In general register 1, DFSORT places the address of a parameter list that contains the two record addresses and the user exit address constant.

The list is three fullwords long and begins on a fullword boundary. The format of the parameter list is:

Bytes 1 through 4
Address of record leaving DFSORT
Address of record in output area
User exit address constant

Return Codes

Your routine must pass one of the following return codes to DFSORT to inform it what to do with the record leaving DFSORT:

0 No Action/Record Altered
4 Delete Record
8 Do Not Return
12 Insert Record
16 Terminate DFSORT

0—No Action

If you want DFSORT to retain the record unchanged, load the address of the record leaving DFSORT in register 1 and return to DFSORT with a zero return code.

0—Record Altered

If you want to change the record before having it placed in the output data set, move the record to a work area, make the change, load the address of the modified record into register 1, and return to DFSORT with a zero return code. If you change record size, you must communicate that fact to DFSORT in a RECORD statement.

4—Delete Record

Your routine can delete the record leaving DFSORT by returning to DFSORT with a return code of 4. You need not place an address in register 1.

8—Do Not Return

DFSORT 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 DFSORT application. When you return with return code 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 a return code of 8 when DFSORT indicates the end of the data set*, which it does by passing your routine zero as the address of the record leaving DFSORT.

If you do not have a SORTOUT data set and would usually return with a return code of 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).

If your exit passes a return code of 8 to DFSORT and there are still input records to be processed, the records are processed *without* being passed to your exit.

12—Insert Record
If you want to add a record to the SORTOUT data set before the record leaving DFSORT, place the address of the new record in register 1 and return to DFSORT with a return code of 12. DFSORT returns to your routine with the same address as before for the record leaving DFSORT, 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 DFSORT. DFSORT does not perform sequence checking for disk sorts. For tape sorts, DFSORT does not perform sequence checking on records that you insert unless you delete the record leaving DFSORT and insert a record to replace it. *DFSORT keeps returning to your routine until you pass a return code of 8.*

16—Terminate DFSORT
If you want to terminate DFSORT, return with a code of 16. DFSORT then returns to its calling program or the system with a return code of 16.

Summarizing Records

You can use the SUM control statement to summarize records.

However, you can summarize records in the output data set by changing the record in the output area and then, if you want, by deleting the record leaving DFSORT. DFSORT returns to your routine with the address of a new record leaving DFSORT and the same record remains in the output area, so that you can summarize further. If you do not delete the record leaving DFSORT, that record is added to the output area, and its address replaces the address of the previous record in the output area; DFSORT returns with the address of a new record leaving DFSORT.

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.

Note: To avoid special linkage editor requirements (see "Summary of Rules for User Exit Routines" on page 183), you can include these functions in your E35 routine rather than in a separate E37 routine.

E38 Exit, Handling Input Data Sets

The routine here is the same as for E18. If you are using a disk sorting technique, then I/O error conditions cannot be handled through E38.

Use with VSAM

This exit can be used during a merge or copy to insert VSAM passwords into VSAM input ACBs and to perform various VSAM exit functions. The example below shows code your program can use to return control to DFSORT.

```
          ENTRY    E38
          .
          .
E38       LA       1,PARMLST
          RETURN
          CNOP     0,4
PARMLST   DS       0H
          DC       X'05'
          DC       AL3(VSAMEXL)
          DC       X'06'
          DC       AL3(PWDLST)
          DC       A(0)
          .
          .
VSAMEXL   EXLST    SYNAD=USYNAD,LERAD=ULERAD
PWDLST    DC       H'3'
          DC       CL8'SORTIN01'     SORTIN01 DDNAME
          DC       CL8'INPASS1'      SORTIN01 PASSWORD
          DC       CL8'SORTIN02'     SORTIN02 DDNAME
          DC       CL8'INPASS2'      SORTIN02 PASSWORD
          DC       CL8'SORTOUT'      SORTOUT DDNAME
          DC       CL8'OUTPASS'      SORTOUT PASSWORD
USYNAD    ...                        VSAM SYNCH ERROR RTN
ULERAD    ...                        VSAM LOGIC ERROR RTN
```

E39 Exit, Handling Output Data Sets

Same as for E19 for QSAM/BSAM.

Use with VSAM

For VSAM, this exit can be used to insert VSAM passwords into VSAM output ACBs and to perform various VSAM exit functions. The example below shows code your program can use to return control to DFSORT.

```
          ENTRY   E39
          .
          .
E39       LA      1,PARMLST
          RETURN
          CNOP    0,4
PARMLST   DS      0H
          DC      X'05'
          DC      AL3(VSAMEXL)
          DC      X'06'
          DC      AL3(PWDLST)
          DC      A(0)
          .
          .
VSAMEXL   EXLST   SYNAD=USYNAD,LERAD=ULERAD
PWDLST    DC      H'1'
          DC      CL8'SORTOUT'      SORTOUT DDNAME
          DC      CL8'OUTPASS'      SORTOUT PASSWORD
USYNAD    ...                       VSAM SYNCH ERROR RTN
ULERAD    ...                       VSAM LOGIC ERROR RTN
```

Sample Routines Written in Assembler

E15: Deleting Expired Records

This routine checks each record's expiration date, and deletes records that 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,C'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 DFSORT 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: Deleting Records

This routine checks byte 5 of each record. If the byte contains the letter 'N', it deletes the record. You could use the INCLUDE or OMIT control statements instead.

```
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
        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
NOINPUT RETURN  (14,12),RC=8    RETURN WITH 'DO NOT RETURN' CODE
DELETE  RETURN  (14,12),RC=4    RETURN WITH 'DELETE' CODE
        END
```

COBOL Exit Routines

E15 and E35 exit routines written in COBOL can perform the same functions as E15 and E35 exit routines written in Assembler. However, the information passed between DFSORT and the COBOL routine is handled differently than for Assembler. These differences are:

- Your COBOL routine must use fields described in the LINKAGE SECTION of the DATA DIVISION, instead of register 1 and pointers in a parameter list.
- Your COBOL routine must use RETURN-CODE (a COBOL special register) instead of register 15 for the return code.

- Your COBOL routine must use return code 20 when you want to alter or replace a record instead of return code 0.
- Your COBOL routine must use the exit area instead of the user address constant.

COBOL Exit Requirements

The following rules apply to COBOL exits. Failure to observe these COBOL exit rules may result in termination or unpredictable results.

- If both E15 and E35 exits are used, they must be in the same version of COBOL.
- Exits written in COBOL must not use STOP RUN statements. To return to DFSORT, you have to use the GOBACK statement.
- VS COBOL II exits must be compiled with the RES/RENT compile-time option.
- Compilation of OS/VS COBOL exits with the RES compiler option aids migration to VS COBOL II; however, exits compiled with NORES execute under DFSORT.
- If an exit contains a READY TRACE, EXHIBIT, or DISPLAY statement, the DFSORT messages normally written to SYSOUT should be directed to another data set using the MSGDDN parameter. For READY TRACE, EXHIBIT, and DISPLAY statements, COBOL writes also to SYSOUT. The messages to SYSOUT could, therefore, be lost because of interleaving of output.

  Another alternative is to direct the COBOL output to another data set, using the SYSx compiler option for OS/VS COBOL or the OUTDD compiler option for VS COBOL II.
- COBOL exits must not contain a SORT or a MERGE verb.
- If invoking DFSORT from a VS COBOL II program, you can use a COBOL E15 if the VS COBOL II FASTSRT option is in effect for input, and/or a COBOL E35 if FASTSRT is in effect for output. The COBOL exits must be compiled with VS COBOL II.
- If you are running with VS COBOL II exits, you must use the VS COBOL II library. If COBEXIT=COB2 is not the default for your installation, make sure you specify the COB2 parameter in the OPTION control statement. Failure to do so results in degraded performance.
- If you run exits compiled with either COBOL compiler and you specify the RES option, the COBOL library routines must be available at execution time. The COBOL library **may** be required for an exit compiled with the OS/VS COBOL, NORES option. See your OS/VS COBOL manual for information on options that may require the COBOL library.
- Exits compiled with OS/VS COBOL may be executed with either the OS/VS COBOL or VS COBOL II library, or in some cases, with no library.

- Exits compiled with VS COBOL II must be executed with the VS COBOL II library.
- Exits compiled with OS/VS COBOL and executing with the VS COBOL II library should not issue STAEs. (OS/VS COBOL compiler options that cause STAE to be issued are: STATE, FLOW, SYMDMP, COUNT, and TRACE.)

Requirements for Copy Processing

For copy processing, all sort requirements apply except for the following restrictions:

- When DFSORT is invoked through JCL and COBEXIT=COB2, *either* a separately compiled COBOL E15 exit **or** a separately compiled COBOL E35 exit is allowed, but not both together.
- When DFSORT is invoked from a VS COBOL II program, the following limitations apply when FASTSRT is in effect for:
  - input only: a separately compiled E15 exit is allowed, but not a separately compiled E35 exit
  - output only: a separately compiled E35 exit is allowed, but not a separately compiled E15 exit
  - input and output: **either** a separately compiled E15 **or** a separately compiled E35 is allowed, but not both together (when COBEXIT=COB2)

  If separately compiled E15 and E35 exits are found together, DFSORT copy processing terminates. Message ICE161A is issued.

Storage Requirements

If you are running the COBOL exits compiled with the RES compiler option, make sure that you have enough storage available for the COBOL library subroutines. (This does not apply if the library has been installed resident.)

Besides the minimum DFSORT main storage requirements, you need an additional 40K bytes of storage in your REGION for the OS/VS COBOL library subroutines, and 150K bytes for the VS COBOL II library subroutines. Most of the VS COBOL II library subroutines can be resident above 16-megabyte virtual. However, whether you can actually load the VS COBOL II library subroutines above 16-megabyte virtual depends on how they were installed.

Under certain conditions, DFSORT can use all the storage in your REGION below 16-megabyte virtual, (true for both MVS and MVS/XA systems) thus leaving no room to load the COBOL library subroutines required during execution of your exit.

On an MVS/XA system, main storage is available above 16-megabyte virtual unless the TMAXLIM or SIZE/MAINSIZE options specify an extremely high value (for example, your system limit for main storage above 16-megabyte virtual), in which case you can use the ARESALL or ARESINV option to release storage.

During execution, the actual amount of storage required for the COBOL library subroutines depends on the functions performed in the COBOL exit. You should add to the size of the exit a minimum of 40K bytes when running with the OS/VS COBOL library subroutines and, in most cases, 20K bytes when running with the VS COBOL II library subroutines. If the exit does I/O, additional storage must be reserved for the I/O buffers. (See Note below for additional circumstances under which you may need to release additional storage for VS COBOL II.) This value is specified by the *m* parameter on the MODS statement. A VS COBOL II exit requires less storage, because DFSORT automatically releases storage for some of the COBOL library subroutines before the exit is called.

When SIZE/MAINSIZE=MAX is in effect, an alternative way to release storage is to use the RESALL or RESINV option.

Note: When you are calling both exits (E15 and E35), when running with nonresident VS COBOL II library subroutines, and executing a sort under MVS or MVS/XA with DFSORT residing above 16-megabyte virtual, you may need to release an additional 70K bytes of storage. This can be done by adding 70K bytes more to one of the following:

- The m parameter of the MODS statement for the E35 exit (m = E35 exit size + 20K + 70K)
- The RESALL option when SIZE/MAINSIZE=MAX is in effect

Input Phase Exit

COBOL E15 Exit, Passing or Changing Records for Sort

DFSORT continues to enter E15 (even when there are no input records) until the exit tells DFSORT, with a return code of 8, not to return.

See Figure 15 on page 138 for logic flow details.

Some uses for E15 are:

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

Notes:

1. *If both E15 and E35 exits are used, they must be in the same version of COBOL.*
2. *If you use the E15 exit, the SORTIN DD statement may be omitted, in which case you must include a RECORD statement in the program control statements.*

3. *If you omit the SORTIN DD statement, all input records are passed to DFSORT through your COBOL E15 exit. You return to DFSORT with a return code of 12. When DFSORT returns to the E15 exit after the last record has been passed, return to DFSORT with return code 8 to indicate "do not return."*

4. *DFSORT continues to re-enter your E15 exit until a return code of 8 is received. However, if STOPAFT is in effect, no additional records are inserted to the sort after the STOPAFT count is satisfied.*

5. *You cannot use dynamic link-editing together with a COBOL E15 exit.*

Interface with COBOL

Each time the E15 exit is called, DFSORT supplies the following fields:

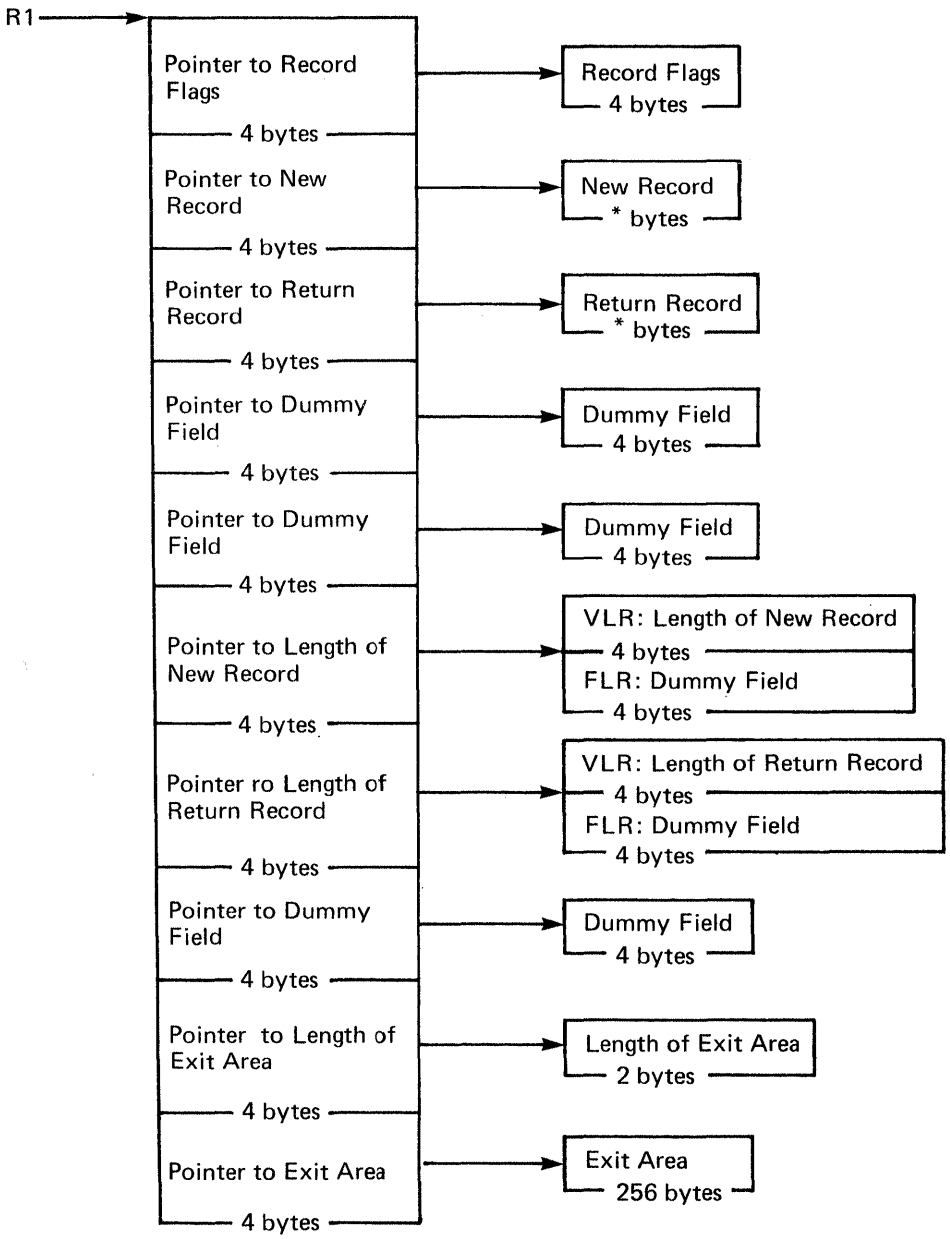
- Record flags
- New record
- Length of the new record (for VLR)
- Length of exit area
- Exit area

When E15 returns to DFSORT, the E15 exit provides to DFSORT some or all of the fields mentioned below. The first field is required; the other four may be modified as appropriate.

- RETURN-CODE (assigned by the exit by setting the COBOL special register RETURN-CODE)
- Return record
- Length of the return record (for VLR)
- Length of exit area
- Exit area

For more information on how these fields are used in a COBOL E15 exit, see "Linkage Section Fields" on page 176.

Figure 18 on page 166 details the interface to COBOL for the E15 exit.



Number of Bytes
* – VLR: Number of bytes is given by the corresponding length field
FLR: Number of bytes is equal to the LRECL

Figure 18. E15 DFSORT Interface with COBOL

Linkage Examples: Figure 19 is an example of the LINKAGE SECTION code for a fixed-length record data set with LRECL of 100, showing the layout of the fields passed to your COBOL routine.

Note: You only need to code up to the last field that your routine actually uses (for example, up to RETURN-REC if you do not use the exit area).

```
LINKAGE SECTION.
01  RECORD-FLAGS          PIC 9(8) COMPUTATIONAL.
    88  FIRST-REC               VALUE 00.
    88  MIDDLE-REC              VALUE 04.
    88  END-REC                 VALUE 08.
01  NEW-REC               PIC X(100).
01  RETURN-REC            PIC X(100).
01  UNUSED1               PIC 9(8) COMPUTATIONAL.
01  UNUSED2               PIC 9(8) COMPUTATIONAL.
01  UNUSED3               PIC 9(8) COMPUTATIONAL.
01  UNUSED4               PIC 9(8) COMPUTATIONAL.
01  UNUSED5               PIC 9(8) COMPUTATIONAL.
01  EXITAREA-LEN          PIC 9(4) COMPUTATIONAL.
01  EXITAREA.
    05  EAREA   OCCURS 1 TO 256 TIMES
            DEPENDING ON EXITAREA-LEN   PIC X.
```

Figure 19. LINKAGE SECTION Code Example for E15 (FLR)

Figure 20 on page 170 is an example of the LINKAGE SECTION code for a variable-length record data set with maximum LRECL of 200, showing the layout of the fields passed to your COBOL routine.

Notes:

1. *If the data used for input was not created by a COBOL run, you need to know the record length (LRECL) that is defined for your data set. For VLR records, the maximum length of the record defined in your COBOL exit is 4 bytes less than the LRECL value, because COBOL does not include the RDW as part of the record. (VLR records have an RDW field of 4 bytes at the beginning of each record that is not included in the record that is passed to your COBOL exit).*

2. *You only need to code up to the last field that your routine actually uses (for example, up to RETURN-REC-LEN if you do not use the exit area).*

```
LINKAGE SECTION.
01  RECORD-FLAGS        PIC 9(8) COMPUTATIONAL.
    88  FIRST-REC             VALUE 00.
    88  MIDDLE-REC            VALUE 04.
    88  END-REC               VALUE 08.
01  NEW-REC.
    05 NREC OCCURS 1 TO 200 TIMES
           DEPENDING ON NEW-REC-LEN      PIC X.
01  RETURN-REC.
    05 RREC OCCURS 1 TO 200 TIMES
           DEPENDING ON RETURN-REC-LEN   PIC X.
01  UNUSED1             PIC 9(8) COMPUTATIONAL.
01  UNUSED2             PIC 9(8) COMPUTATIONAL.
01  NEW-REC-LEN         PIC 9(8) COMPUTATIONAL.
01  RETURN-REC-LEN      PIC 9(8) COMPUTATIONAL.
01  UNUSED3             PIC 9(8) COMPUTATIONAL.
01  EXITAREA-LEN        PIC 9(4) COMPUTATIONAL.
01  EXITAREA.
    05  EAREA   OCCURS 1 TO 256 TIMES
           DEPENDING ON EXITAREA-LEN   PIC X.
```

Figure 20. LINKAGE SECTION Code Example for E15 (VLR)

Linkage Section Fields for FLR and VLR

The fields in the LINKAGE SECTION are used by DFSORT and your routine as stated below. For clarity, the field names from the above code examples have been used.

- To give your COBOL routine the status of the passed records, DFSORT uses the **Record Flags field** (RECORD-FLAGS) in the following way:

  0 (FIRST-REC)
  The new record is the **first** passed record

  4 (MIDDLE-REC)
  The new record is **not the first** passed record

  8 (END-REC)
  There is **no** new record to pass; all records have been passed to your routine or there were no records to pass

- DFSORT places the next input record in the **New Record field** (NEW-REC). A variable-length record does not contain a record descriptor word (RDW), but DFSORT places the length of this variable-length record in the **New Record Length field** (NEW-REC-LEN). The value in the NEW-REC-LEN field is the length of the record only and does not include the 4 bytes for the RDW.

- When your routine places an insertion/replacement record in the **return record field** (RETURN-REC), the variable-length record must not contain an RDW; your routine must place the length of this record in the **return record length**

field (RETURN-REC-LEN). The value of the RETURN-REC-LEN field is the length of the record only and should not include the 4 bytes for the RDW.

- DFSORT passes your routine a 256-byte **exit area field** (EXITAREA) for you to include information to be passed to your COBOL E15 exit each time it is called by DFSORT and/or to your COBOL E35 exit. The first time the exit area field is passed to your COBOL E15 exit, it contains 256 blanks, and the **exit area length field** (EXITAREA-LEN) contains 256.

  Any changes you make to the exit area field or exit area length fields is passed back to your COBOL E15 exit as well as to your COBOL E35 exit.

  Notes:

  1. *Do not set the exit area length field to more than 256 bytes.*

  2. *You only need to code up to the last field that your routine actually uses (for example, up to RETURN-REC if you do not use the exit area).*

Return Codes: Your COBOL E15 routine must pass one of the following return codes to DFSORT in the RETURN-CODE field (a COBOL special register) informing it what to do with the record you have been examining or changing:

0 No Action
4 Delete Record
8 Do not Return
12 Insert Record
16 Terminate DFSORT
20 Alter/Replace Record

0—No Action
If you want DFSORT to retain the record unchanged, return with RETURN-CODE set to 0.

4—Delete Record
If you want DFSORT to delete the record, return with RETURN-CODE set to 4.

8—Do Not Return
DFSORT continues to enter your routine until you return with RETURN-CODE set to 8. After that, the exit is not reentered during the DFSORT application. *Unless you are inserting records after end-of-data set, you must set RETURN-CODE to 8 when DFSORT indicates the end of the data set*, which it does by entering your routine with the Record Flags field set to 8.

If your exit passes a return code of 8 to DFSORT and there are still input records to be processed, the records are processed *without* being passed to your exit.

12—Insert Record
If you want DFSORT to add a record before the new record in the input data set, do the following:

- Move the insert record to the return record field.

- For VLR records, move the length to the return record length field. (Do not include the 4-byte RDW in this length.)
- Return with RETURN-CODE set to 12.

DFSORT reenters your routine with the same record as before in the new record field, allowing your routine to insert more records or handle the new record.

You can also insert records after end-of-data set. *DFSORT keeps returning to your routine as long as you pass it a RETURN-CODE 12 and until you return with a RETURN-CODE set to 8.*

16—Terminate DFSORT

If you want to terminate DFSORT, return with RETURN-CODE set to 16. DFSORT then returns to its calling program or to the system with a return code of 16.

20—Alter Record

If you want to change the new record, do the following:

- Move the new record to the return record field.
- Change the record in the return record field.
- For VLR records, move the length to the return record length field.
- Return with RETURN-CODE set to 20.

Note: If your routine changes record size, you must indicate the new size on the RECORD statement.

20—Replace Record

If you want to replace the new record, do the following:

- Move the replacement record to the return record field.
- For VLR records, move the length to the return record length field. (Do not include the 4-byte RDW in this length.)
- Return with RETURN-CODE set to 20.

Procedure Division Requirements

When coding the PROCEDURE DIVISION, the following requirements must be met:

- To return control to DFSORT, you must use the GOBACK statement.
- In the USING option of the PROCEDURE DIVISION header, you must specify **each** 01-level name in the LINKAGE SECTION. You must specify each name in order up to the last one you plan to use, even when you do not use all the 01-level names preceding the header.

Examples:

For the FLR example, Figure 19 on page 167, you would code:

```
PROCEDURE DIVISION USING RECORD-FLAGS, NEW-REC,
         RETURN-REC, UNUSED1, UNUSED2, UNUSED3,
         UNUSED4, UNUSED5, EXITAREA-LEN, EXITAREA.
```

For the VLR example, Figure 20 on page 168, you would code:

```
PROCEDURE DIVISION USING RECORD-FLAGS, NEW-REC,
         RETURN-REC, UNUSED1, UNUSED2,
         NEW-REC-LEN, RETURN-REC-LEN,
         UNUSED3, EXITAREA-LEN, EXITAREA.
```

Output Phase Exit

COBOL E35 Exit, Changing Records

The E35 routine is entered each time DFSORT prepares to place a record in the output area.

See Figure 15 on page 138 for logic flow details.

Some uses are:

- Add, delete, or change records in the output data set.
- Terminate DFSORT.

When DFSORT indicates the end of the data set (record flags field set to 8), you must set RETURN-CODE to 8 (unless you are inserting records after the end of the data set); otherwise, DFSORT continues to enter E35.

Notes:

1. *If both E15 and E35 exits are used, they must be in the same version of COBOL.*
2. *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.*
3. *If you omit the SORTOUT DD statement, your E35 exit routine must dispose of each output record and return to DFSORT with a return code of 4. When DFSORT returns to your routine after you have disposed of the last record, return to DFSORT with a return code of 8.*
4. *You cannot use dynamic link-editing together with a COBOL E35 exit.*

Interface with COBOL

Each time your E35 exit is called, DFSORT supplies the following fields:

- Record flags
- Record leaving DFSORT
- Length of record leaving DFSORT (for VLR)
- Length of exit area
- Exit area

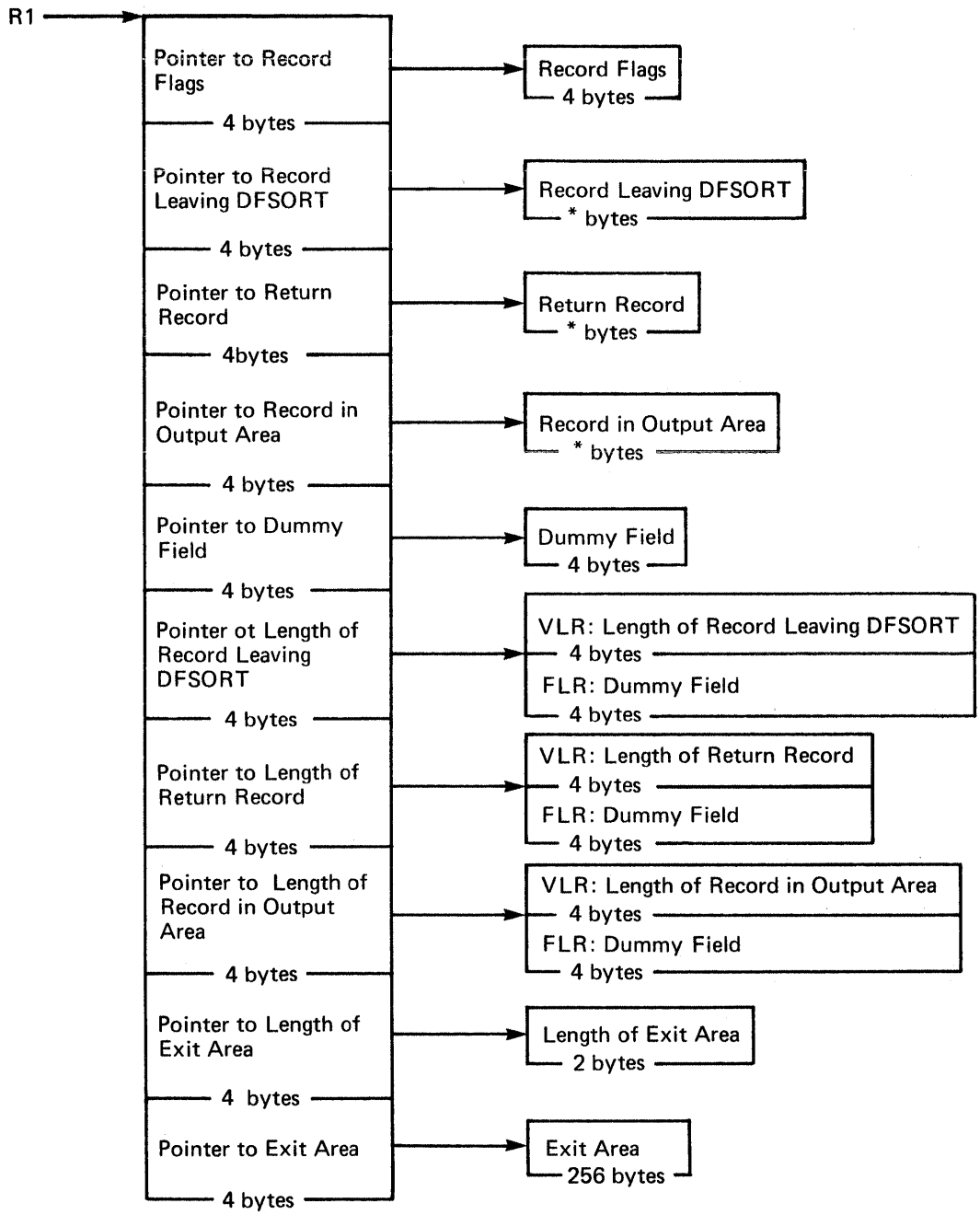
When your E35 exit returns to DFSORT, the E35 exit provides to DFSORT some or all the fields mentioned below. The first field is required, the other four may be modified as appropriate.

- RETURN-CODE (assigned by the exit by setting the COBOL special register RETURN-CODE)

- Return record
- Length of return record (for VLR)
- Length of exit area
- Exit area

For more information on how these fields are used in a COBOL E35 exit, see "Linkage Section Fields" on page 176.

Figure 21 on page 174 details the interface to COBOL for the E35 exit.



Number of Bytes
* – VLR: Number of bytes is given by the corresponding length field
FLR: Number of bytes is equal to the LRECL

Figure 21. E35 Interface with COBOL

Linkage Section Examples

Figure 22 is an example of the LINKAGE SECTION code for a fixed-length record data set with LRECL of 100, showing the layout of the fields passed to your COBOL routine.

Note: You only need to code up to the last field your routine actually uses (for example, up to OUTPUT-REC if you do not use the exit area).

```
LINKAGE SECTION.
01  RECORD-FLAGS          PIC 9(8) COMPUTATIONAL.
    88  FIRST-REC               VALUE 00.
    88  MIDDLE-REC              VALUE 04.
    88  END-REC                 VALUE 08.
01  LEAVING-REC           PIC X(100).
01  RETURN-REC            PIC X(100).
01  OUTPUT-REC            PIC X(100).
01  UNUSED1               PIC 9(8) COMPUTATIONAL.
01  UNUSED2               PIC 9(8) COMPUTATIONAL.
01  UNUSED3               PIC 9(8) COMPUTATIONAL.
01  UNUSED4               PIC 9(8) COMPUTATIONAL.
01  EXITAREA-LEN          PIC 9(4) COMPUTATIONAL.
01  EXITAREA.
    05  EAREA   OCCURS 1 TO 256 TIMES
            DEPENDING ON EXITAREA-LEN   PIC X.
```

Figure 22. LINKAGE SECTION Code Example for E35 (FLR)

Figure 23 on page 176 is an example of the LINKAGE SECTION code for a variable-length record data set with maximum LRECL of 200, showing the layout of the fields passed to your COBOL routine.

Notes:

1. *VLR records have a 4 byte RDW field at the beginning of each record. The maximum record length plus the RDW will be the length defined for the LRECL attribute of your output data set. COBOL programs do not use the RDW and therefore, the maximum length defined in your COBOL exit is 4 bytes less than the LRECL value.*

2. *You only need to code up to the last field your routine actually uses (for example, up to OUTPUT-REC-LEN if you do not use the exit area).*

```
LINKAGE SECTION.
01  RECORD-FLAGS        PIC 9(8) COMPUTATIONAL.
    88  FIRST-REC             VALUE 00.
    88  MIDDLE-REC            VALUE 04.
    88  END-REC               VALUE 08.
01  LEAVING-REC.
    05 LREC OCCURS 1 TO 200 TIMES
           DEPENDING ON LEAVING-REC-LEN      PIC X.
01  RETURN-REC.
    05 RREC OCCURS 1 TO 200 TIMES
           DEPENDING ON RETURN-REC-LEN   PIC X.
01  OUTPUT-REC.
    05 OREC OCCURS 1 TO 200 TIMES
           DEPENDING ON OUTPUT-REC-LEN   PIC X.
01  UNUSED1             PIC 9(8) COMPUTATIONAL.
01  LEAVING-REC-LEN     PIC 9(8) COMPUTATIONAL.
01  RETURN-REC-LEN      PIC 9(8) COMPUTATIONAL.
01  OUTPUT-REC-LEN      PIC 9(8) COMPUTATIONAL.
01  EXITAREA-LEN        PIC 9(4) COMPUTATIONAL.
01  EXITAREA.
    05  EAREA   OCCURS 1 TO 256 TIMES
           DEPENDING ON EXITAREA-LEN   PIC X.
```

Figure 23. LINKAGE SECTION Code Example for E35 (VLR)

Linkage Section Fields: The fields in the LINKAGE SECTION are used by DFSORT and your routine as stated below. For clarity, the field names from the above code examples have been used.

- To give your COBOL routine the status of the passed records, DFSORT uses the **record flags field** (RECORD-FLAGS) in the following way:

  0 (FIRST-REC)
  The record leaving DFSORT is the **first** passed record

  4 (MIDDLE-REC)
  The record leaving DFSORT is **not the first** passed record

  8 (END-REC)
  There is **no** record leaving DFSORT to pass; all records have been passed to your routine or there were no records to pass

- DFSORT places the next output record (which usually follows the record in the output area) in the **record leaving field** (LEAVING-REC). A variable-length record does not contain an RDW; DFSORT places the length of this variable length record in the **record leaving length field** (LEAVING-REC-LEN). The value in the LEAVING-REC-LEN field is the length of the record only, and does not include the 4 bytes for the RDW.

- When your routine places an insertion/replacement record in the **return record field** (RETURN-REC), the variable-length record must not contain an RDW; your routine must place the length of this record in the **return record length field** (RETURN-REC-LEN). The value in the RETURN-REC-LEN field is the length of the record only, and does not include the 4 bytes for the RDW.

- DFSORT places the record already in the output area in the **record in output area field** (OUTPUT-REC). A variable-length record does not contain an RDW. DFSORT places the length, not including the 4 bytes for RDW, of this variable-length record in the **record in output area length field** (OUTPUT-REC-LEN).

- DFSORT passes your routine a 256-byte **exit area field** (EXITAREA) that may contain information passed by your COBOL E15 routine. If no information is passed in this area by your COBOL E15 routine the first time the exit area field is passed to your COBOL E35 routine, it contains 256 blanks, and the **exit area length field** (EXITAREA-LEN) contains 256.

  Any changes you make to the exit area field or exit area length field is passed back to your COBOL E35 routine each time it is called by DFSORT.

  Note: Do not set the exit area length field to more than 256 bytes.

Return Codes: Your COBOL E35 routine must pass one of the following return codes to DFSORT in the RETURN-CODE field (a COBOL reserved keyword) instructing it what to do with the record you have been examining or changing:

0 No Action
4 Delete Record
8 Do Not Return
12 Insert Record
16 Terminate DFSORT
20 Alter/Replace Record

0—No Action
If you want DFSORT to retain the record leaving DFSORT unchanged, return with RETURN-CODE set to 0.

4—Delete Record
If you want DFSORT to delete the record leaving DFSORT, return with RETURN-CODE set to 4.

8—Do Not Return
DFSORT keeps returning to your routine until you pass a RETURN-CODE set to 8. After that, the exit is not reentered during the DFSORT application. *Unless you are inserting records after the end-of-data set, you must set RETURN-CODE to 8 when DFSORT indicates the end of the data set*, which it does by entering your routine with the record flags field set to 8.

If your exit passes a return code of 8 to DFSORT and there are still input records to be processed, the records are processed *without* being passed to your exit.

If you do not have a SORTOUT data set and would usually return with return code 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 DFSORT to add a record to the SORTOUT data set before the record leaving DFSORT, do the following:

- Move the insert record to the return record field.
- For VLR records, move the length to the return record length field.
- Return with RETURN-CODE set to 12.

DFSORT reenters your routine with the inserted record in the record output area field, and with the same record as before in the record leaving DFSORT field. In this way, your routine can insert more records or handle the record leaving DFSORT.

You can also insert records after end-of-data set. *DFSORT keeps returning to your routine as long as you pass it a RETURN-CODE 12 and until you return with RETURN-CODE set to 8.*

DFSORT does not perform sequence checking for disk sorts. For tape sorts, DFSORT does not perform sequence checking on records that you insert unless you delete the record leaving DFSORT and insert a record to replace it.

16—Terminate DFSORT

If you want to terminate DFSORT, return with RETURN-CODE set to 16. DFSORT then returns to its calling program or to the system with a return code of 16.

20—Alter Record

If you want to change the record leaving DFSORT, do the following:

- Move the record leaving DFSORT to the return record field.
- Change the record in the return record field.
- For VLR records, move the length to the return record length field.
- Return with RETURN-CODE set to 20.

Note: If your routine changes record size, you must indicate the new size on the RECORD statement.

20—Replace Record

If you want to replace the record leaving DFSORT, do the following:

- Move the replacement record to the return record field.
- For VLR records, move the length to the return record length field.
- Return with RETURN-CODE set to 20.

Procedure Division Requirements

When coding the PROCEDURE DIVISION, the following requirements must be met:

- To return control to DFSORT, you must use the GOBACK statement.
- In the USING option of the PROCEDURE DIVISION header, you must specify **each** 01-level name in the LINKAGE SECTION. You must specify each name in order up to the last one you plan to use, even when you do not use all the 01-level names preceding the header.

  Examples:

  For the FLR example, Figure 22 on page 175, you would code:

```
PROCEDURE DIVISION USING RECORD-FLAGS, LEAVING-REC,
         RETURN-REC, OUTPUT-REC, UNUSED1, UNUSED2,
         UNUSED3, UNUSED4, EXITAREA-LEN, EXITAREA.
```

  For the VLR example, Figure 23 on page 176, you would code:

```
PROCEDURE DIVISION USING RECORD-FLAGS, LEAVING-REC,
         RETURN-REC, OUTPUT-REC, UNUSED1,
         LEAVING-REC-LEN, RETURN-REC-LEN,
         OUTPUT-REC-LEN, EXITAREA-LEN, EXITAREA.
```

Sample Routines Written in COBOL

COBOL E15:

Figure 24 is an example of a COBOL E15 routine fo a data set with fixed-length records of 100 bytes. It examines the department field in the passed record and takes the following action:

- If the department is D29, it changes it to J99.
- If the department is not D29, it accepts the record unchanged.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
    CE15.
ENVIRONMENT DIVISION.
DATA DIVISION.
LINKAGE SECTION.
01  RECORD-FLAGS        PIC 9(8) COMPUTATIONAL.
    88  FIRST-REC             VALUE 00.
    88  MIDDLE-REC            VALUE 04.
    88  END-REC               VALUE 08.
01  NEW-REC.
 05 NFILL1               PIC X(10).
 05 NEW-DEPT             PIC X(3).
 05 NFILL2               PIC X(87).
01  RETURN-REC.
 05 RFILL1               PIC X(10).
 05 RETURN-DEPT          PIC X(3).
 05 RFILL2               PIC X(87).

PROCEDURE DIVISION USING RECORD-FLAGS, NEW-REC, RETURN-REC.

    IF END-REC
       MOVE 8 TO RETURN-CODE
       GO TO BACK-TO-SORT.

    IF NEW-DEPT EQUAL TO "D29"
       MOVE NEW-REC TO RETURN-REC
       MOVE "J99" TO RETURN-DEPT
       MOVE 20 TO RETURN-CODE

    ELSE
       MOVE 0 TO RETURN-CODE.

BACK-TO-SORT.
    GOBACK.
```

Figure 24. COBOL E15 Routine Example (FLR)

COBOL E35: Inserting Records

Figure 25 on page 182 is an example of a COBOL E35 routine for a data set with variable-length records up to 200 bytes. It examines the department field in each passed record (records are assumed to be sorted by the department field) and takes the following action:

- It inserts a record for Department K22 in the proper sequence.
- It accepts all passed records unchanged.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
    CE35.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  INSERT-DONE PIC 9(1) VALUE 0.
01  K22-REC.
 05 K22-MANAGER PIC X(20) VALUE "J. DOE".
 05 K22-DEPT    PIC X(3)  VALUE "K22".
 05 K22-FUNC    PIC X(20) VALUE "ACCOUNTING".
 05 K22-LATER   PIC X(30) VALUE SPACES.
01  LEAVING-VAR-LEN PIC 9(8) COMPUTATIONAL.
LINKAGE SECTION.
01  RECORD-FLAGS       PIC 9(8) COMPUTATIONAL.
    88  FIRST-REC            VALUE 00.
    88  MIDDLE-REC           VALUE 04.
    88  END-REC              VALUE 08.
01  LEAVING-REC.
 05 LREC-MANAGER PIC X(20).
 05 LREC-DEPT    PIC X(3).
 05 LREC-FUNC    PIC X(20).
 05 LREC-LATER   OCCURS 1 TO 157 TIMES
           DEPENDING ON LEAVING-VAR-LEN PIC X.
01  RETURN-REC.
 05 RREC        OCCURS 1 TO 200 TIMES
           DEPENDING ON RETURN-REC-LEN   PIC X.
01  OUTPUT-REC.
 05 OREC        OCCURS 1 TO 200 TIMES
           DEPENDING ON OUTPUT-REC-LEN   PIC X.
01  UNUSED1            PIC 9(8) COMPUTATIONAL.
01  LEAVING-REC-LEN    PIC 9(8) COMPUTATIONAL.
01  RETURN-REC-LEN     PIC 9(8) COMPUTATIONAL.
01  OUTPUT-REC-LEN     PIC 9(8) COMPUTATIONAL.

PROCEDURE DIVISION USING RECORD-FLAGS, LEAVING-REC,
         RETURN-REC, OUTPUT-REC, UNUSED1,
         LEAVING-REC-LEN, RETURN-REC-LEN,
         OUTPUT-REC-LEN.

    IF END-REC
       MOVE 8 TO RETURN-CODE
       GO TO BACK-TO-SORT.
    IF INSERT-DONE EQUAL TO 1
       MOVE 0 TO RETURN-CODE
       GO TO BACK-TO-SORT.
    SUBTRACT 43 FROM LEAVING-REC-LEN
       GIVING LEAVING-VAR-LEN.
    IF LREC-DEPT GREATER THAN K22-DEPT
       MOVE 1 TO INSERT-DONE
       MOVE 43 TO RETURN-REC-LEN
       MOVE K22-REC TO RETURN-REC
       MOVE 12 TO RETURN-CODE
    ELSE
       MOVE 0 TO RETURN-CODE.
BACK-TO-SORT.
    GOBACK.
```

Figure 25. COBOL E35 Routine Example (VLR)

Assembler and COBOL User Exit Routines and DFSORT Performance

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

- Your routines occupy main storage that would otherwise be available to DFSORT. Because its main storage is restricted, DFSORT may need to execute extra passes to sort the data. This, of course, increases sorting time.
- The execution of user exit routines adds time to the overall execution time. Note that several of the exits give your routine control once for each record until you pass a "do not return" return code to DFSORT. You should remember this when designing your routines.
- Use INCLUDE, OMIT, INREC, OUTREC, and SUM instead of exit routines whenever possible.

Summary of Rules for User Exit Routines

When preparing your routines, remember the following:

- User-written routines must follow standard linkage conventions, and use the described interfaces. COBOL E15 and E35 routines must use the special interface provided.
- To use an E32 exit, your invoking program must pass its address to DFSORT in the parameter list.
- To use any other exit, you must associate your routine with the appropriate exits using the MODS control statement. See "MODS Control Statement" on page 67.
- Your invoking program may alternatively pass the address of an E15, E18, E35, and/or E39 exit to DFSORT in the parameter list.
- When a disk technique is used and your exits are reenterable, the entire DFSORT program is reenterable.
- If you are using ISCII/ASCII input, remember that data presented to your exits at user exits are in EBCDIC format (all data is represented internally in EBCDIC). If the E61 exit is used to resolve ISCII/ASCII collating for special alphabetic characters, substituted characters must be in EBCDIC, but the sequencing result depends on the byte value of the ISCII/ASCII translation for the substituted character.

How to Load User Exit Routines

You must assemble or compile each user exit as a separate program. If your user exit operates independently, link-edit it separately into a partitioned data set (library) with the member name to be used in the MODS statement. If your user exit operates in conjunction with other user exits in the same phase (for example, E11, E15, and E17 all use the same DCB), you can request DFSORT to dynamically link-edit them together (see MODS statement). Alternatively, you can link-edit them together into a partitioned data set following these rules:

1. Specify RENT as a linkage editor parameter.
2. Include an ALIAS statement for each exit routine using the external entry name of the routine (for example, the CSECT name).
3. Specify the appropriate ALIAS name for each exit routine on the MODS statement.

DFSORT includes the names and locations of your user exits in the list of modules to be executed during each phase. No user exit is loaded more than once in a program phase, but the same exit can appear in different phases. For example, you can use the same Read Error user exit in both phases, but not twice in one phase.

The individual lengths of the exits specified on a MODS statement are not important, but the sum of the lengths must be the total length of the modules. For example, all but one length may be specified as zero, and the total length specified for the remaining exits. The length should also include any storage used by your exits outside of the load modules, such as I/O buffers or COBOL library subroutines. The parameters on the MODS statement that defines the exit must be the same as the name of the DD statement that defines the library. For example:

```
//MYLIB DD   DSNAME=MYRTN, etc.
          .
          .
          .
   MODS    E15=(MODNAME,500,MYLIB,N)
```

User Exit Linkage Conventions

The program uses a CALL macro instruction expansion to enter a user exit. Therefore, each user exit must contain an entry point whose name is that of the associated program exit.

The general registers used by DFSORT for linkage and communication of parameters follow operating system conventions (see Figure 26 on page 185).

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

Figure 26. Register Conventions

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

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

How to Dynamically Link-Edit User Exit Routines

You can dynamically link-edit any user exit routine written in any language that has the ability to pass the location/address of a record or parameter in register 1 and a return code in register 15 (see MODS statement). This does not include E15 and E35 routines written in COBOL.

On MVS/XA systems, dynamic link-editing does not support AMODE 31 or RMODE 31 for the link-edit option T. The exits that are link-edited ***together*** by DFSORT are not loaded above 16-megabyte virtual and can not be entered in 31-bit addressing mode. Exits link-edited with the S option retain the AMODE and RMODE attributes of the object modules, and are loaded above or below 16-megabyte virtual depending upon the load's module's RMODE; they are entered in the addressing mode of the exit.

Notes:

1. *The Blockset technique is not used for dynamic link-editing.*
2. *Dynamic link-editing cannot be used with copy.*

Linkage Examples

The CALL macro instruction used by DFSORT to link to your exits is written as follows:

```
CALL   E15
```

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. DFSORT has already placed the register save area address in general register 13.

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

```
      ENTRY  E15
      .
      .
E15   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  E15
      .
      .
E15   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 DFSORT of the action it is to take. The following macro instruction could be used to return to the DFSORT 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 is in *Supervisor Services and Macro Instructions*.

Chapter 5. Invoking DFSORT from an Assembler Program

This chapter describes how you can initiate DFSORT from within your assembler program with a system macro instruction, instead of with the EXEC job control statement in the input stream.

DFSORT 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.

Merge restriction

Merge applications cannot be done when DFSORT is invoked from a PL/I program.

Copy restrictions

- Copy applications cannot be done when DFSORT is invoked from a PL/I program.
- If you invoke DFSORT from a COBOL program, the following restrictions apply:
  - If using OS/VS COBOL, a copy application cannot be done.
  - If using VS COBOL II, the OPTION COPY statement can be placed in either the COBOL II IGZSRTCD data set or the DFSORT SORTCNTL data set.
  - If using the COBOL II FASTSRT compile time option for any part or all of the COBOL SORT statement, a copy application can be done.
  - If using the COBOL MERGE statement, a copy application cannot be done.

See "Requirements for Copy Processing" on page 163 for exit requirements.

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 installed.

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 DFSORT is brought into main storage. The linkage relationship between your program and DFSORT differs according to which of the instructions you have used. For a complete description of the macro instructions and how to use them, refer to *Supervisor Services and Macro Instructions*.

How to Use the Macros

In order to initiate execution of DFSORT with a system macro instruction, you must:

- Write the required job control language DD statements.
- Write DFSORT control statements as operands of assembler DC instructions.
- Write a parameter list containing information to be passed to DFSORT and a pointer containing the address of the parameter list. Two types of parameter lists are accepted by DFSORT: a 24-bit parameter list, and an extended parameter list. Although you can choose either parameter list for OS/VS1, MVS, or MVS/XA applications, the extended parameter list can perform a superset of the functions in the 24-bit parameter list, and thus should be used for new DFSORT applications.
- Prepare the macro instruction, in which you must specify the entry point name of DFSORT.

Note: The save area passed to DFSORT must begin on a fullword boundary.

In addition, the following rule applies:

- If you are invoking DFSORT 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 type shown in Figure 27 are usually required. The statements and their necessary parameters are described in Chapter 3.

```
//SORTLIB  DD  (parameters)
        Defines the data set containing the special DFSORT
        program modules for a sort using tape work files or a merge
        using the conventional technique.

//SORTIN[1]  DD  (parameters)
        Defines the data set to be sorted or copied.
        Not needed if you supply all input through E15.

//SORTINnn[1] DD  (parameters)
        Defines data sets to be merged. Not needed if you supply
        an E32 exit.

//SORTWKnn[1] DD  (parameters)
        Defines work data sets. Needed for most sorting applications
        applications but not for a merge or copy.

//SYSOUT[1]  DD  SYSOUT=A
        Defines the output data set for DFSORT messages.

//SORTOUT[1]  DD  (parameters)
        Defines the output data set. Not needed if you handle
        all output through E35.

//SORTCNTL[1] DD  *
        Defines a data set with overriding control statements. Not
        needed if the control statements in the parameter list are
        acceptable.

//SORTDIAG  DD  DUMMY
        Only used for debugging; usually not needed.

//SYSIN DD
        Contains user exit routines to be link-edited by DFSORT
        in object deck format.

//SORTMODS DD
        Defines a temporary partitioned data set large enough to
        contain all your exit routines that appear in SYSIN for a
        given application.
```

Figure 27 (Part 1 of 2). Example of DD Statements for a Dynamically Invoked Sort

//SYSPRINT DD
Used for messages from the linkage editor.

//SYSUT1 DD
Used as a work area by the linkage editor.

//SYSLIN DD
Defines a data set in which DFSORT places control information for the linkage editor.

//SYSLMOD DD
Defines a data set that contains output from the linkage editor.

Figure 27 (Part 2 of 2). Example of DD Statements for a Dynamically Invoked Sort

[1] These are the default ddnames. They can be changed at execution time by the parameter list. SYSOUT can also be changed at installation time. For override information, see Appendix D.

Program Control Statements for the 24-Bit Parameter List

The program control statements described in Chapter 2 are usually provided in the form of character constants defined by assembler DC instructions. When using the 24-bit parameter list, the address of each control statement must be given in the parameter list. The rules for preparing the program control statements are:

- Program control statements must be in EBCDIC format.
- SORT (or MERGE) and RECORD statements are always required.
- The MODS statement is required when exits other than E15, E32, and E35 are to be used, or when the E15 or E35 routine addresses are not passed by the parameter list.
- ALTSEQ can be used to modify the EBCDIC collating sequence, as described in "ALTSEQ Control Statement" on page 42.
- DEBUG is needed only for debugging.
- At least one blank must follow the operation definer (SORT, MERGE, RECORD, ALTSEQ, DEBUG, or MODS). A control statement may start with one or more blanks and must end with at least one blank. No other blanks are allowed.

- The content and format of the statements are as described in Chapter 2, except:
  - Labels are not allowed; a leading blank is optional.
  - No continuation character is allowed (the statements are not specified in image format).
- Neither comment statements nor comment fields 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.

  For full override and applicability details, see Appendix D.

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.

Program Control Statements for the Extended Parameter List

When using the extended parameter list, the control statements are written in a single area to which the parameter list points. The control statement area consists of:

- A 2-byte field containing the length (in binary) of the character string to follow.
- A character string containing valid images of the control statements to be used at execution time.

The control statements must be separated by one or more blanks; a blank preceding the first statement is optional; however, a trailing blank is required. No labels, comment statements, or comment fields are allowed.

Format of the 24-Bit Parameter List

Figure 28 on page 192 shows the format of the 24-bit parameter list and the pointer containing its address which you must pass to DFSORT. Detailed specifications for each of the entries in the parameter list follow.

For full override and applicability details, see Appendix D.

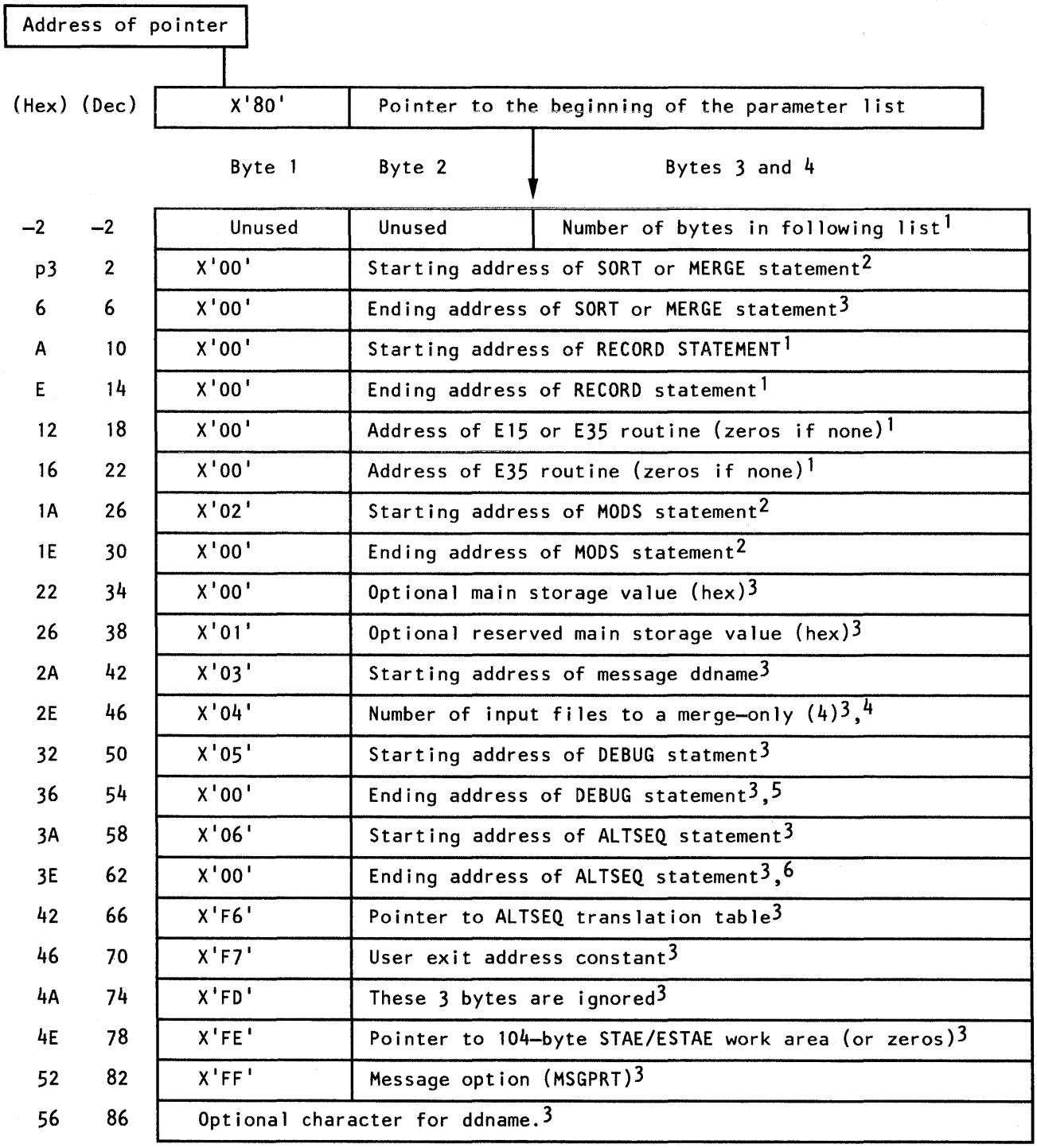


Figure 28. The 24-Bit Parameter List When Attaching the Program

Notes:

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, unless the FILES option of the MERGE statement is specified (see Appendix D).

5 The ending address of the DEBUG statement must appear after the starting address.

6 The ending address of the ALTSEQ statement must appear after the starting address.

Byte	Explanation
-2 to -1	Unused.
0 to +1	The byte count. This 2-byte field contains the length of the parameter list. The length is specified in bytes, in hexadecimal. This 2-byte field 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. The first byte must contain X'00'.
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. The first byte must contain X'00'. (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. You can use this option to temporarily override the SIZE={MAX | n} installation option. (For full override details, see Appendix D.) For an explanation of this value, see the MAINSIZE parameter of the OPTION statement.

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, where the minimum is 4K. For an explanation of this value, see the RESINV parameter of the OPTION statement.

You can use this option to temporarily override the RESINV=n installation default. For full override and applicability details, see Appendix D.

42-45 Message ddname (optional). The ddname for the output data set for program messages. You can use this option to override the installation default. For full override and applicability details, see Appendix D. For details on use of the message data set, see Appendix H.

The first byte must contain X'03'. The following three bytes contain the address of an 8-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 is needed only if the MERGE statement is present and input to the merge is being supplied through the E32 exit. This information may also be supplied on the MERGE statement. The first byte must contain X'04'. The next three bytes contain the number of files, in hexadecimal. For full override and applicability details, see Appendix D.

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. The first byte must contain X'00'.

58-61 The starting address of the ALTSEQ control statement image. The first byte must be X'06'. For full override and applicability details, see Appendix D.

62-65 The ending address of the ALTSEQ control statement image. Must be in the last three bytes. The first byte must contain X'00'.

66-69 The address of 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'. For full override and applicability details, see Appendix D.

70-73 User exit address constant.

These 4 bytes are passed to E15 (at offset 4 in the E15 parameter list) and/or to E35 (at offset 8 in the E35 parameter list) after DFSORT replaces the X'F7' with an X'00'.

74-77 X'FD' in the first byte (the VLSHRT option) specifies that DFSORT is to continue sorting or merging if a variable-length input record is encountered that is too short to contain all specified control fields. For full details of this option, see the VLSHRT parameter on the OPTION statement. You can use this option to temporarily override the NOVLSHRT installation default. For full override and applicability details, see Appendix D.

78-81 If the first byte contains X'FE', the STAE/ESTAE 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/ESTAE work area will be saved; otherwise, these bytes must contain zeros. If this option is omitted, no STAE/ESTAE routine will receive control at program failure.

82-85 The message option. The first byte must contain X'FF'. The following three bytes contain the characters NOF, (I), or (U). This parameter is equivalent to the MSGPRT option of the OPTION statement and specifies the printing of messages as follows:

NOF Messages and control statements are not printed. Critical messages are written to the master console.

(I) All messages except diagnostic messages (ICE800I to ICE999I) are printed. Critical messages are also written to the master console. Control statements are printed only if LIST is in effect.

(U) Only critical messages are printed. They are also written to the master console. Control statements are not be printed (NOLIST is forced).

All messages are written to the message data set. For details on use of the message data set, see Appendix H. For full override and applicability details, see Appendix D.

For compatibility reasons, the forms:
{(NO | (AP | (AC | (CC | (CP | (PC} are also accepted.

Following is the MSGPRT/MSGCON equivalence for these options:

Option	MSGPRT	MSGCON
(NO	NONE	NONE
(AP	ALL	CRITICAL
(AC	NONE	ALL
(CC	NONE	CRITICAL
(CP	CRITICAL	CRITICAL
(PC	ALL	ALL

86-89 Four characters, which replace "SORT" in the following ddnames: SORTIN, SORTOUT, SORTINnn, SORTWKnn, and SORTCNTL. You must use this option when you dynamically invoke DFSORT more than once in a program step.

The four characters must all be alphanumeric or national ($, #, or @), the first character must be alphabetic, and the reserved names DIAG, BALN, OSCL, POLY, CRCX, PEER, LIST, and SYSc (where c is any alphanumeric character) must not be used. Otherwise, the four characters are ignored.

Example: If you use ABC# as replacement characters, DFSORT will use DD statements ABC#IN, ABC#CNTL, ABC#WKnn, and ABC#OUT instead of SORTIN, SORTCNTL, SORTWKnn, and SORTOUT.

Format of the Extended Parameter List

Figure 29 on page 197 shows the format of the extended parameter list and the pointer containing its address, which you must pass to DFSORT.

The first parameter must be specified. A 4-byte field containing X'FFFFFFFF' *must* be used to indicate the end of the parameter list, and can be coded anywhere after the first parameter.

If a parameter is specified, it must appear in the indicated position and must contain a 31-bit address or a clean (the first 8 bits containing zeros) 24-bit address.

If a parameter is not specified, it will be treated as if it were specified as zeros.

For full override and applicability details, see Appendix D.

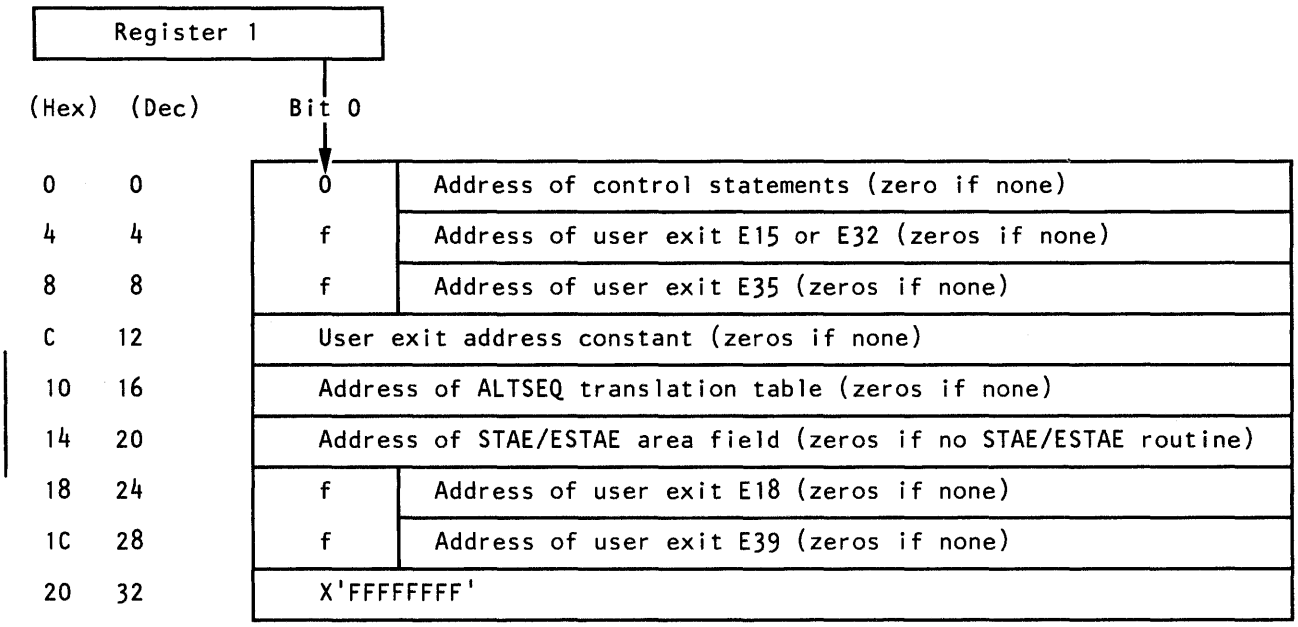


Figure 29. Extended Parameter List

Byte	Explanation
0-3	Required. The address of the area containing the DFSORT control statements, if any; otherwise all zeros. The high order bit must be 0 to identify this as an extended parameter list. If this is the last parameter, it must be followed by X'FFFFFFFF'. If you specify this parameter as zeros, you must supply all the required control statements by means of the SORTCNTL DD statement.
4-7	Optional. The address of the E15 or E32 routine that your program has placed in main storage (for example, via LOAD), if any; otherwise, all zeros. If this is the last parameter, it must be followed by X'FFFFFFFF'. f (bit 0) has the following meaning when executing in an MVS/XA system: • 0 = Enter the exit with 24-bit addressing in effect (AMODE 24). • 1 = Enter the exit with 31-bit addressing in effect (AMODE 31).
8-11	Optional. The address of the E35 routine that your program has placed in main storage (for example, via LOAD), if any; otherwise, all zeros. If this is the last parameter, it must be followed by X'FFFFFFFF'.

f (bit 0) has the following meaning when executing in an MVS/XA system:

- 0 = Enter the exit with 24-bit addressing in effect (AMODE 24).
- 1 = Enter the exit with 31-bit addressing in effect (AMODE 31).

12-15 Optional. The user address constant, if any; otherwise, all zeros. This field will be passed to the E15 and/or E35 routines. If this is the last parameter, it must be followed by X'FFFFFFFF'.

16-19 Optional. The address of a 256-byte translate table supplied instead of an ALTSEQ statement, if any; otherwise, all zeros. If specified, it will override any translate table specified at installation time. If this is the last parameter, it must be followed by X'FFFFFFFF'.

20-23 Optional. The address of a STAE/ESTAE area field if the STAE/ESTAE routine you provide should receive control at program failure; otherwise, all zeros. If this is the last parameter, it must be followed by X'FFFFFFFF'.

The STAE/ESTAE area field is a 4-byte field containing either the address of a 112-byte work area where STAE/ESTAE information will be saved, or all zeros if the STAE/ESTAE information is not to be saved.

24-27 Optional. The address of the E18 routine that your program has placed in main storage (for example, via LOAD), if any; otherwise, all zeros. **This parameter will be ignored for a merge application and for a tape work data set sort application.** If this is the last parameter, it must be followed by X'FFFFFFFF'.

f (bit 0) has the following meaning when executing in an MVS/XA system:

- 0 = Enter the exit with 24-bit addressing in effect (AMODE 24).
- 1 = Enter the exit with 31-bit addressing in effect (AMODE 31).

28-31 Optional. The address of the E39 routine that your program has placed in main storage (for example, via LOAD), if any; otherwise, all zeros. **This parameter is ignored for a conventional merge application and for a tape work data set sort application.** Because this is the last parameter, it must be followed by X'FFFFFFFF'.

f (bit 0) has the following meaning when executing in an MVS/XA system:

- 0 = Enter the exit with 24-bit addressing in effect (AMODE 24).
- 1 = Enter the exit with 31-bit addressing in effect (AMODE 31).

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 may also need the ECB parameter.

If you provide an E15 exit routine address in the parameter list, DFSORT ignores 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, DFSORT ignores the SORTOUT data set if you provide an E35 exit routine address in the parameter list. 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 DFSORT 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 so that they are unique. For ways of doing this, and for the rules of override, see Appendix D.

If you ATTACH more than one DFSORT application from the same program, you must wait for the first to complete before attaching the next, and so forth—unless the application is a disk sort, in which case the program is reenterable (provided that any exit routines you use are also reenterable).

When you initiate DFSORT 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 is overlaid by DFSORT.

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 DFSORT in general register 1.

Examples

Example 1. Specifying a Main Storage Value (24-Bit Parameter List)

Figure 30 shows a 24-bit parameter list when specifying the main storage option for a sort application.

(Hex)(Dec)

-2 -2	Unused	X'001C'
2 2	X'00'	Starting address of SORT statement
6 6	X'00'	Ending address of SORT statement
A 10	X'00'	Starting address of RECORD statement
E 14	X'00'	Ending address of RECORD statement
12 18	X'00'	Address of E15 routine
16 22	X'00'	Address of E35 routine
1A 26	X'00'	Main storage value (in hexadecimal)

Figure 30. Specifying the Main Storage Option (24-Bit Parameter List)

Example 2. Supplying Input through Exit E32 and Giving Control to the STAE/ESTAE Routine (24-Bit Parameter List)

Figure 31 shows a 24-bit parameter list for a merge application when supplying input through exit E32 and giving control to the STAE/ESTAE routine if the program fails.

(Hex)(Dec)

-2 -2	Unused	X'001C'
2 2	X'00'	Starting address of MERGE statement
6 6	X'00'	Ending address of MERGE statement
A 10	X'00'	Starting address of RECORD statement
E 14	X'00'	Ending address of RECORD statement
12 18	X'00'	Address of E32 routine
16 22	X'00'	Zeros (no E35 routine provided)
1A 26	X'04'	Number of input files
1E 30	X'FE'	(Zeros—no work area address provided)

Figure 31. Specifying E32 and STAE/ESTAE Routine (24-Bit Parameter List)

Example 3. How a 24-Bit Parameter List Might Appear in Main Storage

Figure 32 shows how a 24-bit parameter list might appear in main storage.

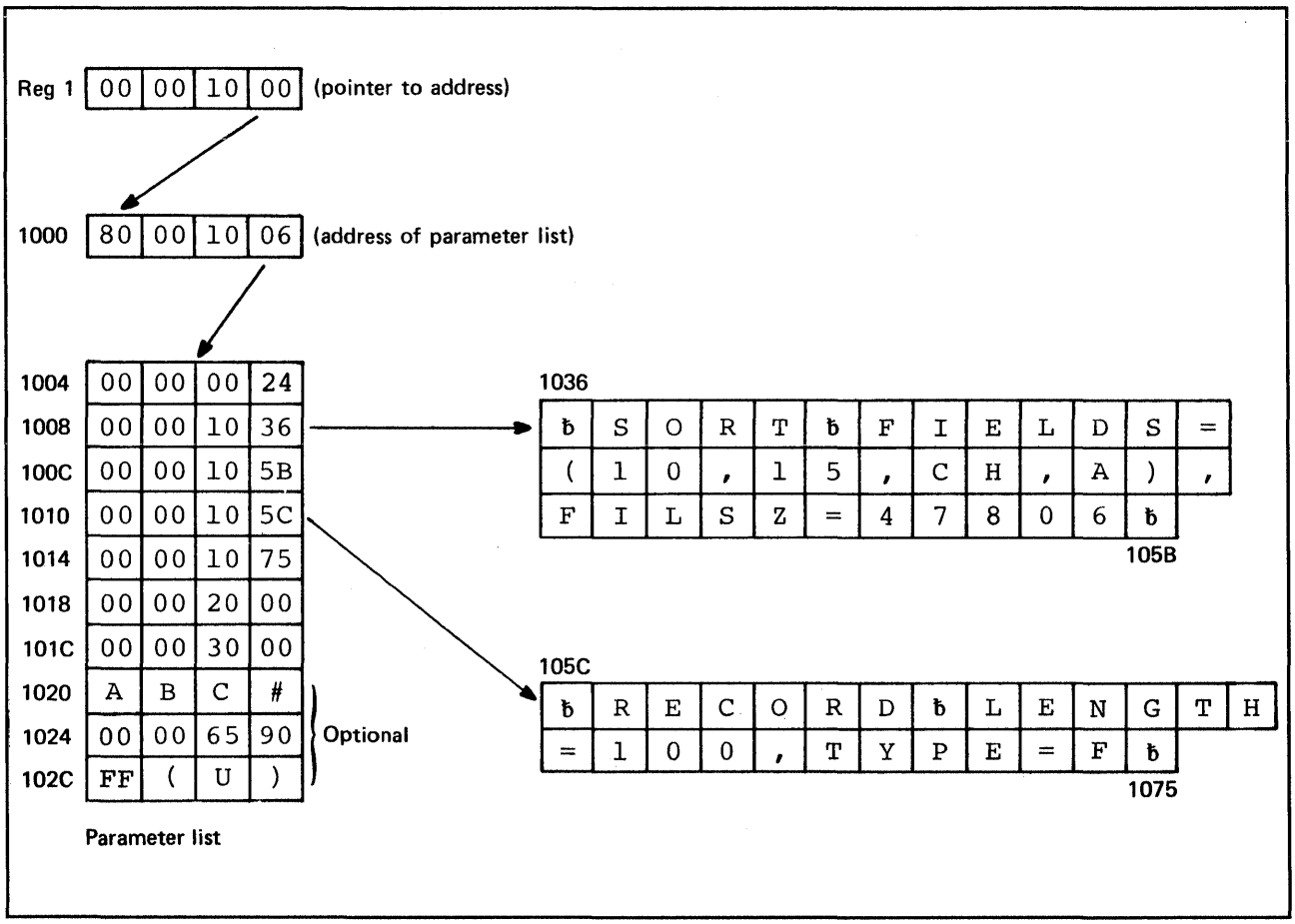


Figure 32. The 24-Bit Parameter List in Main Storage

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 2-byte field 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 4. Coding a 24-Bit Parameter List

The example in Figure 33 shows, in assembler language, how to code the parameters and statement images in Figure 32 on page 202, and how to pass control to DFSORT.

```
         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'720000'               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 33. Coding a 24-Bit Parameter List

Example 5. Coding an Extended Parameter List

The example in Figure 34 shows, in assembler language, how to code parameters and statement images, and pass control to DFSORT, using an extended parameter list.

```
         .
         .
         .
         LA    R1,PL1             SET ADDRESS OF PARAMETER LIST
*                                  TO BE PASSED TO SORT/MERGE
         ST    R2,PL4             SET ADDRESS OF GETMAINED AREA
*                                  TO BE PASSED TO E15
         LINK EP=SORT             INVOKE SORT/MERGE
         .
         .
         .
PL1      DC    A(CTLST)           ADDRESS OF CONTROL STATEMENTS
PL2      DC    A(E15)             ADDRESS OF E15 ROUTINE
PL3      DC    A(0)               NO E35 ROUTINE
PL4      DS    A                  USER EXIT ADDRESS CONSTANT
PL5      DC    F'-1'              INDICATE END OF LIST
CTLST    DS    0H                 CONTROL STATEMENTS AREA
         DC    AL2(CTL2-CTL1)     LENGTH OF CHARACTER STRING
CTL1     DC    C' SORT FIELDS=(4,5,CH,A)'
         DC    C' OPTION '
         DC    C'RESINV=2048,FILSZ=E25000,MSGDDN=MSGOUT '
         DC    C' OMIT COND=(5,8,EQ,13,8),FORMAT=FI '
         DC    C' RECORD TYPE=F,LENGTH=80 '
CTL2     EQU   *
OUT      DCB   DDNAME=SYSOUT,... MYSORT USES SYSOUT
E15      DS    0H                 E15 ROUTINE
         .
         .
         .
         BR    R14                RETURN TO SORT/MERGE
* MAPPING OF PARAMETER LIST PASSED TO E15 FROM SORT/MERGE
SRTLST DS    A                    ADDRESS OF RECORD
GMA      DS    A                  ADDRESS OF AREA GETMAINED BY
*                                  MYSORT
         .
         .
         .
```

Figure 34. Coding an Extended Parameter List

Chapter 6. Improving Program Efficiency

DFSORT automatically optimizes performance by analyzing the information given to it. This automatic optimization sets optimization variables (such as buffer sizes) and selects the most efficient sort or merge technique.

You can optimize program performance by:

- Using System/370-MVS/XA Operating System.
- Planning your application development (including data formats) for efficient use of the program.
- Using the most efficient sort/merge techniques.
- Planning for optimal use of work storage devices.
- Specifying the input/output data set characteristics correctly.
- Using JCL to initiate DFSORT.
- Using options that may enhance performance.
- Avoiding options that may degrade performance.

These techniques are described in detail below.

Using System/370-MVS/XA Operating Systems

On MVS/XA systems, DFSORT can automatically

- Use virtual storage above 16-megabyte virtual.
- Free space below 16-megabyte virtual.
- Allow more efficient sorting.
- Eliminate the need to change JCL, unless SIZE or MAINSIZE is specified.

In addition, System/370-XA Sorting Instructions can enhance DFSORT's performance when sorting FLR records. DFSORT selects the System/370-XA Sorting Instructions if the following requirements are met:

- The System/370-XA Sorting Instructions are activated.

- FLR records are being sorted.
- The Blockset sorting technique is being used.
- DEBUG NOASSIST is not specified.

Diagnostic message ICE807I indicates whether the System/370-XA Sorting Instructions were used.

Planning Applications

You should consider several factors when you design new applications. Some of these factors are discussed in the following sections.

Efficient Blocking

Performance of DFSORT may be significantly improved if you block your input and output records. A blocksize of approximately 6000 bytes is generally considered to be a good value for DASD data sets. For large files, files on tape, or files which you sort often, you may want to choose a larger blocksize. In general, the larger your input and output blocksizes, the better DFSORT's performance.

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 or merging at the beginning of your record in descending order of significance, and
- Use the most efficient control field data formats and control field descriptions.

Control fields may be contiguous, separated, or overlapping. The control fields may occur anywhere within the first 4092 bytes of a data record, but their total length must not exceed 4092 bytes.

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 or merging control field
2 = the less significant sorting or merging control field
A, B, and C are not sorting or merging control fields.

Internally, the program reorganizes the record fields prior to the actual sorting or merging as follows:

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.

Control Field Data Formats and Descriptions: Whenever possible:

- Use either EBCDIC character or binary control fields.
- Start and end binary control fields on byte boundaries.
- Avoid using the alternate collating sequence character translation, because this function not only increases CPU time, but also increases the total length of the internal record.
- Use packed decimal format rather than zoned decimal, because DFSORT 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.

By carefully designing your application from the beginning with the above considerations, the performance for your sorting applications improves.

Tuning Main Storage

Either the REGION value or the MAINSIZE/SIZE value can be the limiting factor in determining how much storage is available to DFSORT. See *DFSORT Planning and Installation* for more details.

Generally, the most efficient way to allocate (virtual) main storage is to specify MAINSIZE/SIZE=MAX. However, problems can arise if the values for TMAXLIM and/or MAXLIM installation options have been set excessively high (or low). Guidelines for setting these values are given in *DFSORT Planning and Installation*.

Note: Do not use SIZE/MAINSIZE=MAX with password-protected data sets if passwords are to be entered through a routine at an exit, because DFSORT cannot then open the data sets during the initialization phase to make the necessary calculations.

If you specify MAINSIZE/SIZE=n and the value is less than that specified for the MINLIM installation option, MINLIM is used.

If the MINLIM value is greater than that specified for REGION on the EXEC statement, DFSORT attempts to use the value specified for MINLIM; if it fails to get the amount specified by MINLIM, DFSORT still tries to execute, provided at least 88K bytes (below 16-megabyte virtual for MVS/XA) are available to DFSORT.

Although DFSORT requires a minimum of 88K bytes (below 16-megabyte virtual for MVS/XA), the minimum amount of main storage required depends on the application.

You may need more main storage if you use:

- Spanned records
- COBOL user-exit routines
- ALTSEQ or CHALT
- INCLUDE, OMIT, SUM, OUTREC, or INREC (although INREC can also reduce storage requirements by shortening record sizes.)
- Very large blocks or logical records
- VSAM data sets (For more information, see your VSAM manuals.)

Note: In some cases, this release may use more storage than prior releases. This may affect the operation of some jobs. For example, there may be jobs that run as in-storage sorts (with no SORTWKnn data sets) that will not run in-storage when using this release.

When sorting records using Blockset on an MVS/XA system, DFSORT attempts to allocate storage above and below 16-megabyte virtual. The total amount of storage allocated is normally controlled by TMAXLIM. A REGION size of at least 300K bytes should be available if DFSORT is to achieve acceptable performance. The allocation of storage can be adversely affected if you have a smaller region value or if DFSORT needs to allocate buffers below 16-megabyte virtual.

The relationship between TMAXLIM, MAXLIM, MINLIM, and REGION might be described as a series of checks and balances.

Your system programmer has set the default storage values according to your installation's major sorting requirements. If you have an overnight or batch time window that must be met, then increasing storage (using REGION or SIZE/MAINSIZE=n) could give you some relief from the time constraint. If you are concerned with processor time, then decreasing storage (using REGION or SIZE/MAINSIZE=n) could reduce the processor time associated with sorting small files.

In general, when you vary main storage that you make available to DFSORT, the following occurs:

1. If you increase the amount of storage:
   - EXCPs are reduced.
   - On a light to medium-loaded system, elapsed time decreases.
   - On a heavily loaded system, elapsed time could increase because DFSORT could be swapped out more often.
   - Processor time may remain stable or increase because of the overhead in managing the extra storage. For large files (more than 64 megabytes), however, processor time may decrease because the overhead in managing the extra storage would be less than the benefit gained by DFSORT making fewer passes over the data.
2. If you decrease the amount of storage:
   - EXCPs increase.
   - Elapsed time increases for most sorts.
   - Processing time decreases for small files, but increases for large files.

Changing the main storage allocation can affect system efficiency: By reducing the amount of main storage allocated, you impair performance of DFSORT in order to allow other programs to have the storage they need to operate simultaneously; and, by increasing the allocation, you can run large DFSORT applications efficiently at the risk of decreasing the efficiency of other jobs sharing the multiprogramming environment.

How to Get DFSORT to Release Storage

Under some circumstances, DFSORT uses all the available storage in your REGION . For MVS/XA, this normally will not occur for storage above 16-megabyte virtual (if it does, use the ARESINV and/or ARESALL options or lower your SIZE/MAINSIZE value). This section explains how to release storage within your REGION.

When SIZE/MAINSIZE=n and n is greater than the REGION parameter or default REGION value for your sort job, or when SIZE/MAINSIZE=MAX and MAXLIM (or TMAXLIM for MVS/XA systems) is greater than your REGION, specify the storage you need released in the following way:

- For jobs with user exits:
  - For JCL-invoked DFSORT, you can choose one of the following:
    - Use the *m* parameter of the MODS control statement.
    - If SIZE=MAX is in effect, you can use the RESALL option.

— Change your REGION so that REGION is greater than SIZE/MAINSIZE (the difference is available).

— If the installation parameter OVERRGN is smaller than your system IEALIMIT value on MVS/370, this difference is available. (OVERRGN is an installation option that can only be modified by your system programmer).

– For dynamically invoked DFSORT, you can choose one of the following:

— If the exit address is not passed in the parameter list (that is, it is specified with a MODS statement), use the *m* parameter on the MODS statement.

— If the exit address is passed in the parameter list, and SIZE/MAINSIZE=MAX is in effect, use the RESINV option.

— If the exit address is passed in the parameter list, and SIZE/MAINSIZE=n is in effect, change your REGION so that the REGION is greater than SIZE/MAINSIZE (the difference is available).

— If the exit address is passed in the parameter list, and SIZE/MAINSIZE=n is in effect, for many of your sort applications, you should consider having the OVERRGN value changed by your system programmer to less than your IEALIMIT value.

• For jobs without user exits:

– For JCL-invoked DFSORT, you can choose one of the following:

— If SIZE/MAINSIZE=MAX is in effect, use the RESALL option.

— If SIZE/MAINSIZE=n is in effect, change your REGION so that REGION is greater than SIZE/MAINSIZE (the difference is available).

— Have the OVERRGN value changed by your system programmer to less than your IEALIMIT value.

– For dynamically invoked DFSORT, you can choose one of the following:

— If SIZE/MAINSIZE=MAX is in effect, use the RESINV option.

— If SIZE/MAINSIZE=n is in effect, change your REGION so that REGION is greater than SIZE/MAINSIZE (the difference is available).

— Have the OVERRGN value changed by your system programmer to less than your IEALIMIT value.

When SIZE/MAINSIZE is less than REGION, make sure the difference between SIZE/MAINSIZE and your REGION specification value or default provides sufficient storage for system or user exit routine use.

Using Efficient Sort/Merge Techniques

Depending on various conditions, DFSORT selects different techniques for sorting and merging. Message ICE143I informs you which technique has been selected.

For **copy** applications, Blockset is the only technique used. If your program cannot use Blockset, error message ICE160A is issued and DFSORT stops processing.

Sorting Techniques

One condition that affects which sorting technique is selected is the type of device used for intermediate storage. The Blockset, Peerage, and Vale techniques can be used only with disk devices. If you use a tape device, the less efficient tape work data set sort technique is used. The Blockset, Peerage, and Vale techniques are discussed below. For more information on using tape devices for intermediate storage, see "Tape Work Storage Devices" on page 216.

Blockset Sorting Techniques

Fixed-Length Records: DFSORT's most efficient fixed-length record technique, FLR-Blockset, is used for most sorting applications. If one or more of the conditions for the FLR-Blockset technique are not met (for example, if the control field is too long), the Peerage or Vale technique is used.

Variable-Length Records: The high-performance VLR-Blockset technique is used for sorting variable-length records in most cases. If one or more of the conditions for the VLR-Blockset technique are not met (for example, if the control field is too long), the Vale technique is used.

Notes:

1. *The Blockset technique may require more intermediate work space than Peerage or Vale. For more information, see "Using Work Storage Devices Efficiently" on page 212.*

2. *If Blockset is not selected, you can use a SORTDIAG DD statement to force message ICE800I, which gives a code indicating why Blockset can not be used.*

Peerage and Vale Disk Sorting Techniques

If the conditions for use of the Blockset sorting technique are not met, DFSORT uses 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

No figure can be given for how long the control word can be if the Peerage technique is used. The control field's length depends on many variables, such as device type for work storage and amount of main storage available for buffers. However, the length limit is probably greater than 256 bytes.

If any one of the conditions mentioned above is not satisfied, DFSORT will use Vale.

Merging Techniques

For merging applications, DFSORT uses either the Blockset or Conventional technique.

Blockset Merging Technique

DFSORT's high-performance Blockset merging technique is used for merging fixed- and variable-length records, in most cases.

Note: If Blockset is not selected, you can use a SORTDIAG DD statement to force message ICE800I, which gives a code indicating why Blockset cannot be used.

Conventional Merging Technique

If the conditions for use of the Blockset merging technique are not met (for example, if the control field is too long), DFSORT uses the Conventional merging technique.

Using Work Storage Devices Efficiently

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 processor 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. (For best performance, you should always allocate work storage in cylinders. If a temporary sort work data set is not allocated in cylinders, DFSORT reallocates it in cylinders.) The numbers given are estimates of the number of SORTWKnn cylinders sort uses for a particular file size when secondary allocation is allowed. You can make primary and secondary allocations by means of the SORTWKnn 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 is generally degraded.

SORTWKnn Space Used[1]	
File Size in Bytes	3380 Cylinders
3M	8
20M	47
40M	77
150M	284

[1] This example is based on jobs run using the Blockset technique with a SIZE/MAINSIZE parameter of 2048K bytes and one SORTWKnn data set on an IBM 3380.

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=3380), DFSORT assigns areas, and some optimization occurs automatically. Best performance is achieved if you follow these recommendations:

- Use high speed disks for SORTWKnn.
- Assign only one data set per spindle, if you can.
- Try to use the same device type as much as possible.
- Use two channel paths to devices whenever you can.
- Make all data sets the same size, or as near as possible.
- Assign SORTIN, SORTOUT, and SORTWKnn 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 that the automatic secondary allocation is not needed.

Elapsed time is decreased when DFSORT can read input while writing to SORTWKnn, and write output while reading from SORTWKnn. If, for example, you have two channels, the best allocation of them is to have SORTIN and SORTOUT on one and the SORTWKnns on the other.

Storage requirements for different disk techniques can be estimated by using the guidelines found in Appendix B.

Device Data Transfer Rate

In general, the faster the data transfer rate of the storage device, the faster the sort. Figure 35 on page 217 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. For example, the 3880 Model 2 or 3 with the speed matching Buffer Feature permits attachment of the IBM 3380 to systems with block multiplexer channels with data rates less than 3 megabytes per second.

Also, the 3880 Model 1 with the Speed Matching Buffer Feature permits attachment of the IBM 3375 to systems with block multiplexer channels with data rates less than 3 megabytes per second.

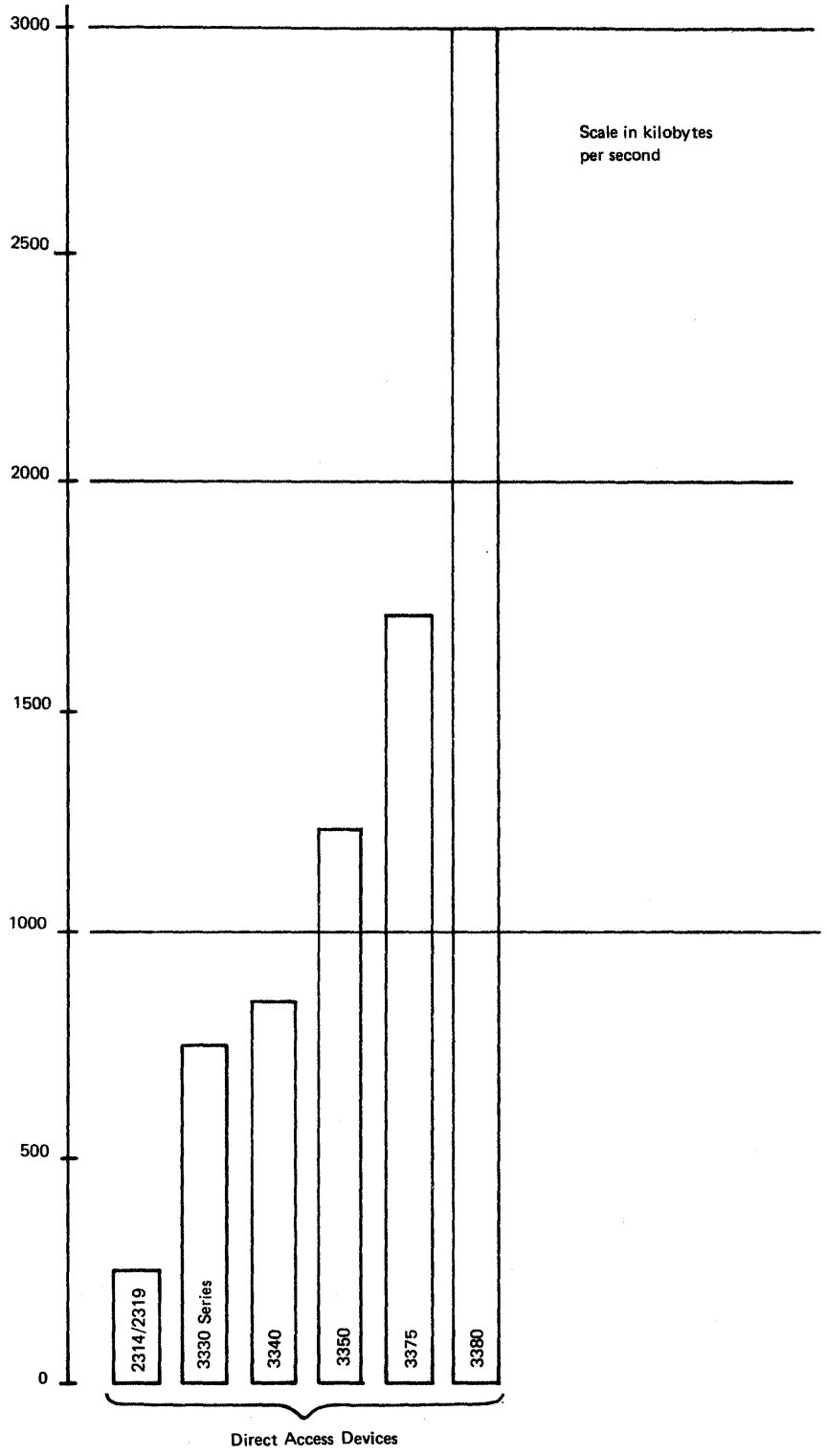


Figure 35. Comparative Data Transfer Rates of Disk Work Storage Devices

Tape Work Storage Devices

Best performance, using tape intermediate storage, is usually 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 then reduces elapsed time and often the number of I/O operations.

Increasing the number of work units, however, 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 DFSORT has only 88K bytes in which to operate, you 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 be taken to apply with more than 100K bytes available for DFSORT.

For information on how to determine storage requirements when using different tape techniques, see Appendix B.

Note: Frequency of tape direction changes, which occur during DFSORT workfile operations, have more of an impact on the effective data rate of IBM 3480 Magnetic Tape Subsystems than on IBM 3420 Magnetic Tape Units. Because of this characteristic, performance comparisons between these tape units for intermediate storage cannot be reliably predicted and may vary widely.

Specifying Input/Output Data Set Characteristics

DFSORT 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, can 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 so 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 DFSORT 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 OPTION or 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 DFSORT is an accurate data set size in the FILSZ parameter of the OPTION or SORT statement.

Variable-Length Records

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.

Using JCL to Initiate DFSORT

Many installations invoke DFSORT from a COBOL or PL/I program instead of initiating DFSORT by means of JCL. Because this is generally done for convenience, you should be aware that the trade-off may be degraded performance. DFSORT defaults specified at installation time and options specified at execution time should be fine-tuned for optimum performance, especially to make use of control statements that "work together," such as INCLUDE/OMIT, INREC/OUTREC, and SUM. Using these functions, you can eliminate records from input files, arithmetically combine records, and reformat records to eliminate unwanted fields.

Using Options That May Enhance Performance

COBEXIT

To take advantage of the COBOL II interface with DFSORT, and thereby enhance performance, specify the COBEXIT=COB2 when running exits compiled with VS COBOL II.

FASTSRT

By specifying the VS COBOL II FASTSRT compile-time option, you can significantly reduce DFSORT processor time, EXCPs, and elapsed time. With FASTSRT, sorting input/output operations are more efficient because DFSORT rather than COBOL does the input and output (see Figure 36 on page 218). For more details, see your COBOL manuals.

The FASTSRT option does not take effect for input and output if input and output procedures are used in the SORT statement. Many of the functions usually performed in an input or output procedure are the same as those done by DFSORT INREC, OUTREC, INCLUDE, OMIT, STOPAFT, SKIPREC, and SUM functions. You may be able to eliminate your input and output procedures by coding the appropriate DFSORT program control statements and placing them in

either the SORTCNTL (DFSORT) or IGZSRTCD (COBOL) data set, thereby allowing your SORT statement to qualify for FASTSRT.

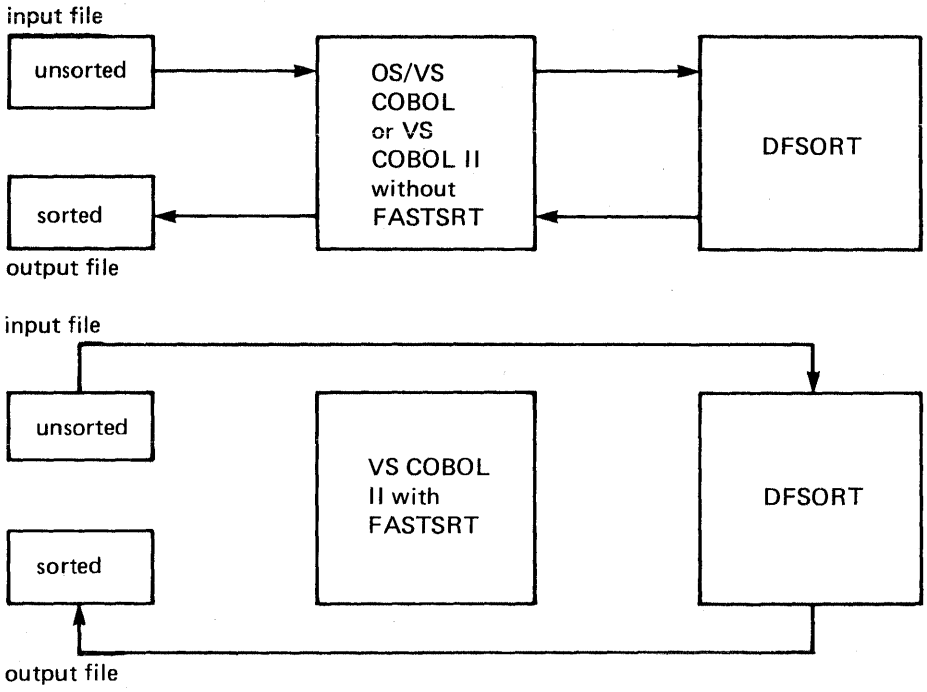


Figure 36. Faster Sorting with VS COBOL II

INCLUDE OR OMIT, STOPAFT, AND SKIPREC

You can use either the INCLUDE or OMIT statement and the STOPAFT and/or SKIPREC option to reduce the size of the input file. Reducing the size of the input file may reduce processor and data transfer time.

- The INCLUDE and OMIT statements allow you to select records by comparing fields with constants and/or other fields.
- The STOPAFT option allows you to specify the maximum number of records that should be accepted for sorting or copying.
- The SKIPREC option allows you to skip records at the beginning of the input file for sorting or copying.

INREC and OUTREC

You can use the INREC statement to reformat the input records *before* processing. This can provide more efficient processing if you reduce the size of the records (or less efficient processing if you increase the size of the records).

OUTREC can be used to lengthen the record after processing, aligning the data fields and introducing blanks to separate fields to make the output more legible.

When INREC and/or OUTREC is used with INCLUDE or OMIT, only the records that remain in the data set are reformatted after the process of elimination initiated by INCLUDE or OMIT.

You must be aware of the change in record size and layout of the resulting reformatted input/output records, and of which DFSORT functions must refer to the layout of the reformatted input records, rather than the layout of the original input records.

Three types of fields may be removed by INREC/OUTREC:

1. Padding fields (blanks or binary zeros) may be removed before processing by using INREC, and then reinserted after processing by using OUTREC.

2. Fields that are not needed in the output records may be removed before processing by using INREC.

3. If a variable-length file is being processed and only the fixed part of the record is needed, the variable part may be eliminated before processing. This allows DFSORT to manage the records internally as if they were fixed length, and can result in significantly more efficient processing.

SUM

You can use the SUM statement to summarize records. SUM is processed after SORT or MERGE, INCLUDE, OMIT, and/or INREC processing has completed. Thereafter, whenever two records with equal control fields are found, the contents of fields defined in the SUM statement are added, the result is placed in one record, and the other is deleted; any resulting reduction in the number of records to be sorted or merged by DFSORT saves processor time and data transfer time.

For a diagram of the processing sequence for record handling statements, exits, and options, see Figure 2 on page 10.

Avoiding Options That May Degrade Performance

VERIFY

The VERIFY option affects performance negatively, because it involves an extra read operation of the written output.

EQUALS

The EQUALS option causes an additional field of four bytes to be added to each record, which increases the time needed for comparison of records and for data transfer. This does not apply to the Blockset technique for sorting variable-length records, which always uses EQUALS.

NOWRKSEC

The NOWRKSEC option prevents automatic allocation of secondary work data set extents. This will cause reuse of the available extents involving extra reads and writes.

NOBLKSET

The NOBLKSET option precludes the use of the more efficient Blockset technique.

CKPT

The CKPT option may preclude the use of the more efficient Blockset technique.

Note: If the installation default, IGNCKPT=YES, has been selected, DFSORT ignores the checkpoint/restart request and select the Blockset technique.

Tape Work Data Sets

Use of tape work data for intermediate storage precludes the use of the much more efficient disk techniques.

User Routines

When user routines are included in an application, the time required to run the job is usually increased.

The execution time required by most user routines is generally small, but the routines at exits E15, E32, and E35 are entered for each record of the data set(s). For large input data sets, the total execution time of these routines can be relatively large.

Dynamic Link-Editing

Dynamic link-edit of user exit routines degrades performance.

Appendix A. Sample Job Streams

The table below describes the examples that are provided in this appendix.

No.	Description	Input	Output
1	Disk sort	Blocked fixed-length records on 3380	Blocked fixed-length records on 9-track
2	3380 sort with exits	Blocked fixed-length records on 3380	Blocked fixed-length records on 3380, same unit as input
3	Sort, one exit, PROC=SORT	Fixed-length unblocked records on 3380	Fixed-length blocked records on 3380
4	3380 sort, tape I/O, exits	Variable-length records on 3400 tape	Variable-length records on 3400 tape
5	COPY using OPTION control statement	Variable-length records	Variable-length records
6	Disk sort, ISCII/ASCII tape I/O	Variable-length ISCII/ASCII records on 9-track tape	Variable-length ISCII/ASCII records on 9-track tape
7	3380 sort, ISCII/ASCII tape I/O	Variable-length ISCII/ASCII records on 9-track tape	Variable length ISCII/ASCII records on 9-track tape
8	Disk sort	Blocked fixed-length records on 9-track tape	Blocked fixed-length records on 9-track tape
9	Disk sort with exits	Fixed-length blocked records on two unlabeled 9-track volumes	Fixed-length blocked records on one 9-track tape
10	3380 sort, exits	Variable-length blocked records on 3350	Variable-length blocked records on 3380
11	Sort with no SORTWKnn, 1 exit	Fixed-length blocked records on 3380	Fixed-length blocked records on 3380
12	Concatenated input, dynamically allocated work areas	A concatenation of three data sets, two on 3380 and one on 3400-3	Blocked fixed-length records on 9-track tape
13	3350 sort called from another program	Fixed-or variable-length records	Fixed-or variable-length records

No.	Description	Input	Output
14	3350 sort using options NOOUTREL, DYNALLOC, DEBUG, and FMTABEND	Blocked fixed-length records	Blocked fixed-length records
15	3350 sort using control statements OMIT, INREC, and SUM	Blocked fixed-length records	Blocked fixed-length records
16	COPY using SORT Control Statement	Blocked fixed- or variable-length records	Blocked fixed- or variable-length records
17	Sort using COBOL exits	Fixed- or variable-length records	Fixed- or variable-length records
18	Dynamic link-editing of user exit routines	Fixed- or variable-length records	Fixed- or variable-length records
19	3380 sort using extended parameter list interface	Blocked fixed-length records	Blocked fixed-length records
20	Merge four files	Blocked fixed-length records on two 3350s and two 3380s	Blocked fixed-length records on one 9-track tape
21	Merge two 3350 files, exits	Variable-length blocked records on 3350	Variable-length blocked records on 3350
22	Merge with equals and sum	Fixed- or variable-length records	Fixed- or variable-length records
23	Define VSAM cluster for DFSORT use	VSAM variable-length records	VSAM variable-length records
24	Sort with VSAM input and output	VSAM variable-length records	VSAM variable-length records
25	Sort with VSAM output, exit	Variable-length records	VSAM variable-length records

Sort Examples

Example 1. DISK SORT

```
//EXAMP   JOB  A402,PROGRAMMER,REGION=512K                          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)                                         10
* KEEP ESTIMATED FILSZ UP-TO-DATE                                  11
   OPTION FILSZ=E2000                                              12
/*
```

This example is the same as that shown in Chapter 1.

Line	Explanation
01	The JOB statement introduces this job to the operating system, and specifies a region of 512K 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 messages and control statements to system output class A.
04	The SORTIN DD statement describes a temporary 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 is recorded on a 9-track tape and is kept. The data set is 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.
11	Comment statement.
12	OPTION statement. The file size is estimated to be 2000 records.

Example 2. 3380, PROC=SORTD, EXITS

INPUT	Blocked fixed-length records on 3380.
OUTPUT	Blocked fixed-length records on 3380, same unit as input.
INTERMEDIATE STORAGE	Three 3380 areas of 10 cylinders each.
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; maximum main storage allocation.

```
//EXAMP    JOB  A402,PROGRAMMER
//STEP1    EXEC   SORTD,PARM='SIZE=MAX'                              01
//SORTIN   DD   UNIT=3380,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=(3380,SEP=(SORTIN,SORTOUT)),                   07
//           SPACE=(CYL,(10),,CONTIG)                                08
//SORTWK02 DD   UNIT=(3380,SEP=(SORTIN,SORTOUT)),                   09
//           SPACE=(CYL,(10),,CONTIG)                                10
//SORTWK03 DD   UNIT=(3380,SEP=(SORTIN,SORTOUT)),                   11
//           SPACE=(CYL,(10),,CONTIG)                                12
//MODLIB   DD   DSNAME=YOURRTNS,DISP=SHR                            13
//SYSIN    DD   *                                                   14
   SORT    FIELDS=(3,8,ZD,E,40,6,CH,D)                             15
   OPTION  FILSZ=E30000                                            16
   RECORD  TYPE=F,LENGTH=(,100,80)                                 17
   MODS    E15=(MODREC,784,MODLIB),E16=(E16,1024,MODLIB),          18
           E35=(ADDUP,912,MODLIB),E61=(CHGE,1000,MODLIB)           19
/*
```

Line	Explanation
01	The EXEC statement specifies the SORTD cataloged procedure. 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 3380. DCB parameters are supplied by the system (since DISP=OLD). The data set is 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 three work data sets on an IBM 3380. 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 user routines.

14 A data set follows in the input stream.

15 SORT statement. The FIELDS operand describes two control fields. The first is 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 is not modified and is placed in descending order.

16 OPTION statement. The file size is estimated to be 30000 records.

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 in a library that is defined on a job control statement with the ddname MODLIB; these routines have the member names MODREC, E16, ADDUP, and CHGE.

Example 3. SYSDA SORT, PROC=SORTD, 1 EXIT

INPUT	Fixed-length unblocked records on a 3380 DASD.
OUTPUT	Fixed-length blocked records on a 3380 DASD.
INTERMEDIATE STORAGE	Three SYSDA 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.

```
//EXAMP     JOB   A402,PROGRAMMER
//STEP1     EXEC  PROC=SORTD,PARM='SIZE=MAX,MSGPRT=NONE'        01
//SORTIN    DD    DSNAME=INFILE,VOL=SER=INP214,UNIT=3380,       02
//            DCB=(RECFM=F,BLKSIZE=80),                         03
//            DISP=(OLD,DELETE)                                 04
//SORTOUT   DD    DSNAME=OUTFILE,VOL=SER=DLIB02,UNIT=3380,      05
//            DCB=(RECFM=FB,LRECL=50,BLKSIZE=500),              06
//            DISP=(NEW,KEEP),SPACE=(CYL,(8,1))                 07
//SORTWK01  DD    UNIT=SYSDA,SPACE=(CYL,(1))                    08
//SORTWK02  DD    UNIT=SYSDA,SPACE=(CYL,(1))                    09
//SORTWK03  DD    UNIT=SYSDA,SPACE=(CYL,(1))                    10
//USERLIB   DD    DSN=EX35,DISP=SHR                             11
//SYSIN     DD    *                                             12
  SORT      FIELDS=(10,5,CH,A)                                  13
  OPTION    FILSZ=1000                                          14
  RECORD    TYPE=F,LENGTH=(,,50)                                15
  MODS      E35=(E35,536,USERLIB)                               16
/*
```

Line	Explanation
01	Invokes the SORTD cataloged procedure; specifies that the maximum amount of main storage available is to be allocated for the program's execution, and that messages and control statements are not to be printed.
02-04	The input data set consists of fixed-length unblocked records on volume INP214 on a 3380. The data set is deleted after this job step.
05-07	The output data set is composed of fixed-length blocked records that requires 8 cylinders on a 3380. Each time space is exhausted, an additional cylinder is allotted. The data set is retained.
08-10	Intermediate storage consists of three SYSDA areas of one cylinder each.
11	Defines the library that contains the E35 module.
12	A data set follows in the input stream.
13	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.

14 OPTION statement. The input data set contains exactly 1000 records.

15 RECORD statement. Indicates that the input data set contains fixed-length records that are shortened to 50 bytes each as they leave the final merge.

16 MODS statement. Describes a user routine that receives control at program exit E35. The name of the routine is E35; it is 536 bytes long and is on the data set defined in the USERLIB DD statement.

Example 4. 3380 SORT, TAPE I/O, PROC=SORTD, EXITS

INPUT	Variable-length records on 3400 tapes.
OUTPUT	Variable-length records on 3400 tapes.
INTERMEDIATE STORAGE	Two 3380 areas of 15 cylinders each.
USER ROUTINES	E11 routine performs initialization for the E16 Nmax routine.
OPTIONS	Estimated data set size.

```
//EXAMP     JOB   B999,PROGRAMMER
//STEPN     EXEC  SORTD,REGION=512K                                  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=3380,SPACE=(CYL,(15))                         05
//SORTWK02  DD    UNIT=3380,SPACE=(CYL,(15))                         06
//SORTOUT   DD    DSNAME=M999999.YFILE,VOL=SER=000258,               07
//            UNIT=3400-3,DISP=(NEW,CATLG)                           08
//USERLIB   DD    DSNAME=MYRTNS,DISP=SHR                             09
//           DD   DSNAME=MORTNS,DISP=SHR                             10
//SYSIN     DD    *                                                  11
   SORT     FIELDS=(20,5,AQ,A)                                       12
   OPTION   FILSZ=E25500                                             13
   RECORD   TYPE=V,LENGTH=(120,,,80,120)                             14
   MODS     E11=(PREPMOD,504,USERLIB),E16=(MODMAX,554,               15
            USERLIB)                                                 16
   ALTSEQ   CODE=(5BEA,7BEB,7CEC)                                    17
/*
```

Line	Explanation
01	Calls the SORTD cataloged procedure and indicates that a 512K-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 3380s 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 contains 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	Defines the library that contains the E11 user routine.

11 A data set follows in the input stream.

12 SORT statement. Describes one control field that begins on byte 16 of each record data area (not byte 20, because 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.

13 OPTION statement. The input data set contains approximately 25500 records.

14 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 assumes 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.

15-16 MODS statement. Describes two user routines. The first, PREPMOD, receives control at exit E11. It is 504 bytes long and resides in MORTNS. The second user routine, named MODMAX, receives control at exit E16. It is 554 bytes long and resides in MYRTNS.

17 ALTSEQ statement. Specifies that the three characters $, #, and @ are to collate in that order after Z.

Example 5. COPY WITH OPTION CONTROL STATEMENT

INPUT	Variable-length records
OUTPUT	Variable-length records
INTERMEDIATE STORAGE	None
USER ROUTINES	None
OPTION	COPY,STOPAFT,ABSTP

```
//EXAMP     JOB   B999,PROGRAMMER01
//S1 EXEC   PGM=SORT,PARM='MSGPRT=CRITICAL'                    02
//SYSOUT    DD    SYSOUT=A                                     03
//SORTIN    DD    DSNAME=INV1,DISP=OLD                         04
//SORTOUT   DD    DSNAME=OUTV1,DISP=(NEW,KEEP),                05
//             UNIT=3380,SPACE=(TRK,(15,2)),VOL=SER=XYZ003     06
//SYSIN     DD    *                                            07
   OPTION   STOPAFT=500,COPY                                   08
   DEBUG    ABSTP                                              09
/*
```

Line	Explanation
01	The JOB statement introduces this job to the operating system.
02	The EXEC statement calls the program by its alias SORT. MSGPRT=CRITICAL in the EXEC parm field specifies that only error messages are to be printed.
03	The SYSOUT DD statement directs DFSORT messages to output class A.
04	The SORTIN DD statement describes a cataloged variable length record input data set named INV1.
05-06	The SORTOUT DD statement directs the output to a new data set named OUTV1 on volume XYZ003 of a 3380.
07	The SYSIN DD statement indicates that data follows in the input stream.
09	THE OPTION statement indicates that a copy is to be done and that only the first 500 records are to be copied.
10	The DEBUG statement indicates that if this application fails in the Blockset execution phase, an abend occurs with a user completion code that is equal to the appropriate message number. (The message is NOT written.)

Example 6. SYSDA SORT, ISCII/ASCII TAPE I/O, PROC=SORTD

INPUT	Variable-length ISCII/ASCII records on 9-track tape.
OUTPUT	Variable-length ISCII/ASCII records on 9-track tape.
INTERMEDIATE STORAGE	Two SYSDA areas of 15 cylinders each and two SYSDA areas of 10 cylinders each.
USER ROUTINES	None.
OPTIONS	Estimated data set size.

```
//EXAMP     JOB   A432,PROGRAMMER
//STEPM     EXEC  SORTD
//SORTIN    DD    DSNAME=SRTFIL,DISP=(OLD,DELETE),UNIT=3400-6,    01
//            DCB=(RECFM=DB,LRECL=80,BLKSIZE=404,OPTCD=Q,         02
//            BUFOFF=L),VOL=SER=311500,LABEL=(1,AL)               03
//SORTWK01  DD    UNIT=SYSDA,SPACE=(CYL,(15))                     04
//SORTWK02  DD    UNIT=SYSDA,SPACE=(CYL,(15))                     05
//SORTWK03  DD    UNIT=SYSDA,SPACE=(CYL,(10))                     06
//SORTWK04  DD    UNIT=SYSDA,SPACE=(CYL,(10))                     07
//SORTOUT   DD    DSN=OUTFIL,UNIT=3400-6,LABEL=(,AL),             08
//            DISP=(,KEEP),DCB=(OPTCD=Q,BUFOFF=L),VOL=SER=311501 09
//SYSIN     DD    *                                               10
  SORT      FIELDS=(10,8,AC,D)                                    11
  OPTION    FILSZ=E525000                                         12
  RECORD    TYPE=D,LENGTH=(,,,20,23)                              13
/*
```

Line **Explanation**

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 ISCII/ASCII records that 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 ISCII/ASCII to EBCDIC (OPTCD=Q).

04-07 Four intermediate storage data sets are defined on SYSDA.

08-09 The output data set is named OUTFIL. It is written on a 9-track tape with a density of 6250 bpi and is retained. It has an ISCII/ASCII label and contains records with the same RECFM, LRECL, and BLKSIZE values as the input (by default).

10 A data set follows in the input stream.

11 SORT statement. The FIELDS operand describes a control field that begins on byte 6 of each record data area (not byte 10, because the record descriptor word takes 4 bytes), and is 8 bytes long. This field contains character (ISCII/ASCII) data, and is sorted in descending order.

12 OPTION statement. The input data set contains approximately 525000 records.

13 RECORD statement. All the records in the input data sets are ISCII/ASCII records. Their maximum length is supplied by default; the minimum is 20. The average length is 23.

Example 7. 3380 SORT, ISCII/ASCII TAPE I/O, PROC=SORTD

INPUT	Variable-length ISCII/ASCII records on 9-track tape.
OUTPUT	Variable-length ISCII/ASCII records on 9-track tape.
INTERMEDIATE STORAGE	One 3380 area of 4 cylinders.
USER ROUTINES	None.
OPTIONS	Estimated data set size.

```
//EXAMP      JOB   A432,PROGRAMMER
//STEPM      EXEC  SORTD
//SORTIN     DD    DSNAME=SRTFIL,DISP=(OLD,DELETE),UNIT=3400-6,    01
//              DCB=(RECFM=D,LRECL=400,BLKSIZE=404,OPTCD=Q,        02
//              BUFOFF=L),VOL=SER=311500,LABEL=(1,AL)              03
//SORTWK01   DD    UNIT=3380,SPACE=(CYL,(4))                      04
//SORTOUT    DD    DSN=OUTFIL,UNIT=3400-6,LABEL=(,AL),            05
//              DISP=(,KEEP),DCB=(OPTCD=Q,BUFOFF=L),VOL=SER=311501 06
//SYSIN      DD    *                                              07
   SORT      FIELDS=(10,8,AC,D)                                   08
   OPTION    FILSZ=E26000                                         09
   RECORD    TYPE=D,LENGTH=(,,,20,80)                             10
/*
```

Line	Explanation
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 ISCII/ASCII records which are blocked and have a maximum length of 400 bytes. It has an ISCII/ASCII label. For this job, the buffer offset is the block length indicator. The records are to be translated from ISCII/ASCII to EBCDIC (OPTCD=Q).
04	One intermediate storage data set is defined on a 3380.
05-06	The output data set OUTFIL is written on a 9-track tape with a density of 6250 bpi and the volume serial number 311501. It has an ISCII/ASCII label and is kept. It contains records with the same RECFM, LRECL, and BLKSIZE values as the input (by default).
07	A data set follows in the input stream.
08	SORT statement. The FIELDS operand describes a control field that begins on byte 6 of each record data area (not byte 10, because the record descriptor word takes 4 bytes), and is 8 bytes long. This field contains character (ISCII/ASCII) data, and is sorted in descending order.
09	OPTION statement. The input data set contains approximately 26000 records.

10 RECORD statement. All the records in the input data sets are ISCII/ASCII records. Their maximum length is supplied by default; the minimum is 20. The average length is 80.

Example 8. DISK SORT, PGM=SORT

INPUT	Blocked fixed-length records on 9-track tape.
OUTPUT	Blocked fixed-length records on 9-track tape.
INTERMEDIATE STORAGE	Four SYSDA devices with 10 cylinders each.
USER ROUTINES	None.
OPTIONS	FORMAT=xx for control fields of like format; estimated data set size.

```
//EXAMP     JOB   A402,PROGRAMMER
//STEP1     EXEC  PGM=SORT,REGION=512K                              01
//SYSOUT    DD    SYSOUT=A                                          02
//SORTLIB   DD    DSNAME=SM01.SORTLIB,DISP=SHR                      03
//SORTIN    DD    DSNAME=INPUT,VOL=SER=000101,UNIT=3400-3,          04
//            DISP=(OLD,DELETE),DCB=(RECFM=FB,                      05
//            LRECL=80,BLKSIZE=800)                                 06
//SORTOUT   DD    DSNAME=M999999.OUTPUT,UNIT=3400-3,                07
//            DISP=(NEW,CATLG),VOL=SER=000102                       08
//SORTWK01  DD    UNIT=SYSDA,SPACE=(CYL,(10))                       09
//SORTWK02  DD    UNIT=SYSDA,SPACE=(CYL,(10))                       10
//SORTWK03  DD    UNIT=SYSDA,SPACE=(CYL,(10))                       11
//SORTWK04  DD    UNIT=SYSDA,SPACE=(CYL,(10))                       12
//SYSIN     DD    *                                                 13
   SORT     FIELDS=(1,6,A,28,5,D),FORMAT=CH                         14
   OPTION   FILSZ=E10000                                            15
/*
```

Line	Explanation
01	This EXEC statement calls the program module by its alias, SORT, and indicates that it wants a 512K-byte region in which to operate.
02	The SYSOUT DD statement directs messages and control statements 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 SYSDA devices with 10 cylinders each.
13	A data set follows in the input stream.

14 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.

15 OPTION statement. The file size is estimated at 10000 records.

Example 9. DISK SORT, PROC=SORTD, EXITS

INPUT	Fixed-length blocked records on two unlabeled 9-track tap volumes.
OUTPUT	Fixed-length blocked records on one 9-track tape.
INTERMEDIATE STORAGE	Four SYSDA devices with 1 cylinder each.
USER ROUTINES	Three: two change record lengths, one decides what to do if Nmax is exceeded.
OPTIONS	Estimated data set size

```
//EXAMP     JOB   A402,PROGRAMMER
//STEP1     EXEC  SORTD                                              01
//SORTIN    DD    DSNAME=INPUT,VOL=SER=(000333,000343),              02
//            UNIT=(3400-3,2),DISP=(OLD,DELETE),LABEL=(,NL),         03
//            DCB=(RECFM=FB,LRECL=120,BLKSIZE=480)                   04
//SORTOUT   DD    DSNAME=OUTPUT,UNIT=3400-3,DISP=(NEW,PASS),         05
//            VOL=SER=456,DCB=(RECFM=FB,LRECL=80,                    06
//            BLKSIZE=3200)                                          07
//SORTWK01  DD    UNIT=SYSDA,SPACE=(CYL,(1))                         08
//SORTWK02  DD    UNIT=SYSDA,SPACE=(CYL,(1))                         09
//SORTWK03  DD    UNIT=SYSDA,SPACE=(CYL,(1))                         10
//SORTWK04  DD    UNIT=SYSDA,SPACE=(CYL,(1))                         11
//MODLIB    DD    DSNAME=YOURRTNS,DISP=SHR                           12
//SYSIN     DD    *                                                  13
   SORT     FIELDS=(3,8,ZD,A,40,6,CH,D)                            14
   OPTION   FILSZ=E30000                                           15
   RECORD   TYPE=F,LENGTH=(120,100,80)                             16
   MODS     E15=(MODREC,784,MODLIB),                               17
      E16=(E16,1024,MODLIB),E35=(ADDUP,912,MODLIB)                 18
                                                                   19
/*
```

Line	Explanation
01	Specifies the cataloged procedure SORTD.
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. Because 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 SYSDA devices with 1 cylinder each.
12	Describes a data set containing the load modules of the user exit routines.
13	A data set follows in the input stream.
14	SORT statement. The FIELDS operand describes two control fields.

15 OPTION statement. The file size is estimated at 30000 records.

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 in a library that is defined on a job control statement with the ddname MODLIB.

Example 10. 3380 SORT, PROC=SORTD, EXITS

INPUT	Variable-length blocked records on 3350.
OUTPUT	Variable-length blocked records on 3380.
INTERMEDIATE STORAGE	One 3380 area of 6 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.

```
//EXAMP    JOB   A402,PROGRAMMER
//STEPONE  EXEC  SORTD,PARM='MSGPRT=CRITICAL,LIST'                01
//SORTIN   DD    UNIT=3350,DSNAME=PAY413,VOL=SER=335001,          02
//           DISP=(OLD,KEEP)                                      03
//SORTOUT  DD    UNIT=3380,DSNAME=PAY414,VOL=SER=335004,          04
//           SPACE=(CYL,(15),RLSE),DISP=(NEW,KEEP)                05
//SORTWK01 DD    UNIT=3380,SPACE=(CYL,(6),,CONTIG)                06
//USERLIB  DD    DSNAME=JIMSMODS,DISP=SHR                         07
//SYSIN    DD    *                                                08
   SORT    FIELDS=(20,5,AQ,A)                                     09
   OPTION  FILSZ=E17000                                           10
   RECORD  TYPE=V,LENGTH=(,,,80,120)                              11
   ALTSEQ  CODE=(5BEA,7BEB,7CEC)                                  12
   MODS    E11=(PREPMOD,504,USERLIB),E16=(MODMAX,554,             13
           USERLIB)                                               14
/*
```

Line	Explanation
01	Specifies the SORTD cataloged procedure. The PARM options indicate that critical messages and program control statements are to be printed.
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 takes 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 3380. It is being created in this job step, and is to be retained. Data set DCB parameters is the same as for SORTIN, by default. Unused space is released.
06	One intermediate storage data set is defined on a 3380.
07	Defines a data set called JIMSMODS, which contains the user exit routines 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.
08	A data set follows in the input stream.

09 SORT statement. The FIELDS operand describes one control field that begins on byte 16 of each record data area (not byte 20, because 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.

10 OPTION statement. The input data set contains approximately 17000 records.

11 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 assumes 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.

12 ALTSEQ statement. Specifies that the three characters $, #, and @ are to collate in that order after Z.

13-14 MODS statement. Describes two user routines. The first, PREPMOD, receives control at exit E11. It is 504 bytes long. The second routine, named MODMAX, receives control at exit E16. It is 554 bytes long. The library in which both reside is described in the job control statement with the ddname USERLIB.

Example 11. SORT WITH NO SORTWKnn, PROC=SORTD, 1 EXIT

INPUT	Fixed-length blocked records on 3380.
OUTPUT	Fixed-length blocked records on 3380.
INTERMEDIATE STORAGE	None.
USER ROUTINES	One routine shortens the records as they leave the final merge phase.
OPTIONS	Exact data set size.

```
//EXAMP    JOB   B600,PROGRAMMER
//STEP1    EXEC  PROC=SORTD,PARM='SIZE(130000)'
//SORTIN   DD    DSNAME=INPUT,UNIT=3380,VOL=SER=333001,      01
//           DISP=SHR                                        02
//SORTOUT  DD    DSNAME=OUTPUT,UNIT=3380,VOL=SER=334010,     03
//           DCB=(RECFM=FB,LRECL=50,BLKSIZE=500),            04
//           DISP=(NEW,KEEP),SPACE=(CYL,(1,1),RLSE)          05
//ERTNLIB  DD    DSN=EXITS,DISP=SHR                          06
//SYSIN    DD    *                                           07
   SORT    FIELDS=(10,5,CH,A)                                08
   OPTION  FILSZ=800                                         09
   RECORD  TYPE=F,LENGTH=(,,50)                              10
   MODS    E35=(E35,534,ERTNLIB)                             11
/*
```

No work areas are defined. If all records cannot be sorted in main storage, the program terminates.

Line **Explanation**

01-02 The input data set is named INPUT, is on a 3380 volume 333001, and consists of fixed-length records with a length of 80 bytes. The DCB information is taken from the data set label.

03-05 The output data set, named OUTPUT, is on volume 334010 of a 3380 and contains 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 is released. Records have been shortened at E35, so DCB information is different from SORTIN and therefore has to be specified.

06 Defines a library that contains the E35 routine.

07 A data set follows in the input stream.

08 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.

09 OPTION statement. The input data set contains exactly 800 records.

10 RECORD statement. Indicates that the input data set contains fixed-length records and that the record length is changed to 50 bytes as records leave the final merge.

11 MODS statement. Describes a user exit routine that receives control at E35 exit. The name of the routine is E35; it is 534 bytes long and resides in the data set described in the ERTNLIB DD statement.

Example 12. CONCATENATED INPUT, DYNAMICALLY ALLOCATED WORK AREAS

INPUT	A concatenation of three data sets, two on 3380 and one 3400-3.
OUTPUT	Blocked fixed-length records on 9-track tape.
INTERMEDIATE STORAGE	Two 3350 areas.
USER ROUTINES	None.
OPTIONS	FORMAT parameter for control fields of like format; estimated data set size.

```
//EXAMP    JOB   A400,PROGRAMMER
//STEPT    EXEC  PGM=ICEMAN,REGION=512K                              01
//SYSOUT   DD    SYSOUT=A                                           02
//SORTLIB  DD    DSNAME=SYS1.SORTLIB,DISP=SHR                       03
//SORTIN   DD    DSNAME=INP1,DISP=OLD,UNIT=3380,                    04
//           DCB=(RECFM=FB,BLKSIZE=7200,LRECL=80),                  05
//           VOL=SER=XB0001                                         06
//         DD    DSNAME=INP2,DISP=OLD,UNIT=3400-3,                  07
//           DCB=(RECFM=FB,BLKSIZE=4000,LRECL=80),                  08
//           VOL=SER=T33333                                         09
//         DD    DSNAME=INP3,DISP=OLD,UNIT=3380,                    10
//           DCB=(RECFM=FB,BLKSIZE=3600,LRECL=80),                  11
//           VOL=SER=DISK01                                         12
//SORTOUT  DD    DSNAME=M999999.OUTPUT,UNIT=3400-3,                 13
//           DISP=(NEW,CATLG),VOL=SER=000102,DCB=(BLKSIZE=800)      14
//SYSIN    DD    *                                                  15
   SORT    FIELDS=(1,6,A,28,5,D),FORMAT=CH                         16
   OPTION  FILSZ=E10000,DYNALLOC=(3350,2)                          17
```

Example 11 differs from Example 7 in three ways: The input is a concatenation of three input data sets on unlike devices; the region specified is 512K bytes; and work storage is dynamically allocated on tape.

Line **Explanation**

01 Indicates that a 512K-byte region is needed.

02 The SYSOUT DD statement directs messages and control statements to system output class A.

03 Sort program modules required for a sort using tape work files 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 3380. 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 3380 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 A data set follows in the input stream.

16 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.

17 OPTION statement. The FILSZ operand indicates that the input data set contains an estimated 10000 records. The DYNALLOC operand indicates that two work data sets are to be dynamically allocated on 3350 (valid only when DFSORT is running under MVS).

Example 13. 3350 SORT USING SORTCNTL AND OPTION

INPUT	Fixed- or variable-length blocked records.
OUTPUT	Fixed- or variable-length blocked records.
INTERMEDIATE STORAGE	One 3350 area of 5 cylinders.
USER ROUTINES	None.
OPTIONS	Exact size file and alternate collating sequence for EBCDIC fields.

```
//EXAMP    JOB   A402,PROGRAMMER
//SORT1    EXEC  PGM=MYPGM                                  01
//STEPLIB  DD    DSN=NAME1.NAME2.NAME3,DISP=SHR             02
//SYSOUT   DD    SYSOUT=A                                   03
//SYSPRINT DD    SYSOUT=A                                   04
//SORTIN   DD    DSN=M999999.INPUT.FILE,DISP=SHR            05
//SORTWK01 DD    UNIT=3350,SPACE=(CYL,(5))                  06
//SORTOUT  DD    DSN=MY.OUTPUT.FILE,UNIT=SYSDA,             07
//           SPACE=(CYL,(3,2)),DISP=(NEW,CATLG)             08
//SORTCNTL DD    *                                          09
   OPTION  FILSZ=2270,CHALT                                 10
/*
```

Line	Explanation
01	Specifies the name of the program calling DFSORT.
02	The STEPLIB DD statement describes where MYPGM is located.
03	The SYSOUT DD statement directs messages and control statements to system output class A.
04	MYPGM output is to be directed to system output class A.
05	The SORTIN DD statement describes an input data set named M999999.INPUT.FILE. The DISP parameter indicates that the data set is known to the operating system.
06	The SORTWK01 DD statement describes a work data set on a 3350. The area contains five cylinders.
07-08	The SORTOUT DD statement describes an output data set named M999999.OUTPUT.FILE. The DISP parameter indicates that the data set is new and will be cataloged.
09	The SORTCNTL DD statement defines the data set that contains control statements used to provide overrides or optional information for the sort application.

10 OPTION statement. The file size is specified as exactly 2270 records. Both CH and AQ format record fields are sorted as if they were AQ format.

Example 14. OVERRIDING INSTALLATION OPTIONS AND DEBUGGING

INPUT	Blocked fixed-length records on 3350.
OUTPUT	Blocked fixed-length records on SYSDA.
INTERMEDIATE STORAGE	Dynamically allocated.
USER ROUTINES	None.
OPTIONS	Unused temporary SORTOUT space is not to be released; needed work space is to be dynamically allocated; debug information is to be obtained.

```
//EXAMP     JOB   A400,PROGRAMMER                                    01
//STEP1     EXEC  PGM=SORT,PARM='MSGPRT=CRITICAL,SIZE(800K)'        02
//SYSOUT    DD    SYSOUT=A                                          03
//SORTIN    DD    DSNAME=INP1,DISP=OLD,UNIT=3350,                   04
//            DCB=(RECFM=FB,BLKSIZE=7200,LRECL=80),                 05
//            VOL=SER=XB0001                                        06
//SORTOUT   DD    DSNAME=&&OUTPUT,DISP=(,PASS),UNIT=SYSDA,          07
//            SPACE=(CYL,(5,1))                                     08
//SYSIN     DD *                                                    09
  SORT FIELDS=(1,6,A,28,5,D),FORMAT=CH                            10
  OPTION NOOUTREL,DYNALLOC                                        11
* THE NEXT STATEMENT WILL PRODUCE A SPECIALLY FORMATTED           12
* DUMP IF THE STEP ABENDS                                         13
  DEBUG FMTABEND                                                  14
/*                                                                15
//SORTDIAG DD DUMMY                                               16
```

This job is only applicable to MVS systems because of the use of dynamic allocation for work data sets. For purposes of illustration, assume that none of the standard defaults for JCL invocation of DFSORT have been changed at installation time.

Line **Explanation**

01 The JOB statement introduces this job to the operating system.

02 The EXEC statement calls the program by its alias SORT. MSGPRT=CRITICAL specifies that only error messages are to be printed, overriding the standard default of MSGPRT=ALL. Because the standard default of LIST=YES has not been overridden, control statements are also printed. SIZE (800K) specifies the main storage to be allocated to DFSORT, overriding the standard default of SIZE=MAX.

03 The SYSOUT DD statement directs messages, control statements, and the specially formatted dump to system output class A.

04-06 The SORTIN DD statement describes an input data set named INP1 on volume XB0001 of a 3350. It consists of fixed-length blocked records with a record length of 80 and a block size of 7200.

07-08 The SORTOUT DD statement describes a temporary output data set, which is passed to the next step for further processing.

09 The SYSIN DD statement indicates that a data set follows in the input stream.

10 SORT statement. Describes two control fields in the input records.

11 OPTION statement. NOOUTREL specifies that unused temporary SORTOUT space is not to be released, overriding the standard default of OUTREL=YES. DYNALLOC specifies that needed work space is to be dynamically allocated using the standard default device (SYSDA) and number of work data sets (1).

12-13 Comment statements. These statements are printed, but otherwise ignored.

14 DEBUG statement. FMTABEND specifies that the program is to produce a specially formatted dump if an abend situation occurs. The specially formatted dump is produced on the message data set and includes a SNAP dump equivalent to a SYSUDUMP and a formatted dump of the communication area.

15 Marks the end of the SYSIN data set.

16 The SORTDIAG DD statement indicates that all messages (including diagnostic messages) and control statements are to be printed, overriding MSGPRT=CRITICAL in the EXEC statement.

Note that the cumulative effect of the SORTDIAG DD statement, the options in the EXEC PARM field, and the control statements in the SYSIN data set is the following equivalent set of control statements for the run:

```
SORT FIELDS=(1,6,A,28,5,D),FORMAT=CH
OPTION NOOUTREL,DYNALLOC,MSGPRT=ALL,MAINSIZE=819200,LIST
DEBUG FMTABEND
```

Example 15. OMIT, INREC, AND SUM CONTROL STATEMENTS

INPUT	Blocked fixed-length records on 3350.
OUTPUT	Blocked fixed-length records on SYSDA.
INTERMEDIATE STORAGE	Dynamically allocated.
USER ROUTINES	None.
OPTIONS	Needed work space is to be dynamically allocated.

```
//EXAMP     JOB   A400,PROGRAMMER                                        01
//STEP1     EXEC  PGM=SORT                                              02
//SYSOUT    DD    SYSOUT=A                                              03
//SORTIN    DD    DSNAME=INP1,DISP=OLD,UNIT=3350,                       04
//            DCB=(RECFM=FB,BLKSIZE=7200,LRECL=80),                     05
//            VOL=SER=XB0001                                            06
//SORTOUT   DD    DSNAME=&&OUTPUT,DISP=(,PASS),UNIT=SYSDA,              07
//            SPACE=(CYL,(5,1)),DCB=(LRECL=25,BLKSIZE=5000)             08
//SYSIN     DD    *                                                     09
  OMIT COND=(5,1,CH,EQ,C'M')                                           10
  INREC FIELDS=(10,3,20,8,33,11,2Z,5,1)                                11
  SORT FIELDS=(4,8,CH,A,1,3,FI,A),DYNALLOC=(,2)                        12
  SUM FIELDS=(17,4,BI)                                                 13
/*
```

This example shows how to use the OMIT, INREC, SORT, and SUM control statements. This job is only applicable to MVS systems because of the use of dynamic allocation for work data sets. For purposes of illustration, assume that none of the standard defaults for JCL invocation of DFSORT have been changed at installation time.

Line	Explanation
01	The JOB statement introduces this job to the operating system.
02	The EXEC statement calls the program by its alias SORT.
03	The SYSOUT DD statement directs messages and control statements to system output class A.
04-06	The SORTIN DD statement is identical to that in Example 13. The input records have a record length of 80 and a block size of 7200.
07-08	The SORTOUT DD statement is identical to that in Example 13 except that the reformatted records have a record length of 25 and a block size of 5000.
09	The SYSIN DD statement indicates that a data set follows in the input stream.

10 OMIT statement. COND specifies that input records with a character M in position 5 are to be deleted.

11 INREC statement. FIELDS specifies how the input records are to be reformatted before they are sorted. The reformatted input records are fixed-length, with a record size of 25 bytes (a significant reduction from the original size of 80 bytes). They look as follows:

Position	Content
1-3	Input positions 10 through 12
4-11	Input positions 20 through 27
12-22	Input positions 33 through 43
23-24	Zeros
25	Input position 5

12 SORT statement. FIELDS specifies two control fields starting at positions 4 and 1 in the reformatted record, which correspond to positions 20 and 10 in the input record. DYNALLOC=(,2) specifies that needed work space is to be dynamically allocated using the standard default device (SYSDA) and 2 work data sets.

13 SUM statement. FIELDS specifies a 4-byte binary summary field at position 17 in the reformatted record, which corresponds to position 38 in the input record. Whenever two reformatted records with the same control fields are found, their summary fields are to be added and placed in one of the reformatted records, and the other reformatted record is to be deleted.

Example 16. SORT WITH COPY OPTION

INPUT	Block fixed- or variable-length records
OUTPUT	Blocked fixed- or variable-length records
INTERMEDIATE STORAGE	None
USER ROUTINES	One: E35
OPTIONS	FIELDS=COPY

```
//EXAMP   JOB  A402,PROGRAMMER                                    01
//STEP1  EXEC  PGM=SORT,PARM='MSGPRT=CRITICAL'                    02
//SYSOUT   DD  SYSOUT=A                                           03
//SORTIN   DD  DSNAME=INP1,DISP=OLD                               04
//SORTOUT  DD  DSNAME=OUT1,DISP=(NEW,KEEP),UNIT=3380,             05
//             SPACE=(TRK,(15,2)),VOL=SER=XYZ003                  06
//MODLIB   DD  DSNAME=MY.LOADLIB,DISP=SHR,                        07
//             UNIT=3380,VOL=SER=XYZ012                           08
//SYSIN    DD  *                                                  09
  OMIT COND=(1,8,CH,EQ,C'        ')                             10
  MODS E35=(ALTREC,11000,MODLIB)                                  11
  SORT FIELDS=COPY
/*
```

This example shows how a SORT/MERGE control statement can be used to specify a copy application.

Line	Explanation
01	The JOB statement introduces this job to the operating system.
02	The EXEC statement calls the program by its alias SORT. MSGPRT=CRITICAL in the EXEC parm field specifies that only error messages are to be printed.
03	The SYSOUT DD statement directs DFSORT messages to output class A.
04	The SORTIN DD statement describes a cataloged input data set INP1.
05-06	The SORTOUT DD statement directs the output to a new data set named OUT1 on volume XYZ003 of a 3380.
07-08	The MODLIB DD statement describes the exit library as a data set named MY.LOADLIB on a volume XYZ012 of a 3380.
09	The SYSIN DD statement indicates that data follows in the input stream.
10	The OMIT statement specifies that records with blanks in the first 8 bytes of the input records are to be omitted.

11 The MODS statement specifies an E35 exit named ALTREC that is 11000 bytes long and in a data set defined by ddname MODLIB.

12 The SORT statement indicates that a copy is to be done.

Example 17. SORT USING COBOL EXITS

INPUT	Fixed- or variable-length records
OUTPUT	Fixed- or variable-length records
INTERMEDIATE STORAGE	One SYSDA area of 3 cylinders
USER ROUTINES	Two: E15 and E35 written in COBOL and executed with VS COBOL II library subroutines
OPTIONS	Alternate message data set, VS COBOL II library, accept 100 records

```
//EXAMP    JOB A402,PROGRAMMER                                        01
//STEP1    EXEC PGM=SORT,REGION=1000K,PARM='MSGDDN=MYQUE'             02
//STEPLIB  DD DSN=SYS1.COB2LIB,DISP=SHR                               03
//EXITC    DD DSN=COBEXITS.LOADLIB,DISP=SHR                           04
//MYQUE    DD SYSOUT=A                                                05
//SYSOUT   DD SYSOUT=A                                                06
//SORTIN   DD DSN=SORT1.IN,DISP=SHR                                   07
//SORTOUT  DD DSN=SORT1.OUT,DISP=OLD                                  08
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,3)                                09
//SYSIN    DD *                                                       10
 OPTION MAINSIZE=MAX,RESALL=70K,                                     11
        STOPAFT=100,COBEXIT=COB2                                     12
 SORT FIELDS=(5,4,BI,A),EQUALS                                       13
 MODS E15=(COBOLE15,37000,EXITC,C),                                  14
      E35=(COBOLE35,37000,EXITC,C)                                   15
/*
```

Line	Explanation
01-02	The EXEC statement calls the program by its alias SORT, and indicates that a 1000K-byte region is needed for program execution. The PARM statement specifies that the DFSORT messages are written to a data set defined by a DD statement called MYQUE.
03	The STEPLIB DD statements indicate where the VS COBOL II library is located.
04	The EXITC DD statement defines the library in which the exit routines are located.
05	The MYQUE DD statement specifies the alternate message data set (to keep DFSORT messages separate from COBOL messages) for DFSORT messages and control statements.
06	The SYSOUT DD statement specifies the message data set for COBOL messages.
07	The SORTIN DD statement specifies the input data set SORT1.IN.
08	The SORTOUT DD statement specifies the output data set SORT1.OUT.

09 The SORTWK01 DD statement describes the work data set on a SYSDA device of 3 cylinders.

10 The SYSIN DD statement indicates that the DFSORT control statements follow in the input stream.

11-12 The OPTION statement. MAINSIZE=MAX instructs the program to calculate the amount of main storage available and allocate the maximum amount. STOPAFT indicates that DFSORT accepts 100 records before sorting. COBEXIT specifies that E15 and E35 exit routines be executed with the VS COBOL II library.

13 SORT statement. FIELDS describes the control fields in the input records on which the program sorts.

14-15 MODS statement. E15 specifies a user exit routine written in COBOL. The name of the routine is COBOLE15, it is 7000 bytes long, and it requires 30000 bytes for the COBOL library subroutines. The exit routine resides in the data set described in the EXITC DD statement. E35 specifies a user exit routine written in COBOL. The name of the routine is COBOLE35, it is 7000 bytes long, and it requires 30000 bytes for the COBOL library subroutines. The exit routine resides in the data set described in the EXITC DD statement.

Example 18. DYNAMIC LINK-EDITING OF USER EXIT ROUTINES, PROC=SORT

INPUT	Fixed- or variable-length records.
OUTPUT	Fixed- or variable-length records.
INTERMEDIATE STORAGE	One SYSDA area of 1 cylinder.
USER ROUTINES	9 user routines; see MODS statement below.
OPTIONS	4 user routines link-edited together for the input phase; 1 user routine link-edited separately for the input phase; 4 user routines link-edited together for the output phase.

```
//EXAMP    JOB A400,PROGRAMMER                                          01
//S1       EXEC SORT                                                    02
//SORTIN   DD DSN=SMITH.INPUT,DISP=OLD                                  03
//SORTOUT  DD DSN=SMITH.OUTPUT,DISP=(NEW,CATLG),                        04
//           UNIT=3380,SPACE=(TRK,(10,2)),VOL=SER=XYZ003
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1,1))                              05
//EXIT     DD DSN=SMITH.EXIT.OBJ,DISP=SHR                               06
//EXIT2    DD DSN=SMITH.EXIT2.OBJ,DISP=SHR                              07
//SORTMODS DD UNIT=SYSDA,SPACE=(TRK,(10,,3))                            08
//SYSIN    DD *                                                         09
   SORT FIELDS=(1,8,CH,A,20,4,BI,D)                                     10
   MODS E11=(EXIT11,1024,EXIT,S),                                       11
        E15=(E15,1024,SYSIN,T),
        E17=(EXIT17,1024,EXIT2,T),
        E18=(EXIT18,1024,EXIT,T),
        E19=(E19,1024,SYSIN,T),
        E31=(PH3EXIT,1024,EXIT,T),
        E35=(PH3EXIT,1024,EXIT,T),
        E38=(PH3EXIT,1024,EXIT,T),
        E39=(E39,1024,SYSIN,T)
   END                                                                  12
<object deck for E15 exit here>                                         13
<object deck for E19 exit here>
<object deck for E39 exit here>
/*
```

Line	Explanation
01	The JOB statement introduces this job to the operating system.
02	The EXEC statement calls the program through the SORT cataloged procedure, which also sets up the four DD statements (not shown) required by the linkage editor.
03	The SORTIN DD statement describes a cataloged input data set named SMITH.INPUT.
04	The SORTOUT DD statement directs the output to a new data set named SMITH.OUTPUT on volume XYZ003 of a 3380.

05 The SORTWK01 DD statement specifies that SYSDA space of 1 primary cylinder and secondary extents of 1 cylinder each can be used for work space by DFSORT.

06 The EXIT DD statement specifies the partitioned data set containing the object decks for the E11, E18, E31, E35, and E38 exit routines.

07 The EXIT2 DD statement specifies the partitioned data set containing the object deck for the E17 exit routine.

08 The SORTMODS DD statement defines a partitioned data set to hold user exit routine object decks from SYSIN for input to the linkage editor. SYSDA space of 10 primary tracks and 3 directory blocks is reserved.

09 The SYSIN DD statement indicates that data follows in the input stream.

10 The SORT statement defines two control fields in the input records.

11 The MODS statement specifies that:

- The EXIT11 routine in the EXIT library is to be link-edited separately from other input phase exit routines and associated with exit E11;
- The E15 and E19 routines in SYSIN, the EXIT17 routine in EXIT2, and the EXIT18 routine in EXIT are to be link-edited together and associated with exits E15, E19, E17, and E18, respectively;
- The E31, E35, and E38 routines in the PH3EXIT object deck and the E39 routine in SYSIN are to be link-edited together and associated with exits E31, E35, E38, and E39, respectively.

12 The END statement marks the end of the DFSORT control statements and the beginning of the exit routine object decks.

13 The three object decks for E15, E19, and E39 exit routines follow the END statement.

Example 19. EXTENDED PARAMETER LIST INTERFACE

INPUT	Fixed-length records from a preloaded E15 exit routine
OUTPUT	Blocked fixed-length records
INTERMEDIATE STORAGE	One 3380 area
USER ROUTINES	None
OPTIONS	The order of identically collating records must be preserved; estimated file size; automatic secondary allocation should not be used.

```
 //EXAMP    JOB   A400,PROGRAMMER                                    01
 //STEP1    EXEC   PGM=MYSORT                                        02
 //MSGOUT   DD     SYSOUT=A                                          03
 //STEPLIB  DD     DSNAME=NAME1.NAME2.NAME3,DISP=SHR                 04
 //SORTOUT  DD     DSNAME=&&OUTPUT,DISP=(,PASS),UNIT=SYSDA,          05
 //           SPACE=(CYL,(8,4)),DCB=(RECFM=F,LRECL=111)
 //SORTWK01 DD     SPACE=(CYL,(10)),UNIT=3380                        06
 //SORTCNTL DD     *                                                 07
   OPTION EQUALS,FILSZ=E30000,NOWRKSEC                               08
   INCLUDE COND=(5,8,GT,13,8),FORMAT=FI                              09
   OUTREC FIELDS=(5X,5,8,5X,13,8,5X,1,80)                            10
   RECORD TYPE=F,LENGTH=80                                           11
 /*                                                                  12
 //SYSOUT DD SYSOUT=A                                                13
MYSORT CSECT                                                        14
       .
       .
       .
       LA   R1,PL1            SET ADDRESS OF PARAMETER LIST        15
*                              TO BE PASSED TO SORT/MERGE
       ST   R2,PL4            SET ADDRESS OF GETMAINED AREA
*                              TO BE PASSED TO E15
       LINK EP=SORT           INVOKE SORT/MERGE
       .
       .
       .
PL1    DC   A(CTLST)          ADDRESS OF CONTROL STATEMENTS        16
PL2    DC   A(E15)            ADDRESS OF E15 ROUTINE
PL3    DC   A(0)              NO E35 ROUTINE
PL4    DS   A                 USER EXIT ADDRESS CONSTANT
PL5    DC   F'-1'             INDICATE END OF LIST
CTLST  DS   0H                CONTROL STATEMENTS AREA              17
       DC   AL2(CTL2-CTL1)    LENGTH OF CHARACTER STRING
CTL1   DC   C' SORT FIELDS=(4,5,CH,A)'                              18
       DC   C' OPTION '                                             19
       DC   C'RESINV=2048,FILSZ=E25000,MSGDDN=MSGOUT'
       DC   C' OMIT COND=(5,8,EQ,13,8),FORMAT=FI '                  20
CTL2   EQU  *
OUT    DCB  DDNAME=SYSOUT,... MYSORT USES SYSOUT
E15    DS   0H                E15 ROUTINE                          21
       .
       .
       .
       BR   R14               RETURN TO SORT/MERGE
       .
       .
```

Example 17 shows the use of the extended parameter list. The JCL for program MYSORT and highlights of the code for program MYSORT are shown in the two boxes, respectively. This job is applicable to all systems supported by DFSORT in all addressing modes. For purposes of illustration, assume that none of the standard defaults for dynamic invocation of DFSORT have been changed at installation time.

Line	Explanation
01	The JOB statement introduces this job to the operating system.
02	The EXEC statement specifies the name of the program calling DFSORT.
03	The MSGOUT DD statement directs the DFSORT messages to output class A. (The SYSOUT DD statement cannot be used for sort messages because it is being used by MYSORT.)
04	The STEPLIB DD statement describes where MYSORT is located.
05	The SORTOUT DD statement describes a temporary data set on SYSDA with fixed-length records and a logical record length of 111 bytes.
06	The SORTWK01 DD statement describes a temporary work data set on a 3380 containing 10 cylinders.
07	The SORTCNTL DD statement indicates that a data set follows in the input stream.
08	OPTION statement. EQUALS specifies that the order of records with equal control fields is to be preserved, overriding the standard default of EQUALS=NO. FILSZ=E30000 specifies the estimated number of records to be sorted, overriding FILSZ=E25000 in the OPTION statement of the invocation parameter list. NOWRKSEC specifies that no automatic secondary allocation is to take place for the temporary work data set, overriding the standard default of WRKSEC=YES.
09	INCLUDE statement. COND and FORMAT specify that input records in which the fixed-integer number in positions 5 to 12 is greater than the fixed-integer number in positions 13 to 20 are the only input records that are included in the output data set. The INCLUDE statement causes the OMIT statement of the invocation parameter list to be ignored.
10	OUTREC statement. FIELDS specifies how the input records are to be reformatted before they are output. The output records are fixed length, with a record size of 111 bytes. They look as follows:

Position	Content
1-5	Blanks
6-13	Input positions 5 through 12
14-18	Blanks
19-26	Input positions 13 through 20
27-31	Blanks
32-111	Input positions 1 through 80

11 The RECORD statement. Indicates that the input records are fixed-length and 80 bytes long.

12 Marks the end of the SORTCNTL data set.

13 The SYSOUT DD statement is used by MYSORT and thus cannot be used by DFSORT.

14 This is the start of the MYSORT program. Assume that it GETMAINs a work area, saves its address in register 2, and initializes the work area for use by its E15 routine.

15 Before calling DFSORT, MYSORT places the address of the parameter list to be passed to DFSORT in register 1, places the address of the GETMAINed work area in the user exit address constant field in the parameter list, and indicates that the user exit address constant field is the last field in the parameter list (that is, there is no ALTSEQ table or STAE routine). Then MYSORT calls DFSORT.

16 The parameter list which MYSORT passes to DFSORT contains the address of the control statements area, the address of the E15 routine, and the address of the GETMAINed work area. It indicates there is no E35 routine.

17 The control statements area contains the length of the control statements character string, followed by the character string which contains a SORT statement and an OPTION statement.

18 SORT statement. FIELDS specifies a control field in the input records.

19 OPTION statement. RESINV=2048 specifies the number of bytes to be reserved for the invoking program (MYSORT), overriding the standard default of RESINV=0. FILSZ=E25000 specifies the estimated number of records to be sorted. (There is no standard default for FILSZ.) MSGDDN=MSGOUT specifies the ddname to be used for program messages, overriding the standard default of MSGDDN=SYSOUT. Note that RESINV=2048 and MSGDDN=MSGOUT cannot be overridden by corresponding options specified in the SORTCNTL data set; these options are ignored when specified in the SORTCNTL data set because it is too late to use them when they are read.

20 OMIT statement. COND and FORMAT specify that input records in which the fixed-integer number in positions 5 to 12 is equal to the fixed-integer number in positions 13 to 20 are to be deleted.

21 The E15 routine is preloaded; therefore, it generates input records and passes them to DFSORT. For this reason, a SORTIN DD statement is not used.

Note that the cumulative effect of the control statements in the SORTCNTL data set and the control statements in the invocation parameter list is the following equivalent set of control statements for the run:

```
SORT FIELDS=(4,5,CH,A)
OPTION EQUALS,FILSZ=E30000,NOWRKSEC,RESINV=2048,MSGDDN=MSGOUT
INCLUDE COND=(5,8,GT,13,8),FORMAT=FI
OUTREC FIELDS=(5X,5,8,5X,13,8,5X,1,80)
```

Merge Examples

Example 20. MERGE FOUR DATA SETS, PROC=SORTD

INPUT	Four data sets containing blocked fixed-length records; data sets are on four devices: two 3350s and two 3380s.
OUTPUT	One data set containing 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

```
//EXAMP     JOB   A402,PROGRAMMER
//STEP1     EXEC  SORTD                                                  01
//SORTIN01  DD    DSNAME=MERGIN01,VOL=SER=SCR760,DISP=OLD,               02
//            UNIT=3380,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6000)             03
//SORTIN02  DD    DSNAME=MERGIN02,VOL=SER=SYS004,DISP=OLD,               04
//            UNIT=3350,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6000)             05
//SORTIN03  DD    DSNAME=MERGIN03,VOL=SER=SCR764,DISP=OLD,               06
//            UNIT=3380,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6000)             07
//SORTIN04  DD    DSNAME=MERGIN04,VOL=SER=SYS005,DISP=OLD,               08
//            UNIT=3350,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6000)             09
//SORTOUT   DD    DSNAME=MERGOUT,                                        10
//            VOL=SER=(,RETAIN,SER=(000101)),DISP=(NEW,KEEP),          11
//            LABEL=(,NL),UNIT=3400-3                                  12
//SYSIN     DD    *                                                      13
   MERGE    FIELDS=(1,6,A,28,5,D),FORMAT=CH                            14
   OPTION   FILSZ=E10000                                               15
/*
```

Line	Explanation
01	The EXEC statement invokes the cataloged procedure SORTD.
02-09	The SORTINnn DD statements describe the merge input data sets. They are all on different devices and consist of fixed-length records with a blocking factor of 75. Because they all have the same block size, the order in which they are specified is unimportant. Had they been different, the data set with the largest block size would have had to be specified first.
10-12	By default 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).
13	A data set follows in the input stream.

14 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.

15 OPTION statement. The input data sets contain a total of approximately 10000 records.

Example 21. MERGE TWO 3350 FILES; PROC=SORTD, EXITS

INPUT	Variable-length blocked records on 3350.
OUTPUT	Variable-length blocked records on 3350.
INTERMEDIATE STORAGE	None.
USER ROUTINES	E35 (CALC) routine shortens records.
OPTIONS	Exact input data set size.

```
//EXAMP    JOB   A402,PROGRAMMER
//STEPONE  EXEC  SORTD                                              01
//SORTIN01 DD    DSNAME=WEEKLY,VOL=SER=000101,UNIT=3350,            02
//           DISP=OLD,DCB=(RECFM=VB,LRECL=240,                     03
//           BLKSIZE=4800)                                         04
//SORTIN02 DD    DSNAME=DAILY,VOL=SER=000113,UNIT=3350,            05
//           DISP=(OLD,DELETE),DCB=(RECFM=VB,LRECL=240,            06
//           BLKSIZE=1200)                                         07
//SORTOUT  DD    DSNAME=WEEKA,VOL=SER=000111,UNIT=3350,            08
//           DISP=(NEW,KEEP),SPACE=(TRK,(200,10)),                 09
//           DCB=(RECFM=VB,LRECL=200,BLKSIZE=2000)                 10
//USERLIB  DD    DSNAME=MYMODS,DISP=SHR                            11
//         DD    DSNAME=XYZ,DISP=SHR                               12
//SYSIN    DD    *                                                 13
   MERGE  FIELDS=(5,6,CH,A)                                        14
   OPTION FILSZ=8150                                               15
   RECORD TYPE=V,LENGTH=(,,200)                                    16
   MODS   E35=(CALC,800,USERLIB)                                   17
/*
```

Line	Explanation
02	Calls the SORTD cataloged procedure.
02-04	The first of two input data sets for the merge. The data set, named WEEKLY, is on a 3350 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 3350 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 is a data set named WEEKA. It is new and will be retained in the system on a 3350 disk with the serial number 000111. The data set is recorded on 200 tracks. If this space is not sufficient, additional space is allotted in blocks of 10 tracks. The data set consists of variable-length blocked records with a maximum record length of 200 (see L3 on the RECORD statement) and a block size of 2000.

11-12 The libraries on which the user exit routines reside. Because CALC resides in MYMODS, and MODRTN resides in XYZ, these data sets must be concatenated.

13 A data set follows in the input stream.

14 MERGE statement. The FIELDS operand describes one control field. The start of the control field is given as byte 5; note that this points to the first byte of the record data itself, because, for a variable-length record, the first four bytes are occupied by the record descriptor word. The field is six bytes long.

15 OPTION statement. The input data set contains exactly 8150 records.

16 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.

17 MODS statement. A routine named CALC receives control at exit E35. It is approximately 800 bytes long and resides in MYMODS.

Example 22. MERGE WITH EQUALS AND SUM

INPUT	Fixed- or variable-length records.
OUTPUT	Fixed- or variable-length records.
INTERMEDIATE STORAGE	None
USER ROUTINES	None
OPTIONS	EQUALS (on MERGE control statement), and SUM

```
//EXAMP    JOB A400,PROGRAMMER                                          01
//S1 EXEC  PGM=SORT                                                     02
//SYSOUT   DD  SYSOUT=A                                                 03
//SORTIN01 DD  DSNAME=M1234.INPUT1,DISP=SHR                             04
//SORTIN02 DD  DSNAME=M1234.INPUT2,DISP=SHR                             05
//SORTIN03 DD  DSNAME=M1234.INPUT3,DISP=SHR                             06
//SORTOUT  DD  DSNAME=M1234.MERGOUT,DISP=(NEW,KEEP),                    07
//             UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD  *                                                        08
  MERGE FIELDS=(1,8,CH,A,20,4,FI,A),EQUALS                              09
  SUM   FIELDS=(32,4,FI)                                                10
/*
```

This example shows how MERGE with SUM and EQUALS is specified.

Line **Explanation**

01 The JOB statement introduces this job to the operating system.

02 The EXEC statement calls the program by its alias SORT.

03 The SYSOUT DD statement directs DFSORT messages to output class A.

04 The SORTIN01 DD statement describes a cataloged input data set named M1234.INPUT1.

05 The SORTIN02 DD statement describes a second cataloged input data set named M1234.INPUT2.

06 The SORTIN03 DD statement describes a third cataloged input data set named M1234.INPUT3.

07 The SORTOUT DD statement directs the output to a new data set named M1234.MERGOUT. The data set is written to a SYSDA device.

08 The SYSIN DD statement indicates that a data set follows in the input stream.

09 The 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 merged into ascending order. The second field begins on byte 20, is 4 bytes long, contains fixed integer data, and is

to be merged into ascending order. The optional EQUALS operand specifies that the order of output records with equal control fields is the same as the original order within a file and is based on the file number for records from different files.

10 The SUM statement specifies that whenever two records with equal control fields are found, the contents of their 4-byte fixed integer fields beginning at position 32 are added and the sum saved in one of the records. The other record is to be deleted. Because of the EQUALS parameter on the MERGE statement, the first record always receives the sum and is kept.

Sort Examples Using VSAM Data Sets

Example 23. DEFINE VSAM DATA SETS FOR DFSORT PROCESSING

```
//EXAMP     JOB   A402,'JOHN DOE'                                  01
//DEFINE EXEC     PGM=IDCAMS,REGION=512K                           02
//SYSPRINT DD     SYSOUT=A                                         03
//DISK DD         UNIT=3330V,VOL=SER=VSM999,DISP=OLD               04
//SYSIN    DD*                                                     05
 DEFINE CL ( NAME(TEST.SORTIN.FILE) -                             06
             KEYS(4 7) -                                          07
             VOL(VSM999) -                                        08
             TRK (40 20) -                                        09
             RECSZ(91 110) -                                      10
             BUFSP(5000) ) -                                      11
 DEFINE CL ( NAME(TEST.SORTOUT.FILE) -                            12
             KEYS(4 25) -                                         13
             VOL(VSM999) -                                        14
             TRK (40 20) -                                        15
             RECSZ(91 110) -                                      16
             BUFSP(5000))                                         17
/*
```

Line	Explanation
01	The JOB statement introduces this job to the operating system.
02	The EXEC statement directs the system to execute the IDCAMS utility.
03	The SYSPRINT statement identifies the output device for messages.
04	The DISK statement ensures that the required volume is mounted.
05	The SYSIN DD statement indicates that a data set follows in the input stream.
06-11	The DEFINE CL statement defines a key-sequenced cluster named TEST.SORTIN.FILE. Its parameters are: • KEYS, which specifies that the length of the key is 4 bytes and that the key field begins in the 8th byte (offset 7) of each data record. • VOL, which specifies that the cluster is to reside on volume VSM999. • TRK, which specifies that 40 tracks are allocated for the cluster's space. When the cluster is extended, it is to be extended in increments of 20 tracks. • RECSZ, which specifies that the records are variable-length with an average size of 91 bytes and a maximum size of 110 bytes. • BUFSP, which specifies that a minimum of 5000 must be provided for I/O buffers.

12-17 The second DEFINE CL statement defines a similar cluster TEST.SORTOUT.FILE as having keys 4 bytes long beginning in position 26 (offset 25).

Example 24. SORT WITH VSAM INPUT AND OUTPUT

INPUT	VSAM variable-length records
OUTPUT	VSAM variable-length records
INTERMEDIATE STORAGE	Three SYSDA areas of five cylinders each
USER ROUTINES	None
OPTIONS	None

```
//SORT1 EXEC PGM=SORT,REGION=512K                              01
//SYSOUT   DD     SYSOUT=A                                     02
//SORTIN   DD     DSN=TEST.SORTIN.FILE,DISP=SHR                03
//SORTWK01 DD     UNIT=SYSDA,SPACE=(CYL,5)                     04
//SORTWK02 DD     UNIT=SYSDA,SPACE=(CYL,5)                     05
//SORTWK03 DD     UNIT=SYSDA,SPACE=(CYL,5)                     06
//SORTOUT  DD     DSN=TEST.SORTOUT.FILE                        07
//SYSIN    DD     *                                            08
  SORT FIELDS=(30,4,BI,A)                                     09
  RECORD TYPE=V,LENGTH=110                                    10
/*
```

Line	Explanation
01	The EXEC statement calls the program using the program name.
02	The SYSOUT statement identifies the output device for messages.
03	The SORTIN DD statement describes as input the VLR VSAM data set, TEST.SORTIN.FILE, previously created by DFSORT.
04-06	The SORTWK DD statements specify that SYSDA space of 5 cylinders each can be used for work space by DFSORT.
07	The SORTOUT DD statement directs output to a VLR VSAM data set named TEST.SORTOUT.FILE.
08	The SYSIN DD statement indicates that data follows in the input stream.
09	The SORT statement defines one control field in the input records that begins in position 30 and is 4 bytes long. Since these are variable-length records, 4 bytes have been added for the record descriptor word (RDW) which DFSORT supplies at input and removes at output for VSAM records. (SORTOUT file key at byte position 26 + 4 = 30.) Note that in order to produce the output data set in sequence on positions 26 (offset 25) through 29 as defined in the cluster definition shown in Example 23, positions 30 through 33 must be specified in the FIELDS entry.

10 The RECORD statement indicates that the input file is of VLR format and the input record length is 110 bytes.

Example 25. NON-VSAM SORTIN DATA SET, VSAM SORTOUT

INPUT	Variable-length records
OUTPUT	VSAM variable-length records
INTERMEDIATE STORAGE	Three SYSDA areas of five cylinders each
USER ROUTINES	E39
OPTIONS	None

```
//SORT2 EXEC PGM=SORT,REGION=512K                                  01
//SYSOUT   DD     SYSOUT=A                                         02
//EXITC    DD     DSN=TEMPE39,DISP=SHR                             03
//SORTIN   DD     DSN=SORTINPT,DISP=SHR                            04
//SORTWK01 DD     UNIT=SYSDA,SPACE=(CYL,5)                         05
//SORTWK02 DD     UNIT=SYSDA,SPACE=(CYL,5)                         06
//SORTWK03 DD     UNIT=SYSDA,SPACE=(CYL,5)                         07
//SORTOUT  DD     DSN=TEST.SORTOUT.FILE,DISP=SHR                   08
//SYSIN    DD     *                                                09
  SORT FIELDS=(30,4,BI,A)                                         10
  RECORD TYPE=V                                                   11
  MODS E39=(E39,7000,EXITC)                                       12
/*
```

Line	Explanation
01	The EXEC statement calls the program using the program name.
02	The SYSOUT statement identifies the output device for messages.
03	The EXITC DD statement defines the library in which the exit routines are located.
04	The SORTIN DD statement describes a sequential input data set named SORTINPT.
05-07	The SORTWK DD statements specify that SYSDA space of 5 cylinders each can be used for work space by DFSORT.
08	The SORTOUT DD statement directs output to a VLR VSAM data set named TEST.SORTOUT.FILE.
09	The SYSIN DD statement indicates that data follows in the input stream.
10	The SORT statement defines one control field in the input records that begins in position 30 and is 4 bytes long.
11	The RECORD statement indicates that the input file is VLR format.

12 The MODS statement describes a user routine that will receive control at program exit E39. The name of the routine is E39; it is 7000 bytes long and is on the data set defined in the EXITC DD statement.

Appendix B. Calculating Storage Requirements

This appendix describes the guidelines for allocating main storage to DFSORT.

Storage devices used for intermediate storage, the factors determining the amount of intermediate storage required for a DFSORT program, and the program's method of selecting a sorting technique are also discussed.

Main Storage

The amount of main storage to be made available to DFSORT is defined when the program is installed. If for any reason this default value is unsuitable, you can override it with the MAINSIZE/SIZE parameter (for ways to specify this value, see Appendix D).

In general, the more (virtual) main storage you make available to the program (up to a certain limit), the better the performance. The effect of the main storage amount on performance is discussed under "Tuning Main Storage" on page 207.

Intermediate Storage

Most sorting applications need work space on disk or tape. Merge and copy 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. You can assign intermediate storage on either mixed direct access devices or magnetic tape, but not both.

Direct Access

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
- All system supported models of the 3880 are supported for use with 3380.

Notes:

1. *The 3880 Model 1 with the Speed Matching Buffer Feature permits attachment of the 3375 to systems with block multiplexer channels with data rates less than 3 megabytes per second.*

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

3. *The 3880 Model 23 with cache provides high-speed access to data in the cache memory located in the control unit.*

For MVS, system performance is improved if storage is specified in cylinders rather than tracks. Storage on temporary SORTWKnn data sets is reallocated in cylinders. The number of tracks per cylinder for direct access devices is shown in Figure 37.

Device	Tracks per Cylinder	Maximum Bytes used per Track
2314	20	7193
3330/3333 series	19	13030
3340	12	8368
3350	30	19059
3375	12	35616
3380	15	47476

Figure 37. Number of Tracks per Cylinder for Direct Access Devices

The program allocates secondary extents as required, even if not requested in the JCL, if DFSORT has been installed with the option WRKSEC=YES, unless the data set is virtual I/O.

For allocation of sort work data sets it is usually adequate to allocate twice the space used by the input data set(s). Certain conditions may cause additional space requirements. These include:

- Use of long control words (>150 bytes).
- Using different device types or work data sets.
- Use of an alternate collating sequence.

Tape

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 can use neither the data converter, nor the translation feature for anything but character data.

Three different techniques are available to the program: Balanced, Polyphase, and Oscillating. For information on how to calculate their requirements, see Figure 38.

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 need more intermediate work space. Space requirements are also summarized in Figure 38. DFSORT selects the most appropriate tape technique using these criteria.

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

Figure 38. External Work Storage Requirements of the Various Tape Techniques

* Number of input volumes of blocking equals work storage blocking.

Exceeding Intermediate Storage Capacity

At the beginning of a sorting operation, DFSORT estimates a maximum sorting capacity (Nmax) and generates an informative message: ICE092I or ICE093I for a disk sort, ICE038I for a tape sort. See the explanation of these messages for details.

The message gives the *approximate capacity* in number of records. With disk work space, the value is usually 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 sorting, the allocation of secondary space on one of the sort work data sets fails, the system issues a B37 informational message. DFSORT can recover from this by allocating space on one of the other work data sets, if one is available.

Work Storage on Disk

DFSORT normally allocates secondary extents for work data sets, even if not requested in the JCL. This reduces the probability of exceeding intermediate storage capacity.

Work Storage on Tape

For magnetic tape, a tape length of 2400 feet is assumed in calculating Nmax, so, for tapes of other lengths, the figure is not 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 or ICE116A - 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, DFSORT is sure it will fail. In case of doubt, the message is not issued.

Appendix C. Converting to the Extended Parameter List

Programs that use the 24-bit parameter list can be converted to use the extended parameter list by applying a combination of the parameters in the extended parameter list and the control statements in its control statements area. Figure 39 shows the correspondence between the two parameter lists.

24-Bit List	Equivalent in Extended List
SORT | MERGE, RECORD, MODS, ALTSEQ control statements	Corresponding control statements
Address of E15 or E32 routine	Address of E15 or E32 routine
Address of E35 routine	Address of E35 routine
Main storage value	MAINSIZE option of the OPTION control statement
Reserved main storage value	RESINV option of the OPTION control statement
Message ddname	MSGDDN option of the OPTION control statement
Number of input files to a merge	FILES option of the MERGE control statement
ALTSEQ translation table	ALTSEQ translation table
STAE/ESTAE information	STAE/ESTAE information
Message type option	MSGPRT option of the OPTION control statement
Option characters for ddnames	SORTDD option of the OPTION control statement
Allow short variable records	VLSHRT option of the OPTION control statement

Figure 39. Converting to the Extended Parameter List

Appendix D. Specification/Override of DFSORT Options

Listed below are the places in DFSORT where you can specify various options. The sources for the options are listed in override order; that is, any option specified in a higher place in the list overrides one specified in a lower place.

JCL Invoked DFSORT

- EXEC statement PARM field
- SYSIN data set
  - DEBUG and OPTION control statements
  - Other control statements
- Installation macro (ICEMAC JCL)
- Standard defaults

Dynamically Invoked DFSORT

- SORTCNTL data set
  - DEBUG and OPTION control statements
  - Other control statements
- Parameter list
  - DEBUG and OPTION control statements
  - Other control statements
- Installation macro (ICEMAC INV)
- Standard defaults

The following sections show the possible sources of specification and order of override for individual options.

JCL Invoked DFSORT

Figure 40 shows where each sort, merge, or copy option may be specified when DFSORT is invoked through JCL. Unless otherwise noted, *the order of override between sources of specification is from right to left*; that is, a specification overrides all specifications to its left. *The order of override within a source is from bottom to top*; that is, a specification overrides all specifications above it.

The column on the left tells which functions (S=sort, M=merge, or C=copy) can use the option. Notes follow the figure.

Function	Description of Option	Specified with ICEMAC JCL	Specified with SYSIN	Specified with EXEC PARM
S,M	Alternate sequence	ALTSEQ	ALTSEQ CODE	NO
S	System storage above 16-megabyte virtual	ARESALL	OPTION ARESALL	ARESALL
S,M,C	Force BSAM	NO	DEBUG BSAM	BSAM
S	Bypass Sorting Instructions	NO	DEBUG NOASSIST	NO
S	Placement of buffers	NO	DEBUG BUFFERS	NO
S,M	CH field sequence	CHALT	OPTION CHALT | NOCHALT	NO
S,M	Record count check	CHECK	OPTION CHECK | NOCHECK	NO
S,M,C	COBOL library	COBEXIT	OPTION COBEXIT	NO
S,M,C	ABEND record count	NO	DEBUG CTRx	NO
S,M,C	Abnormal stop	NO	DEBUG ABSTP	NO
S	Dynamic SORTWKs	DYNALLOC[1]	SORT DYNALLOC OPTION DYNALLOC	NO
S,M	Equal record order	EQUALS	SORT | MERGE EQUALS | NOEQUALS OPTION EQUALS | NOEQUALS	NO
S,M,C	Error action	ERET	DEBUG ABEND | NOABEND	NO
S	EXCPVR for SORTWK	EXCPVR	NO	NO
S,M,C	Include | Omit fields	NO	INCLUDE | OMIT COND/FORMAT	NO

Figure 40 (Part 1 of 3). JCL DFSORT Option Specification/Override

Function	Description of Option	Specified with ICEMAC JCL	Specified with SYSIN	Specified with EXEC PARM
S,M,C[8]	Exit Exx (xx=11,15-19,31,35, 37-39, and 61)	NO	MODS Exx	E15=COB E35=COB
S,M,C	Inrec fields	NO	INREC FIELDS	NO
S,M,C	Outrec fields	NO	OUTREC FIELDS	NO
S,M	Control fields	NO	SORT | MERGE FIELDS/FORMAT	NO
C	Copy records	NO	SORT | MERGE FIELDS OPTION COPY	NO
S,M	Sum fields	NO	SUM FIELDS/FORMAT	NO
M	Merge input files	NO	MERGE FILES	NO
S,M	File size	NO	SORT | MERGE FILSZ | SIZE OPTION FILSZ | SIZE	NO
S,M,C	Formatted dump	NO	DEBUG FMTABEND	NO
S,M	Checkpoints	IGNCKPT	SORT | MERGE CKPT | CHKPT[2] OPTION CKPT | CHKPT[2]	NO
S,M,C	Record lengths	NO	RECORD LENGTH	NO
S,M,C	Print control statements[3]	LIST	NO[5]	LIST | NOLIST
S,M,C	Maximum storage below 16-megabyte virtual[4]	MAXLIM	NO	NO
S,M,C	Minimum storage	MINLIM	NO	NO
S,M,C	Alternate message data set	MSGDDN	NO[7]	MSGDDN
S,M,C	Write messages on master console	MSGCON	NO	NO
S,M,C	Print messages	MSGPRT	NO[6]	MSGPRT | FLAG
S,M	Bypass Blockset	NO	OPTION NOBLKSET	NO
S,M,C	Release SORTOUT space	OUTREL	OPTION NOOUTREL	NO
S,M,C	SORTOUT secondary allocation	OUTSEC	OPTION NOOUTSEC	NO

Figure 40 (Part 2 of 3). JCL DFSORT Option Specification/Override

Function	Description of Option	Specified with ICEMAC JCL	Specified with SYSIN	Specified with EXEC PARM
S,M,C	Storage over REGION	OVERRGN	NO	NO
S,M,C	System reserved storage[4]	RESALL	OPTION RESALL	RESALL
S,M,C	Resident modules	RESDNTx	NO	NO
S,M,C	Storage	SIZE	OPTION MAINSIZE	SIZE | CORE
S,M,C	SMF records	SMF	NO	NO
S,C	Skip records	NO	SORT SKIPREC OPTION SKIPREC	NO
S,M,C	User of STIMER	STIMER	OPTION NOSTIMER	NO
S,C	Input limit	NO	OPTION STOPAFT	NO
S,M,C	Record format	NO	RECORD TYPE	NO
S,M,C	User SVC number	SVC	NO	NO
S	Maximum storage above and below 16-megabyte virtual	TMAXLIM	NO	NO
S,M	Sequence check	VERIFY	OPTION VERIFY | NOVERIFY	NO
S	SORTWK virtual I/O	VIO	NO	NO
S,M	Variable records do not contain all specified control fields	VLSHRT	OPTION VLSHRT | NOVLSHRT	NO
S	Release SORTWK space	WRKREL	OPTION NOWRKREL	NO
S	SORTWK secondary allocation	WRKSEC	OPTION NOWRKSEC	NO

Figure 40 (Part 3 of 3). JCL DFSORT Option Specification/Override

Notes to Figure 40:

1. Does not request dynamic allocation; just supplies defaults.
2. Not used if Blockset is selected and IGNCKPT=YES was specified.
3. Not used if MSGPRT=NONE is in effect; in this case control statements are not printed.
4. Not used unless MAINSIZE=MAX is in effect.
5. OPTION LIST | NOLIST in SYSIN is not used.
6. OPTION MSGPRT in SYSIN is not used.
7. OPTION MSGDDN in SYSIN is not used.
8. All functions do not apply to all exits. See Figure 16 on page 138 and Figure 17 on page 139 for applicable exits.

Dynamically Invoked DFSORT with an Extended Parameter List

Figure 41 shows where each sort, merge, or copy option may be specified when DFSORT is dynamically invoked and an extended parameter list is passed to it. Unless otherwise noted, *the order of override between sources of specification is from right to left*; that is, a specification overrides all specifications to its left. *The order of override within a source is from bottom to top*; that is, a specification overrides all specifications above it. Note that control statements other than DEBUG and OPTION specified in the SORTCNTL data set completely override corresponding control statements specified through the extended parameter list. (SORT and MERGE and INCLUDE and OMIT, are considered to be corresponding control statements.)

The column on the left tells which functions (S=sort, M=merge, or C=copy) can use the option. Notes follow the figure.

Function	Description of Option	Specified with ICEMAC INV	Specified with Extended Parameter List	Specified with SORTCNTL
S,M	Alternate sequence	ALTSEQ	ALTSEQ CODE Offset 16 entry	ALTSEQ CODE
S	System storage above 16-megabyte virtual	ARESALL	OPTION ARESALL	OPTION ARESALL
S	Storage above 16-megabyte virtual for invoking program	ARESINV	OPTION ARESINV	OPTION ARESINV
S,M,C	Force BSAM	NO	DEBUG BSAM	DEBUG BSAM
S	Bypass Sorting Instructions	NO	DEBUG NOASSIST	DEBUG NOASSIST
S	Placement of buffers	NO	DEBUG BUFFERS	DEBUG BUFFERS
S,M	CH field sequence	CHALT	OPTION CHALT | NOCHALT	OPTION CHALT | NOCHALT
S,M	Record count check	CHECK	OPTION CHECK | NOCHECK	OPTION CHECK | NOCHECK
S,M,C	COBOL library	COBEXIT	OPTION COBEXIT	OPTION COBEXIT
S,M,C	ABEND record count	NO	DEBUG CTRx	DEBUG CTRx
S,M,C	Abnormal stop	NO	DEBUG ABSTP	DEBUG ABSTP

Figure 41 (Part 1 of 4). Extended Parameter List DFSORT Option Specification/Override

Function	Description of Option	Specified with ICEMAC INV	Specified with Extended Parameter List	Specified with SORTCNTL
S	Dynamic SORTWKs	DYNALLOC[1]	SORT DYNALLOC OPTION DYNALLOC	SORT DYNALLOC[2] OPTION DYNALLOC
S,M	Equal record order	EQUALS	SORT | MERGE EQUALS | NOEQUALS OPTION EQUALS | NOEQUALS	SORT | MERGE EQUALS | NOEQUALS[2] OPTION EQUALS | NOEQUALS
S,M,C	Error action	ERET	DEBUG ABEND | NOABEND	DEBUG ABEND | NOABEND
S	EXCPVR for SORTWK	EXCPVR	NO	NO
S,M,C	Include | Omit fields	NO	INCLUDE | OMIT COND/FORMAT	INCLUDE | OMIT COND/FORMAT
S,C	Exit E15	NO	MODS E15[3] Offset 4 entry[3]	MODS E15[3]
S	Exit E18	NO	MODS E18[3] Offset 24 entry[3]	MODS E18[3]
M	Exit E32	NO	Offset 4 entry	NO
S,M,C	Exit E35	NO	MODS E35[3] Offset 8 entry[3]	MODS E35[3]
S,M,C	Exit E39	NO	MODS E39[3] Offset 28 entry[3]	MODS E39[3]
S,M,C[15]	Exit Exx (xx=11,16,17,19,31, 37,38, and 61)	NO	MODS Exx	MODS Exx
S,M,C	Inrec fields	NO	INREC FIELDS	INREC FIELDS
S,M,C	Outrec fields	NO	OUTREC FIELDS	OUTREC FIELDS
S,M	Control fields	NO	SORT | MERGE FIELDS/FORMAT	SORT | MERGE FIELDS/FORMAT
C	Copy records	NO	SORT | MERGE FIELDS OPTION COPY	SORT | MERGE FIELDS[2] OPTION COPY
S,M	Sum fields	NO	SUM FIELDS/FORMAT	SUM FIELDS/FORMAT
M	Merge input files	NO	MERGE FILES	MERGE FILES

Figure 41 (Part 2 of 4). Extended Parameter List DFSORT Option Specification/Override

Function	Description of Option	Specified with ICEMAC INV	Specified with Extended Parameter List	Specified with SORTCNTL
S,M	File size	NO	SORT | MERGE FILSZ | SIZE OPTION FILSZ | SIZE	SORT | MERGE FILSZ | SIZE[2] OPTION FILSZ | SIZE
S,M,C	Formatted dump	NO	DEBUG FMTABEND	DEBUG FMTABEND
S,M	Checkpoints	IGNCKPT	SORT | MERGE CKPT | CHKPT[4] OPTION CKPT | CHKPT[4]	SORT | MERGE CKPT | CHKPT[2,4] OPTION CKPT | CHKPT[4]
S,M,C	Record lengths	NO	RECORD LENGTH	RECORD LENGTH
S,M,C	Print control statements[5]	LIST	OPTION LIST | NOLIST	NO[6]
S,M,C	Maximum storage below 16-megabyte virtual[7]	MAXLIM	NO	NO
S,M,C	Minimum storage	MINLIM	NO	NO
S,M,C	Alternate message ddname	MSGDDN	OPTION MSGDDN	NO[8]
S,M,C	Write messages on master console	MSGCON	NO	NO
S,M,C	Print messages	MSGPRT	OPTION MSGPRT	NO[9]
S,M	Bypass Blockset	NO	OPTION NOBLKSET	OPTION NOBLKSET
S,M,C	Release SORTOUT space	OUTREL	OPTION NOOUTREL	OPTION NOOUTREL
S,M,C	SORTOUT secondary allocation	OUTSEC	OPTION NOOUTSEC	OPTION NOOUTSEC
S,M,C	Storage over REGION	OVERRGN	NO	NO
S,M,C	System reserved storage[7]	RESALL	OPTION RESALL	OPTION RESALL
S,M,C	Resident modules	RESDNTx	NO	NO

Figure 41 (Part 3 of 4). Extended Parameter List DFSORT Option Specification/Override

Function	Description of Option	Specified with ICEMAC INV	Specified with Extended Parameter List	Specified with SORTCNTL
S,M,C	Program reserved storage[7]	RESINV	OPTION RESINV	OPTION RESINV
S,M,C	Storage	SIZE	OPTION MAINSIZE	OPTION MAINSIZE
S,M,C	SMF records	SMF	NO	NO
S,C	Skip records	NO	SORT SKIPREC OPTION SKIPREC	SORT SKIPREC[2] OPTION SKIPREC
S,M,C	ddname prefix	NO	OPTION SORTDD	NO[10]
S,C	Alternate input ddname	NO	OPTION SORTIN[11]	NO[12]
S,M,C	Alternate output ddname	NO	OPTION SORTOUT[13]	NO[14]
S,M,C	Use of STIMER	STIMER	OPTION NOSTIMER	OPTION NOSTIMER
S,C	Input limit	NO	OPTION STOPAFT	OPTION STOPAFT
S,M,C	User SVC number	SVC	NO	NO
S,M,C	Record format	NO	RECORD TYPE	RECORD TYPE
S	Maximum storage above and below 16-megabyte virtual	TMAXLIM	NO	NO
S,M	Sequence check	VERIFY	OPTION VERIFY | NOVERIFY	OPTION VERIFY | NOVERIFY
S	SORTWK virtual I/O	VIO	NO	NO
S,M	Variable records do not contain all specified control fields	VLSHRT	OPTION VLSHRT | NOVLSHRT	OPTION VLSHRT | NOVLSHRT
S	Release SORTWK space	WRKREL	OPTION NOWRKREL	OPTION NOWRKREL
S	SORTWK secondary allocation	WRKSEC	OPTION NOWRKSEC	OPTION NOWRKSEC

Figure 41 (Part 4 of 4). Extended Parameter List DFSORT Option Specification/Override

Notes to Figure 41:

1. Does not request dynamic allocation; only supplies defaults.
2. Does not override corresponding option in an OPTION statement specified via the extended parameter list.
3. DFSORT terminates if the exit is specified via the parameter list entry and the exit is specified in a MODS statement.
4. Not used if Blockset is selected and IGNCKPT=YES was specified.

5 Not used if MSGPRT=NONE is in effect; in this case control statements are not printed.
6 OPTION LIST | NOLIST in SORTCNTL is not used.
7 Not used unless MAINSIZE=MAX is in effect.
8 OPTION MSGDDN in SORTCNTL is not used.
9 OPTION MSGPRT in SORTCNTL is not used.
10 OPTION SORTDD in SORTCNTL is not used.
11 Overrides SORTDD for the sort input DDname.
12 OPTION SORTIN in SORTCNTL is not used.
13 Overrides SORTDD for the sort output DDname.
14 OPTION SORTOUT in SORTCNTL is not used.
15 All functions do not apply to all exits. See Figure 16 on page 138 and Figure 17 on page 139 for applicable exits.

Dynamically Invoked DFSORT with 24-Bit List

Figure 42 shows where each sort, merge, or copy option may be specified when DFSORT is dynamically invoked and a 24-bit parameter list is passed to it. Unless otherwise noted, *the order of override between sources of specification is from right to left*: a specification overrides all specifications to its left. *The order of override within a source is from bottom to top*: a specification overrides all specifications above it. Note that control statements other than DEBUG specified in the SORTCNTL data set completely override corresponding control statements specified via the 24-bit parameter list. (SORT and MERGE and INCLUDE and OMIT are considered to be corresponding control statements.)

The column on the left tells which functions (S=sort, M=merge, or C=copy) can use the option. Notes follow the figure.

Function	Description of Option	Specified with ICEMAC INV	Specified with 24-Bit List	Specified with SORTCNTL
S,M	Alternate sequence	ALTSEQ	ALTSEQ CODE X'F6' entry	ALTSEQ CODE
S	System storage above 16-megabyte virtual	ARESALL	NO	OPTION ARESALL
S	Storage above 16-megabyte virtual for invoking program	ARESINV	NO	OPTION ARESINV
S,M,C	Force BSAM	NO	DEBUG BSAM	DEBUG BSAM
S	Bypass Sorting Instructions	NO	DEBUG NOASSIST	DEBUG NOASSIST
S	Placement of buffers	NO	DEBUG BUFFERS	DEBUG BUFFERS
S,M	CH field sequence	CHALT	NO	OPTION CHALT | NOCHALT
S,M	Record count check	CHECK	NO	OPTION CHECK | NOCHECK
S,M,C	COBOL library	COBEXIT	NO	OPTION COBEXIT
S,M,C	ABEND record count	NO	DEBUG CTRx	DEBUG CTRx
S,M,C	Abnormal stop	NO	DEBUG ABSTP	DEBUG ABSTP
S	Dynamic SORTWKS	DYNALLOC[1]	SORT DYNALLOC	SORT DYNALLOC OPTION DYNALLOC

Figure 42 (Part 1 of 4). 24-Bit List DFSORT Option Specification/Override

Function	Description of Option	Specified with ICEMAC INV	Specified with 24-Bit List	Specified with SORTCNTL
S,M	Equal record order	EQUALS	SORT | MERGE EQUALS | NOEQUALS	SORT | MERGE EQUALS | NOEQUALS OPTION EQUALS | NOEQUALS
S,M,C	Error action	ERET	DEBUG ABEND | NOABEND	DEBUG ABEND | NOABEND
S	EXCPVR for SORTWK	EXCPVR	NO	NO
S,M,C	Include | Omit fields	NO	NO	INCLUDE | OMIT COND COND/FORMAT
S,C	Exit E15	NO	MODS E15[2] Offset 18 entry[2]	MODS E15[2]
M	Exit E32	NO	Offset 18 entry	NO
S,M,C	Exit E35	NO	MODS E35[2] Offset 22 entry[2]	MODS E35[2]
S,M,C[10]	Exit Exx (xx=11,16-19,31, 37-39, and 61)	NO	MODS Exx	MODS Exx
S,M,C	Inrec fields	NO	NO	INREC FIELDS
S,M,C	Outrec fields	NO	NO	OUTREC FIELDS
S,M,C	Control fields	NO	SORT | MERGE FIELDS/FORMAT	SORT | MERGE FIELDS/FORMAT
C	Copy records	NO	SORT | MERGE FIELDS	SORT | MERGE FIELDS OPTION COPY
S,M	Sum fields	NO	NO	SUM FIELDS/FORMAT
M	Merge input files	NO	MERGE FILES X'04' entry	MERGE FILES
S,M	File size	NO	SORT | MERGE FILSZ | SIZE	SORT | MERGE FILSZ | SIZE OPTION FILSZ | SIZE
S,M,C	Formatted dump	NO	DEBUG FMTABEND	DEBUG FMTABEND
S,M	Checkpoints	IGNCKPT	SORT | MERGE CKPT | CHKPT[3]	SORT | MERGE CKPT | CHKPT[3] OPTION CKPT | CHKPT[3]

Figure 42 (Part 2 of 4). 24-Bit List DFSORT Option Specification/Override

Function	Description of Option	Specified with ICEMAC INV	Specified with 24-Bit List	Specified with SORTCNTL
S,M,C	Record lengths	NO	RECORD LENGTH	RECORD LENGTH
S,M,C	Print control statements[4]	LIST	NO	NO[5]
S,M,C	Maximum storage below 16-megabyte virtual[6]	MAXLIM	NO	NO
S,M,C	Minimum storage	MINLIM	NO	NO
S,M,C	Alternate message ddname	MSGDDN	X'03' entry	NO[7]
S,M,C	Write messages on master console	MSGCON	NO	NO
S,M,C	Print messages	MSGPRT	X'FF' entry	NO[8]
S,M	Bypass Blockset	NO	NO	OPTION NOBLKSET
S,M,C	Release SORTOUT space	OUTREL	NO	OPTION NOOUTREL
S,M,C	SORTOUT secondary allocation	OUTSEC	NO	OPTION NOOUTSEC
S,M,C	Storage over REGION	OVERRGN	NO	NO
S,M,C	System reserved storage[6]	RESALL	NO	OPTION RESALL
S,M,C	Resident modules	RESDNTx	NO	NO
S,M,C	Program reserved storage[6]	RESINV	X'01' entry	OPTION RESINV
S,M,C	Storage	Size	X'00' entry	OPTION MAINSIZE
S,M,C	SMF records	SMF	NO	NO
S,C	Skip records	NO	SORT SKIPREC	SORT SKIPREC OPTION SKIPREC
S,M,C	ddname prefix	NO	Prefix entry	NO[9]
S,M,C	Use of STIMER	STIMER	NO	OPTION NOSTIMER
S,C	Input limit	NO	NO	OPTION STOPAFT
S,M,C	User SVC number	SVC	NO	NO
S,M,C	Record format	NO	RECORD TYPE	RECORD TYPE

Figure 42 (Part 3 of 4). 24-Bit List DFSORT Option Specification/Override

Function	Description of Option	Specified with ICEMAC INV	Specified with 24-Bit List	Specified with SORTCNTL
S	Maximum storage above and below 16-megabyte virtual	TMAXLIM	NO	NO
S,M	Sequence check	VERIFY	NO	OPTION VERIFY | NOVERIFY
S	SORTWK virtual I/O	VIO	NO	NO
S,M	Variable records do not contain all specified control fields	VLSHRT	X'FD' entry	OPTION VLSHRT | NOVLSHRT
S	Release SORTWK space	WRKREL	NO	OPTION NOWRKREL
S	SORTWK secondary allocation	WRKSEC	NO	OPTION NOWRKSEC

Figure 42 (Part 4 of 4). 24-Bit List DFSORT Option Specification/Override

Notes to Figure 42:

1. Does not request dynamic allocation; just supplies defaults.
2. DFSORT terminates if the exit is specified via the parameter list entry and the exit is specified in a MODS statement.
3. Not used if Blockset is selected and IGNCKPT=YES was specified.
4. Not used if MSGPRT=NONE or MSGPRT=CRITICAL is in effect; in this case control statements is not printed.
5. OPTION LIST | NOLIST in SORTCNTL is not used.
6. Not used unless MAINSIZE=MAX is in effect.
7. OPTION MSGDDN in SORTCNTL is not used.
8. OPTION MSGPRT in SORTCNTL is not used.
9. OPTION SORTDD in SORTCNTL is not used.
10. All functions do not apply to all exits. See Figure 16 and Figure 17 on page 139 for applicable exits.

Appendix E. 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.

Format	Description
FI	(fixed point, signed). The complete number is represented by its binary equivalent in either halfword or fullword 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
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 into cards, 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

Format	Description
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[1]	(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 last decimal digit of the number. Example: +247 with + overpunched on 7 becomes F2 F4 C7 hexadecimal
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

[1] The overpunch sign bit is always X'C' for positive and X'D' for negative.

A detailed description of CH, ZD, PD, FI, BI, and FL data formats will be found in the *OS/VS—DOS/VSE—VM/370 Assembler Language Manual*, Section G.

Appendix F. EBCDIC and ISCII/ASCII Collating Sequences

EBCDIC

Figure 41 on page 287 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, and so forth) 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

Figure 43 (Part 1 of 3). EBCDIC Collating Sequence

Collating Sequence	Bit Configuration	Symbol	Meaning
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	=	Equal sign
127	01111111	"	Quotation marks
.			
.			
129	10000001	a	
130	10000010	b	
131	10000011	c	
132	10000100	d	
133	10000101	e	
.			
.			
134	10000110	f	
135	10000111	g	
136	10001000	h	
137	10001001	i	
.			
.			
145	10010001	j	
146	10010010	k	
147	10010011	l	
148	10010100	m	
149	10010101	n	
150	10010110	o	
151	10010111	p	
152	10011000	q	
153	10011001	r	
.			
.			
162	10100010	s	
163	10100011	t	
164	10100100	u	
165	10100101	v	
166	10100110	w	
167	10100111	x	
168	10101000	y	
169	10101001	z	

Figure 43 (Part 2 of 3). EBCDIC Collating Sequence

Collating Sequence	Bit Configuration	Symbol	Meaning
193	11000001	A	
194	11000010	B	
195	11000011	C	
196	11000100	D	
197	11000101	E	
198	11000110	F	
199	11000111	G	
200	11001000	H	
201	11001001	I	
.			
.			
209	11010001	J	
210	11010010	K	
211	11010011	L	
212	11010100	M	
213	11010101	N	
214	11010110	O	
215	11010111	P	
216	11011000	Q	
217	11011001	R	
.			
.			
226	11100010	S	
227	11100011	T	
228	11100100	U	
229	11100101	V	
230	11100010	W	
231	11100111	X	
232	11101000	Y	
233	11101001	Z	
.			
.			
240	11110000	0	
241	11110001	1	
242	11110010	2	
243	11110011	3	
244	11110100	4	
245	11110101	5	
.			
.			
246	11110110	6	
247	11110111	7	
248	11111000	8	
249	11111001	9	
.			
.			
255	11111111		

Figure 43 (Part 3 of 3). EBCDIC Collating Sequence

ISCII/ASCII

Figure 42 on page 292 shows the collating sequence for ISCII/ASCII, character, and unsigned decimal data. The collating sequence ranges from low (00000000) to high (01111111). Bit configurations that 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	

Figure 44 (Part 1 of 3). ISCII/ASCII Collating Sequence

Collating Sequence	Bit Configuration	Symbol	Meaning
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	
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	

Figure 44 (Part 2 of 3). ISCII/ASCII Collating Sequence

Collating Sequence	Bit Configuration	Symbol	Meaning
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

Figure 44 (Part 3 of 3). ISCII/ASCII Collating Sequence

Appendix G. SMF Record (TYPE 16)

The SMF record produced by DFSORT has the following format:

```
-------------------------------------------------------------------------
DISP   NAME                    DESCRIPTION
-------------------------------------------------------------------------
0000   ICESMF   DSECT
0000   ICERDW   DS    XL4      RECORD DESCRIPTOR WORD
0004   ICESIND  DS    B        SYSTEM INDICATOR
       *                       BIT0: 1 = SUBSYSTEM ID FOLLOWS SYSTEM ID
       *                       BIT 1-7: RESERVED
0005   ICERTYP  DS    X        SMF RECORD TYPE. (16)
0006   ICEBTIME DS    FL4      TIME RECORD WAS MOVED TO SMF BUFFER
000A   ICEBDATE DS    PL4      DATE RECORD WAS MOVED TO SMF BUFFER
000E   ICESID   DS    CL4      SYSTEM IDENTIFICATION
0012   ICEJOBNM DS    CL8      JOBNAME
001A   ICERST   DS    FL4      TIME READER RECOGNIZED JOBCARD
001E   ICERDS   DS    PL4      DATE READER RECOGNIZED JOBCARD
0022   ICEUIF   DS    CL8      USER ID (TAKEN FROM COMMON EXIT
       *                         PARAMETER AREA)
002A   ICESTN   DS    XL1      STEP NUMBER
002B   ICERES1  DS    CL3      RESERVED
002E   ICESUBID DS    CL4      SUBSYSTEM ID
0032   ICERSUB  DS    XL2      RECORD SUBTYPE
       ICERSUBS EQU   X'01'    SHORT RECORD
       ICERSUBF EQU   X'02'    FULL RECORD
       ************************************************************
       *  SELF DEFINING SECTION                                   *
       ************************************************************
0034   ICEPROD  DS    F        OFFSET TO PRODUCT SECTION
0038   ICEPRODL DS    H        PRODUCT SECTION LENGTH (10)
003A   ICEPRODN DS    H        NUMBER OF PRODUCT SECTIONS (1)
       *
003C   ICEDATA  DS    F        OFFSET TO SECTION COMMON TO SHORT AND FULL RECS
0040   ICEDATAL DS    H        DATA SECTION LENGTH (38)
0042   ICEDATAN DS    H        NUMBER OF DATA SECTIONS (1)
       *
0044   ICESTAT  DS    F        OFFSET TO RECORD LENGTH STATISTICS
0048   ICESTATL DS    H        STATISTICS SECTION LENGTH (64)
004A   ICESTATN DS    H        NUMBER OF DATA SECTIONS (1|0)
       ************************************************************
       *  PRODUCT SECTION                                         *
       ************************************************************
```

```
DISP   NAME                    DESCRIPTION
004C   ICERECV  DS    CL2      RECORD VERSION.
004E   ICEPRDCT DS    CL8      PRODUCT NAME. '5740SM1'
       ********************************************************************
       *  DATA SECTION. COMMON PART                                       *
       ********************************************************************
0056   ICECDAT  DS    0H       START OF DATA SECTION
0056   ICERES2  DS    CL2      TO PUT NEXT FIELD ON FULLWORD BOUNDARY
0058   ICESTPNM DS    CL8      STEPNAME
0060   ICERCDS  DS    F        NUMBER OF RECORDS SORTED
0064   ICEBYTES DS    F        NUMBER OF BYTES SORTED (SUM OF RECORD LENGTHS)
0068   ICECPUT  DS    F        DFSORT CPU TIME, HUNDREDTHS OF A
SECOND
006C   ICELEN   DS    H        SPECIFIED RECORD LENGTH
006E   ICEIBLK  DS    H        INPUT BLOCKSIZE (MAX)
0070   ICEOBLK  DS    H        OUTPUT BLOCKSIZE
0072   ICEKEYLN DS    H        TOTAL CONTROL FIELD LENGTH (NUMBER
       *                         OF BYTES ACTUALLY COMPARED BY DFSORT)
0074   ICEWBLK  DS    F        NUMBER OF WORK DATA SET TRACKS USED
0078   ICEFLBYT DS    B        BIT 0: RESERVED
       *                       BIT 1-2: 00=FIXED-LENGTH RECORDS
       *                                01=VARIABLE-LENGTH RECORDS
       *                                10=VARIABLE-LENGTH SPANNED RECORD
       *                       BIT 3-4: 00=BLOCKSET
       *                                01=PEERAGE
       *                                10=VALE
       *                                11=CONVENTIONAL TECHNIQUE
       *                         BIT 5: '1'B IF DFSORT DYNAMICALLY INVOKED
       *                         BIT 6-7: RESERVED
0079   ICENDYNA DS    AL1      NUMBER OF ALLOCATED WORK DATA SETS
007A   ICERES3  DS    CL2      RESERVED
       ********************************************************************
       *  DATA SECTION. STATISTICS PART                                   *
       ********************************************************************
         ICEVAR   EQU   ICECTR BEGINNING OF VARIABLE PART
007C   ICECTR   DS    16F      RECORD INTERVAL COUNTERS
00BC             ORG   ICECTR
007C   ICECTR01 DS    F        RECORD RANGE        4 -    15  (   4 -    F)
0080   ICECTR02 DS    F        RECORD RANGE       16 -    31  (  10 -   1F)
0084   ICECTR03 DS    F        RECORD RANGE       32 -    63  (  20 -   3F)
0088   ICECTR04 DS    F        RECORD RANGE       64 -   127  (  40 -   7F)
008C   ICECTR05 DS    F        RECORD RANGE      128 -   191  (  80 -   BF)
0090   ICECTR06 DS    F        RECORD RANGE      192 -   255  (  C0 -   FF)
0094   ICECTR07 DS    F        RECORD RANGE      256 -   511  ( 100 -  1FF)
0098   ICECTR08 DS    F        RECORD RANGE      512 -  1023  ( 200 -  3FF)
009C   ICECTR09 DS    F        RECORD RANGE     1024 -  2047  ( 400 -  7FF)
00A0   ICECTR10 DS    F        RECORD RANGE     2048 -  2095  ( 800 -  FFF)
00A4   ICECTR11 DS    F        RECORD RANGE     4096 -  7167  (1000 - 1BFF)
00A8   ICECTR12 DS    F        RECORD RANGE     7168 - 10751  (1C00 - 29FF)
00AC   ICECTR13 DS    F        RECORD RANGE    10752 - 15359  (2A00 - 3BFF)
00B0   ICECTR14 DS    F        RECORD RANGE    15360 - 20991  (3C00 - 51FF)
00B4   ICECTR15 DS    F        RECORD RANGE    20992 - 26623  (5200 - 67FF)
00B8   ICECTR16 DS    F        RECORD RANGE    26624 - 32767  (6800 - 7FFF)
       ICESMFND EQU   *        END OF RECORD
                END
```

Appendix H. DFSORT Messages and Codes

This section lists the DFSORT messages, explains them, and, when applicable, suggests appropriate responses.

Depending on the DFSORT technique being used, a different message may be issued for the same error situation.

DFSORT produces three types of messages:

- Critical error messages
- Information messages
- Diagnostic messages

Message Format

For printed messages: ICEnnns c text

For displayed messages: ICEnnns c jjjjjjj.ssssssss text

nnn	Message number.
s	Severity code.
	The severity codes are:
	A Error message; programmer action is required. I Information message; usually no programmer action is required.
c	Diagnostic trace character. (This character is not usually needed, but may be requested by your IBM representative for diagnostic purposes.)
jjjjjjj	Jobname
ssssssss	Stepname.
text	Message text.

Figure 45. Message Format

Printing Messages and Control Statements

The type of messages to be printed can be selected at either installation or execution time (for override information, see Appendix D). The messages are written on the message data set; the only exceptions are ICE097I and ICE158A, which are written only to the master console.

For a JCL-invoked DFSORT, a SYSOUT DD statement must be provided when messages are to be printed. For a dynamically invoked DFSORT, a ddname DD statement (where ddname is the name of the alternate message data set specified at either installation or execution time), or a SYSOUT DD statement must be provided. *If a required message data set ddname DD statement is not provided, DFSORT terminates with a return code of 20.*

The following message and control statement printing hierarchy is observed for Blockset, Peerage, and Vale (note that MSGPRT and LIST are used here to represent all methods of specifying the appropriate function; for example, FLAG(I) in the EXEC PARM field is equivalent to MSGPRT=ALL):

- SORTDIAG DD statement—print all messages (including diagnostic messages); print control statements. A message data set ddname DD statement must be provided.

  Note: When SORTDIAG DD is specified, Blockset error messages that are normally suppressed are printed. This may result in an error message being printed for a run that is actually successfully completed by another technique (as indicated by a return code of 0).

- MSGPRT=ALL and LIST in effect—print all messages except diagnostic messages; print control statements. A message data set ddname DD statement must be provided.

- MSGPRT=ALL and NOLIST in effect—print all messages except diagnostic messages; do not print control statements. A message data set ddname DD statement must be provided.

- MSGPRT=CRITICAL and LIST in effect—print only critical error messages; print control statements. A message data set ddname DD statement must be provided.

  Note: Control statements are not printed if MSGPRT=CRITICAL is in effect and DFSORT is dynamically invoked using the 24-bit list.

- MSGPRT=CRITICAL and NOLIST in effect—print only critical error messages; do not print control statements. A message data set ddname DD statement must be provided.

- MSGPRT=NONE in effect—do not print messages; do not print control statements.

Writing Messages to the Master Console

The type of messages to be written to the master console can be selected at installation time.

Note: Because of the addition of the jobname and the stepname, some DFSORT messages may be truncated from the right when they are written to the master console if you change the text in ICEMSGS (see *Installation Guide*).

At installation time, you can specify, through the ICEMAC parameter MSGCON, the class of messages you want to have written to the master console:

MSGCON={ALL | CRITICAL | NONE}

ALL Specifies that all messages except option-in-effect messages (ICE128I through ICE133I) and diagnostic messages (ICE800I through ICE999I) are written to the master console.

CRITICAL Specifies that only critical messages are to be written to the master console.

NONE Specifies that messages are not to be written to the master console.

Notes:

1. *Console message choices are independent of the message data set choices.*
2. *Messages ICE097I and ICE158A are written only to the master console.*
3. *Inclusion of a SORTDIAG DD statement has no effect on console messages.*

Control Statement Coding Errors

DFSORT analyzes control statements in two ways:

- The general format (syntax) of control statements.
- 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 message data set (usually SYSOUT) DD statement is in error or missing and such a statement is required because messages and/or control statements are to be printed, DFSORT prints a message and continues the scan on successive statements.

If control statements are printed, and an error occurs that can be associated with a specific statement, the message follows it in the listing. If the statement is in SYSIN or SORTCNTL and the error can be associated with a specific operation, operand, or value, a pointer ($) is printed on the line below the statement, near to the character in error.

If an error has occurred, the program usually terminates after all control input has been analyzed. However, in some cases, if an error is found while a Blockset technique is being used, DFSORT reverts to another technique instead of terminating.

Return Codes

Upon successful completion, DFSORT returns a return code to the operating system (or the invoking program). If completion is unsuccessful, either a return code or an ABEND is issued, depending on what was specified at installation or execution time. (If the ABEND option is in effect, the user abend code is equal to the error message number.) This code may be interrogated by succeeding job steps. The codes are:

0 Successful Completion
16 Unsuccessful Completion
20 Message Data Set Missing

0—Successful Completion
When DFSORT has been successfully executed, a code of zero is returned and the sort terminates.

16—Unsuccessful Completion

If DFSORT encounters an error during execution that does 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.

20—Message Data Set Missing

For a JCL-invoked DFSORT, a SYSOUT DD statement was not provided. For a dynamically invoked DFSORT, neither a message data set DD statement nor a SYSOUT DD statement was provided.

ICE000I --- CONTROL STATEMENTS/MESSAGES --- 5740-SM1 REL x.y...

Explanation: This indicates the DFSORT release level (x.y) and the time and date for the run.

System Action: None.

Programmer Response: None.

ICE001A TEXT BEGINS IN WRONG COLUMN

Explanation: Critical. A continuation line following a line broken at a comma does not begin within columns 2 through 71; or a continuation line following a line broken at column 71 (with a nonblank entry in 72) does not begin in column 16.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check continuation lines for text beginning in a wrong column.

ICE002I DUPLICATE OR CONFLICTING xxxxxxxx STATEMENT

Explanation: This message is issued if the same operation definer, or mutually exclusive operation definers (SORT and MERGE, or INCLUDE and OMIT), appear more than once in the same source (for example, SORTCNTL).

System Action: The program does not analyze duplicate or conflicting statements. The first one encountered is used.

Programmer Response: No action necessary. For later runs, check control statements.

ICE003A CONTINUATION LINE MISSING

Explanation: Critical. A continuation line has been indicated by the previous line ending with a comma, or with a nonblank entry in column 72, and no line follows.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check for an overflow of parameters into column 72 or a missing continuation line.

ICE005A STATEMENT DEFINER ERROR

Explanation: Critical. A control statement does not contain one of the acceptable operation definers (SORT, MERGE, OPTION, RECORD, MODS, ALTSEQ, DEBUG, INCLUDE, OMIT, INREC, OUTREC, SUM, or END). You may also receive this message for continuation lines after a line that has an error.

System Action: Termination when all control statements scanning is complete.

Programmer Response: Check for blank lines in SYSIN or SORTCNTL. 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).

ICE006A OPERAND DEFINER ERROR

Explanation: Critical. The first operand of a control statement does not begin on the same line as the operation definer, or an operand or operand value is not valid.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check for statements that contain invalid operands, invalid operand values, or 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
- Invalid format type
- Invalid operator

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 been processed and no SORT or MERGE control statement or OPTION COPY statement has been found, or a 24-bit parameter list does not contain a SORT or MERGE control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that the 24-bit parameter list contains a SORT or MERGE control statement. If you are not using a 24-bit parameter list, make sure you have supplied a SORT or MERGE control statement or an OPTION COPY 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.

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 MISSING FIELDS OPERAND DEFINER

Explanation: Critical. A SORT, MERGE, INREC, OUTREC, or SUM control statement does not contain a field definition.

System Action: None.

Programmer Response: Check SORT, MERGE, INREC, OUTREC, or SUM control statement for lack of a field definition (FIELDS 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.

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 duplicate 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. DFSORT has detected a variable-length record too short to contain all fields or, if Blockset is not used, shorter than L4.

For Peerage and Vale, the record does not contain the first byte of the first control field when VLSHRT is in effect.

System Action: The program terminates.

Programmer Response: Decrease L4 if too large. Check the input in both the SORTIN data set and all records inserted at exit E15 to see that all records contain all fields. Remove any which are too short. Check your E15 routine and correct any errors. If you wish to continue processing if DFSORT encounters a short variable-length record, specify VLSHRT.

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, or the maximum length plus displacement has been exceeded when VLSHRT is specified for a sort application.

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. If VLSHRT is specified for variable length sorts, make sure that the length values plus the position values do not exceed 4086; control field manipulation or overlapping may reduce the limit of 4086.

ICE018A INVALID FORMAT

Explanation: Critical. A SORT, MERGE, SUM, INCLUDE, or OMIT statement contains an invalid or missing format type, or more than 112 control fields are specified and Blockset cannot be used. Or- for a SORT or MERGE control statement, the format is invalid for the length specified.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check that each format type is valid for the control statement specified.

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 invalid keywords.

ICE021A NO TYPE OPERAND

Explanation: Critical. A TYPE operand is required 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. The RECORD statement with the TYPE operand is required for VSAM SORTIN/SORTINxx data sets.

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 data entry 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, because no DD statement with the name SORTIN has been supplied. Can also be issued if SORTIN processing is bypassed because of a previous error (for example, ICE021A).

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 or with the input or output length values obtained from SORTINxx or SORTOUT, respectively.

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, and so forth).
- Minimum length (L4) specified or determined as the last field displacement plus length 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.
- The maximum length (L2) is greater than the output record length (L3), but there is no E35 exit.

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 entire 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 probable cause is that you have not specified a SORTOUT data set, have specified E35, and from your E35 routine have passed a return code of 8 (do not return) too early,

when there are still output records remaining. If this is the cause, you can avoid receiving this message by specifying OPTION NOCHECK. If this is not the cause, examine any exit routines (especially E15 and E35) for possible return code or other errors.

If a COBOL invoking program contains an output procedure, make sure the RETURN statement is integrated until the AT END condition is executed. This occurs the next time the RETURN statement is executed *AFTER* the last record has been returned to you.

If the iteration of the RETURN is controlled by a PERFORM statement, the PERFORM logic should be controlled by the execution of the AT END clause of the RETURN statement.

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 SMF (SMFWTM macro).

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 FIELD BEYOND MAXIMUM RECORD LENGTH

Explanation: Critical. A SORT, MERGE, INREC, OUTREC, SUM, INCLUDE, or OMIT field has been defined as extending beyond the maximum input record length, or if INREC is specified, a SORT, MERGE, OUTREC, or SUM field has been defined as extending beyond the maximum reformatted record length.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check SORT, MERGE, INREC, OUTREC, SUM, INCLUDE, and OMIT statements for incorrectly specified field displacement or length. Check RECORD statement for incorrectly specified 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 that may be specified in the MODS statement are E11, E15, E16, E17, E18, E19, 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 errors that resulted in the specification of an invalid program exit number. Numbers that may be specified are E11, E15, E16, E17, E18, E19, 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 a 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 $, @, or #.

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 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 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. One or more of the following errors exist:

- An incorrect number of parameters follows an operand defined on a MODS control statement.
- SYSIN is specified as the third parameter for an exit whose fourth parameter is 'N' or 'C'.
- For those exits prelink-edited by the user (fourth parameter is 'N' or 'C', or null), the third parameter (library ddname) is different for two or more exits.
- For a copy application, SYSIN is specified as the third parameter, or 'T' or 'S' is specified as the fourth parameter.
- An invalid value is specified for the fourth parameter of the statement.
- T is specified for the fourth parameter for an E15 (E35) for which the EXEC statement parameter 'E15=COB' ('E35=COB') is also specified.
- S is specified for the fourth parameter, but the exit is not E11 or E31.

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} [*,N* | *,C* | *,T* | *S*])...

Check that SYSIN is not specified for any exit with the fourth parameter as 'N' or 'C'.

Make sure that T has not been specified as the fourth parameter for an E15 or E35 exit when the EXEC PARM 'E15=COB' or 'E35=COB' has been specified.

Make sure that S has not been specified for the fourth parameter for an exit other than E11 or E31.

Make sure that the dynamic link-edit parameters (SYSIN, T, and S) have not been specified for a copy application.

ICE035A DUPLICATE MODS ROUTINE OPERAND

Explanation: Critical. The same user-written routine is being used for more than one exit in a DFSORT 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 (conventional 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 = n

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 DFSORT for the current application. The number replaces n in the text of the message as shown above. For magnetic tape, Nmax is calculated assuming that 2400-foot tapes are used. For variable-length records, the value is based on maximum record length.

System Action: None.

Programmer Response: None.

ICE039A INSUFFICIENT MAIN STORAGE— ADD nK BYTES

Explanation: Critical. There is not enough main storage available for DFSORT to execute, or main storage is fragmented.

System Action: The program terminates.

Programmer Response: The message gives an estimation of how much more main storage is needed. Add at least that amount to the main storage already allocated to the program by recoding the REGION parameter and/or the MAINSIZE/SIZE value (see Appendix D for ways to specify this).

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.

For MVS/XA, make sure that storage allocation is permitted above 16-megabyte virtual and enough storage is available above 16-megabytes virtual to load the DFSORT modules.

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. In a merge-only application, this message may be caused by incorrect specification of one or more input units (SORTIN01, and so forth), or use of an E32 exit without specifying the number of files.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that the DD statements do not contain errors. For a sort using tape work files, the SORTWKnn numbers must be in sequence, starting with SORTWK01, and at least three work data sets must be assigned to the program. For a conventional merge application, make sure that the numbers for the SORTINnn DD statements are in sequence, starting with SORTIN01.

ICE041A N GT NMAX

Explanation: Critical. The exact number of records specified in the FILSZ or SIZE operand of an OPTION or SORT control statement is greater than the maximum sort capacity calculated by the program.

System Action: The program terminates.

Programmer Response: Check FILSZ or SIZE operand of the OPTION or SORT control statement for errors. 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 DFSORT.

   Examples:

   - a SORTWKnn DD statement and a SORTINnn DD statement are both specified
   - a SORTIN DD statement and a SORTINnn statement are both specified
   - A VSAM SORTINnn DD statement and a non-VSAM SORTINnn DD statement are both specified

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, eliminate the improper DD statements for the application. For the examples listed above, the following statements apply:

- A SORTWKnn DD statement is improper for a merge application.
- The SORTIN and SORTINnn statements are mutually exclusive.
- MERGE input data sets may be either QSAM or VSAM, but not both.

For case 2, eliminate duplicate ddnames.

For case 3, check that the device type specified is supported by the program (see "Direct Access" on page 275) and available at your installation; and check whether you have exceeded the maximum number of areas permitted for the storage type used.

ICE043A INVALID DATA SET ATTRIBUTES SPECIFIED xxxxxx [yyyyyy]

Explanation: Critical. One of the following conditions has been encountered:

- 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 14 on page 123 for a summary of DCB subparameters).

- 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.
- Input data sets to a merge contain different record lengths, and the largest blocksize is not specified first.

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: For case 1, check DD statements for input and output data sets for conflict in the BLKSIZE (blocksize), RECFM (record format), and LRECL (logical record length) subparameters. Input and output must have the same record type (fixed or variable). When using variable-length records without E15 or E35 exits, the maximum SORTIN/SORTINnn LRECL, reformatted record length, or RECORDSIZE (for VSAM data sets), must not exceed the maximum SORTOUT LRECL or RECORDSIZE (for VSAM data sets).

When using fixed-length records with E15 or E35 exits, the following are true:

- For a merge application, SORTOUT LRECL (or RECORDSIZE for VSAM data sets) must not exceed SORTINnn LRECL (or RECORDSIZE for VSAM data sets).
- For a sort application, except with Blockset, SORTOUT LRECL (or RECORDSIZE FOR VSAM data sets) must not exceed SORTIN LRECL (or RECORDSIZE for VSAM data sets).

For case 2, check the volumes of input data sets.

For case 3, check that the SORTIN01 data set has the largest block size.

ICE044I EXIT Exx INVALID OPTION

Explanation: An invalid input/output option was passed to DFSORT at exit E18, E19, E38, or E39. The xx value in the above message text is replaced by the number of the exit at which the error occurred.

System Action: The invalid option is ignored.

Programmer Response: 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 that 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.

ICE046A SORT CAPACITY EXCEEDED—RECORD COUNT: n

Explanation: Critical. Sort capacity has been reached. The count n is an approximation of the number of records that DFSORT 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, DFSORT overrides any system B37 abend and continues processing; this message is only issued when no more space is available on any allocated SORTWKnn data set.

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 intermediate storage space to DFSORT. Note that reverse sequence files may require more space. Alternatively, increase the main storage available to DFSORT.

ICE047A RECORD COUNT OFF, SPECIFIED n, RECEIVED n

RCD COUNT OFF

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 the input phase (and the

intermediate phase of tape work sorts) are not equal, provided an actual value for the FILSZ or SIZE parameter was specified on the OPTION, SORT, or MERGE control statement. The IN field contains the specified value for FILSZ or SIZE. The OUT field contains the end of phase record count, which has been adjusted by the number of records inserted or deleted by exits E15/E32, SKIPREC, STOPAFT, and/or INCLUDE/OMIT processing.

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 causes message ICE025A to be issued together with messages ICE054I and ICE055I.

System Action: The program terminates.

Programmer Response: Make sure that the value of the FILSZ (or SIZE) parameter on the OPTION or SORT control statement is accurate. See also message ICE025A above.

ICE048I NMAX EXCEEDED

Explanation: DFSORT 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 DFSORT. 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.

ICE050I END MERGE PH

Explanation: A tape technique program's intermediate merge phase has been successfully executed.

ICE051A UNENDING MERGE

Explanation: Critical. Nonstandard 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 DFSORT

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: Critical. The current record leaving the intermediate or output phase is not in collating sequence with the last record blocked for output.

System Action: The program terminates.

Programmer Response: If message ICE143I indicates the Blockset technique was selected, specify OPTION NOBLKSET to bypass the DFSORT error.

ICE054I RECORDS—IN: n, OUT: n RCD IN n,OUT n

Explanation: This message lists the number of records read by DFSORT from the input data set(s) and the number of records written to the output data set. The numbers replace n in the text of the message as shown above. In a conventional merging

application, if an exact file size is not specified, the IN field is zero; if an exact file size is specified, it appears in the IN field.

If Peerage or Vale is used with DEBUG CTRx, the n values may not be meaningful.

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 n, DELETE n

Explanation: The number of records inserted and/or deleted during a DFSORT program execution replaces the values shown as n in the above format.

If Peerage or Vale is used with DEBUG CTRx, the n values may not be meaningful.

System Action: None.

Programmer Response: See message ICE054I above.

ICE056A ddname NOT DEFINED

Explanation: Critical. A required DD statement was not supplied. This message appears for:

- Sort or copy without a SORTIN DD statement or E15 exit routine
- Merge without a SORTINnn DD statement or E32 routine
- Sort, merge, or copy without a SORTOUT DD statement or E35 exit routine
- The same OPTION SORTIN and OPTION SORTOUT ddname
- CKPT in effect with no intermediate storage data set specified.

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 errors.

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 | ddname}

Explanation: Critical. The intermediate record created by DFSORT from the input record is either less than 18 bytes when work units are tape (indicated by DEVICE) or is too large for the assigned intermediate storage device with the indicated ddname.

System Action: The program terminates.

Programmer Response: If the intermediate record is too small, redefine input record length to at least 18 bytes. If the length is too large, assign an intermediate storage device with a larger track capacity.

If EQUALS is in effect the maximum record length is reduced by 4 bytes.

ICE061A I/O ERROR, DD name, 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 'DEV' field in the message.
- 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.

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 DFSORT to use a different sorting technique. This is done with the OPTION control statement.

ICE062A LINK-EDIT ERROR

Explanation: Critical. The linkage editor found a serious error.

System Action: The program terminates.

Programmer Response: Check that all the DD statements used by the linkage editor were included (SYSPRINT, SYSLIN, SYSUT1, and SYSLMOD), and that they are correct.

If the DD statements are correct, make sure that:

- All user routines in libraries are valid object decks or load modules
- All user routines in the system input stream (SYSIN) are valid object decks
- Modules to be link-edited together do not contain duplicate entry point names.

ICE063A OPEN ERROR ddname

Explanation: Critical. An error occurred during execution of the OPEN routine for the data set with the indicated ddname.

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.
- Data sets required for dynamic link-editing cannot be opened.

ICE064A DELETE ERR

Explanation: Critical. DFSORT 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 DFSORT 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:

- Check that the MODS statement specifies the correct routines.
- Make sure the SYSIN data set contains all —and only those— exit routines specified by the MODS statement.
- Check for misplaced job control language statements, especially preceding a user exit routine on SYSIN.
- Make sure nothing follows the last object deck in SYSIN.

ICE066I APROX RCD CNT xxxxxxxx

Explanation: Sort capacity has been reached. The count xxxxxxxx is an approximation of the number of records the DFSORT 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 IN JCL EXEC PARM OR INVOKED PARMLIST

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 DFSORT initiated by ATTACH, LINK, or XCTL. If a parameter is specified more than once, the first entry is used (if valid).

System Action: Processing continues. Invalid and duplicate parameters are ignored.

Programmer Response: No action is necessary. For later runs, be sure that the optional parameters are valid. Valid parameters are described in Chapter 3.

ICE068A OUT OF SEQ SORTINxx

Explanation: Critical. During a merge, 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 forth.

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, in the same order, and that they all have a similar format. Check whether you have also received message ICE072I.

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.

ICE071A INVALID RETURN CODE FROM EXIT Exx

Explanation: Critical. A user routine at the exit Exx (can be E15, E32, or E35) has returned an invalid return code to the program, or a return code of 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 8, 12, or 16 for E32).

Check also that:

- An E15 or E35 routine always finishes by returning 8 (do not return) or 16 (terminate).
- If no SORTIN DD statement is provided, the E15 routine only passes back a return code of 8 (do not return), 12 (insert) or 16 (terminate).
- If no SORTOUT DD statement is provided, the E35 routine only passes back a return code of 4 (delete), 8 (do not return), or 16 (terminate).
- In a COBOL invoking program containing an input and/or output procedure, control passes through the end of the input procedure before the output procedure is entered, and through the end of the output procedure before the sort is terminated.

ICE072I FIELD NOT WITHIN MINIMUM RECORD LENGTH

Explanation: A RECORD statement specifies a minimum record length (L4) which cannot contain all fields specified in the SORT, MERGE, INCLUDE, OMIT, INREC, OUTREC, or SUM statement(s).

System Action: The L4 value is adjusted. Processing continues.

Programmer Response: None.

ICE073A VARIABLE RECORD TOO LONG

Explanation: Critical. A variable-length record was larger than the maximum length specified or defaulted.

System Action: The program terminates.

Programmer Response: Check the input both at E15/E35, if used, and in SORTIN. Then either delete the extra long records or increase the SORTIN/SORTOUT DD statement DCB LRECL value or the RECORD statement L1/L2/L3 value.

If you are using INREC and/or OUTREC, check that a SORTOUT DD LRECL value or L3 value is at least as large as the reformatted output record length.

If you have VSAM records, remember that they are increased in length by the 4-byte record descriptor word added when they enter DFSORT. 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) overridden.

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 or ICE089I), 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 or ICE089I.

ICE075A VSAM CB ERROR (xx) AT aaaaaa

Explanation: Critical. aaaaaa represents the storage address of the control block 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 tries to close all remaining VSAM data sets before terminating.

Programmer Response: Refer to *VSAM Programmer's Guide* or *VSAM Reference* for the meaning of the return code, and if possible take appropriate action.

ICE076A VSAM INPUT ERROR i(xxx) yyyyyyyy

Explanation: Critical. i is replaced by either P (physical) or L (logical), describing the type of error encountered. *xxx* is the VSAM feedback code (RPLERRCDE) from a GET macro, in decimal; and *yyyyyyyy* is the ddname of the data set in error.

System Action: The program terminates.

Programmer Response: Refer to the *VSAM Programmer's Guide* or *VSAM Reference* for the meaning of the return code, and if possible take appropriate action.

ICE077A VSAM OUTPUT ERROR i(xxx) [yyyyyyyy]

Explanation: Critical. i is replaced by either P (physical) or L (logical), describing the type of error encountered. *xxx* is the VSAM feedback code (RPLERRCDE) from a PUT macro, in decimal. *yyyyyyyy* (if available) is the ddname of the data set in error.

System Action: The program terminates.

Programmer Response: Refer to the *VSAM Programmer's Guide* or *VSAM Reference* for the meaning of the return code, and if possible take appropriate action.

ICE078A VSAM OPEN ERROR (xxx) yyyyyyyy

Explanation: Critical. *xxx* is the VSAM OPEN ERROR return code (ACBERFLG) 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* or *VSAM Reference* 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: Critical. *xxx* is the VSAM CLOSE ERROR return code (ACBERFLG), 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: Critical. Certain dynamic areas and routines have exceeded the fixed amount of storage allocated for them. This situation can only arise if a large number of data sets are allocated and/or a large number of SORT, MERGE, INCLUDE, OMIT, INREC, OUTREC, or SUM fields are specified.

System Action: The program terminates.

Programmer Response: Specify fewer work data sets and/or fewer fields. Alternatively, if message ICE143I indicates the Blockset technique was selected, specify OPTION NOBLKSET to bypass the Blockset technique.

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: Critical. 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 *System Programming Library: Job Management* or *System Programming Library: System Modifications* for the codes.

If VIO=NO is in effect and the virtual devices at your installation do not have corresponding real disks, specify non-virtual work data sets.

ICE084I {BSAM | EXCP | VSAM} ACCESS METHOD USED FOR ddname

Explanation: Identifies the access method used for the identified data set.

System Action: None.

Programmer Response: None, unless you have any problems reading or writing the data set. If EXCP were used, you could force DFSORT not to use EXCP by use of the DEBUG control statement or the EXEC statement.

ICE087I EXCPVR CANCELLED

Explanation: Not enough pages were available for page fixing. The program uses normal EXCP for its disk work files.

System Action: None.

Programmer Response: None.

ICE088I jobname.stepname, INPUT LRECL=n, BLKSIZE=n, 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 are present
```

System Action: None.

Programmer Response: None.

ICE089I jobname.stepname, INPUT LRECL=n, TYPE={F | V}

Explanation: As for ICE088I, but used when all records are supplied via exits E15/E32, or when SORTIN is a VSAM data set, and Blockset was not selected.

System Action: None.

Programmer Response: None.

ICE092I MAIN STORAGE = (x,y,z), NMAX = n

Explanation: Information related to the DFSORT application:

x is the main storage value specified, or supplied by default.

y is the main storage theoretically available to DFSORT, considering MINLIM figure specified when the program was installed.

z is the main storage actually available to DFSORT, after any other program took what it needed from the partition or region (invoking program and/or exit routines).

This value is not meaningful if the storage taken by the invoking program or exit is greater than REGION or SIZE/MAINSIZE.

For MVS/XA, the x, y, and z references to main storage apply to the total storage above and below 16-megabyte virtual.

n is the approximate number of records that can be sorted:

- If no intermediate data sets are specified, n is determined using total available main storage (for MVS/XA, storage above and below 16-megabyte virtual).
- If intermediate data sets are specified, n is determined using the sum of the extents allocated when sorting begins (that is, primary extents only for temporary data sets, and/or previously allocated extents for permanent data sets). Note that the availability of unallocated secondary extents, especially when intermediate storage is reallocated in cylinders, can make the actual number of records which can be sorted MUCH HIGHER than the value shown for n.

System Action: None.

Programmer Response: None, unless DFSORT subsequently terminated abnormally. In that case, check the *z* value to see how much storage was really available to DFSORT. If space was the problem, you have probably also received message ICE039A; but if storage was heavily fragmented, the result could instead be a system 80A abend in either DFSORT or one of your own routines. Note that you could need considerably more than the normal minimum if the partition or region is fragmented.

ICE093I MAIN STORAGE = (MAX,y,z), NMAX = n

Explanation: Information related to the DFSORT application:

MAX was specified or the value specified is the same as MAXLIM.

y is the main storage theoretically available to DFSORT, considering any TMAXLIM, or MAXLIM figures specified when the program was installed.

z is the main storage actually available to DFSORT, after any other program took what it needed from the partition or region (invoking program and/or exit routines). RESALL and RESINV are not considered.

This value is not meaningful if the storage taken by the invoking program or exit is greater than REGION or SIZE/MAINSIZE.

For MVS/XA, the y, and z references to main storage apply to the total storage above and below 16-megabyte virtual.

n is the approximate number of records that can be sorted:

- If no intermediate data sets are specified, n is determined using total available main storage (for MVS/XA, storage above and below 16-megabyte virtual).
- If intermediate data sets are specified, n is determined using the sum of the extents allocated when sorting begins (that is, primary extents only for temporary data sets, and/or previously allocated extents for permanent data sets). Note that the availability of unallocated secondary extents, especially when intermediate storage is reallocated in cylinders, can make the actual number of records which can be sorted MUCH HIGHER than the value shown for n.

System Action: None.

Programmer Response: None, unless DFSORT subsequently terminated abnormally. In that case, check the *z* value to see how much storage was really available to DFSORT. If space was the problem, you probably also received message ICE039A; but if storage was heavily fragmented, the result could instead be a system 80A abend in either DFSORT or one of your own routines. Note that you could need considerably more than the normal minimum if the partition or region is fragmented.

ICE094I SMF RECORDS NOT WRITTEN

Explanation: SMF records were requested, but the SMF option is not present in the system or SMF is not active.

System Action: The data collection for the record length statistics and the writing of the SMF record to the SMF data set is 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. For valid keywords, refer to Chapter 2.

ICE096I SUCCESSFUL RECOVERY FROM B37 ABEND(S) FOR WORK DATA SET(S)

Explanation: DFSORT 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 RECOVERING FROM B37 ABEND ON SORTWK DATASET

Explanation: Issued only to the master console after a B37 ABEND 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 = n BYTES

Explanation: n 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: None.

ICE099A BLDL FAILED FOR ddname DATA SET

Explanation: Critical. A bad return code was returned from a BLDL macro instruction issued when the identified data set is defined as a PDS member.

System Action: The program terminates.

Programmer Response: Ensure that the PDS member specified as ddname exists.

ICE100A OPERATING SYSTEM NOT SUPPORTED

Explanation: Critical. This operating system is not supported by this release of DFSORT. Only current or subsequent releases of the following systems are supported:

- OS/VS1 Release 7.0
- OS/VS2 MVS Release 3.8
- MVS/XA
- MVS/370 (OS/VS MVS with MVS/370 Data Facility Product installed)

System Action: The program terminates.

Programmer Response: Execute DFSORT on one of the supported operating systems.

ICE101A xxxxxxxxxxxx STATEMENT NOT SUPPORTED FOR TECHNIQUE USED

Explanation: Critical. The control statement indicated by xxxxxxxxxxxx is not supported for the technique used (tape work data set sort or conventional MERGE). xxxxxxxxxxxx is replaced by INCLUDE/OMIT, SUM, INREC, or OUTREC.

System Action: The program terminates.

Programmer Response: Rerun job with SORTDIAG DD statement to get reason for the revert from message ICE800I. Remove cause of revert or remove the indicated statement.

ICE102A MISSING COND OPERAND DEFINER

Explanation: Critical. An INCLUDE or OMIT control statement does not contain a logical expression definition.

System Action: Termination when all control scanning is complete.

Programmer Response: Check the INCLUDE or OMIT control statement for lack of a logical expression definition (COND operand).

ICE103A INVALID INCLUDE OR OMIT STATEMENT OPERAND

Explanation: Critical. An invalid keyword operand has been detected on an INCLUDE or OMIT control statement.

System Action: Termination when all control scanning is complete.

Programmer Response: Make sure that the INCLUDE or OMIT control statement does not contain an invalid keyword operand.

ICE104A INVALID INREC OR OUTREC STATEMENT OPERAND

Explanation: Critical. An invalid keyword operand has been detected on an INREC or OUTREC control statement.

System Action: Termination when all control scanning is complete.

Programmer Response: Make sure that the INREC or OUTREC control statement does not contain an invalid keyword operand.

ICE105A INVALID SORT, MERGE, OR SUM STATEMENT OPERAND

Explanation: Critical. An invalid keyword operand has been detected on a SORT, MERGE, OR SUM control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response:

- Make sure that the SORT, MERGE, OR SUM control statement does not contain an invalid keyword operand.
- Make sure that FIELDS=COPY and FORMAT=f are not both specified for the SORT or MERGE control statement.
- Make sure that FIELDS=NONE and FORMAT=f are not both specified for the SUM control statement.

ICE106A DUPLICATE INCLUDE OR OMIT STATEMENT OPERAND

Explanation: Critical. A keyword operand is defined twice on an INCLUDE or OMIT control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check INCLUDE or OMIT control statement for a duplicated keyword operand.

ICE107A DUPLICATE INREC OR OUTREC STATEMENT OPERAND

Explanation: Critical. A keyword operand is defined twice on an INREC or OUTREC control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check INREC or OUTREC control statement for a duplicated keyword operand.

ICE108A DUPLICATE SUM STATEMENT OPERAND

Explanation: Critical. A keyword operand is defined twice on a SUM control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Check SUM control statement for a duplicated keyword operand.

ICE109A SUM FIELD DISPLACEMENT OR LENGTH VALUE ERROR

Explanation: Critical. An invalid length or displacement (position) value is specified in a sum field definition on a SUM 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 the SUM control statement were specified correctly. For BI and FI, length must be 2, 4, or 8 bytes; for PD, length must be 1 through 16 bytes; for ZD, length must be 1 through 18 bytes. Make sure that the length value plus the position value does not exceed 4093.

ICE110I VERIFY NOT USED WITH SUM

Explanation: A sum control statement has been specified and VERIFY is in effect. VERIFY cannot be used with SUM (with a technique other than BLOCKSET). NOVERIFY is forced into effect.

System Action: Processing continues with NOVERIFY in effect.

Programmer Response: None.

ICE111A INREC OR OUTREC FIELD DISPLACEMENT OR LENGTH VALUE ERROR

Explanation: Critical. One of the following errors has been detected in an INREC or OUTREC field value (p,m,a are referred to as an input field; nX and nZ are referred to as separation fields):

1. The position or length value of an input field is less than 1 or the position value plus the length value of an input field is greater than 32001.
2. The first input field consists of only one value (that is, it contains a position value without a length value).
3. An invalid input field alignment value is specified; only H, F, or D is allowed.
4. An invalid separation field value is specified. Only nX or nZ is allowed, where n must be between 1 and 256

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that field values in the FIELDS operand of the INREC or OUTREC control statements are specified correctly.

ICE112I EQUALS NOT USED WITH SUM

Explanation: A SUM control statement has been specified and EQUALS is in effect, but the BLOCKSET technique was not selected. EQUALS with SUM is not supported by the other techniques.

System Action: Processing continues; EQUALS is not used.

Programmer Response: Determine why BLOCKSET could not be used (check diagnostic message ICE800I, which is produced if SORTDIAG DD is specified), and, if possible, remove the cause of the revert.

ICE113A CONDITION VALUE INVALID

Explanation: Critical. An invalid value is specified in a condition definition on an INCLUDE or OMIT control statement. The value may be a displacement, length, format, or constant.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that all the values are correct in the COND operand of the INCLUDE or OMIT control statement. Make sure that the length value plus the position value does not exceed 4093.

ICE114A CONDITION FIELDS COMPARISON INVALID

Explanation: Critical. An invalid field-to-field or field-to-constant comparison is specified in a condition definition on an INCLUDE or OMIT control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that all the field-to-field and field-to-constant comparisons in the COND or FORMAT operands of the INCLUDE or OMIT control statements contain valid combinations.

ICE115A INSUFFICIENT MAIN STORAGE

Explanation: Critical. Storage is fragmented and/or reserved storage value is too large, or exit sizes are too large compared to the total storage available to DFSORT.

For MVS/XA, main storage refers to the storage **below** 16-megabyte virtual.

Programmer Response: See"Tuning Main Storage" on page 209 (or add more main storage in 8K increments).

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.

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.

ICE116A SORT CAPACITY EXCEEDED

Explanation: Critical. Sort capacity has been reached. If intermediate storage is on disk, and secondary allocations have been allowed, DFSORT overrides any system B37 abend and continue processing; this message is only issued when no more space is available on any allocated SORTWKnn data set.

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 space to DFSORT. Note that reverse sequence files may require more space. Alternatively, increase the main storage available to DFSORT.

ICE117A I/O ERROR, jobname, stepname, unit address, device type, DDname, operation attempted, error description, last seek address or block count, access method. (SYNADAF)

Explanation: Critical. This message is generated for one of the 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 permanent error occurred during an I/O operation on the indicated device.

The most probable 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.

ICE119A SUM FIELD OVERLAPS CONTROL FIELD

Explanation: Critical. A sum field is specified in a SUM control statement which overlaps a control field specified in a SORT or MERGE control statement.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that the sum fields in the FIELDS operand of the SUM control statement do not overlap control fields in the FIELDS operand of the SORT or MERGE control statement.

ICE120I EXIT Enn IGNORED

Explanation: Exit E14 or E2n has been specified on the MODS statement, as identified in the message by Enn. Exit Enn is not supported and can not be entered.

System Action: The syntax of the Enn operand is checked for the format "(a,b,c)" or "(a,b,c,d)", but the exit is not used.

Programmer Response: Optional. Correct the MODS statement by removing the identified Enn operand.

ICE123I CKPT or CHKPT OPTION IGNORED

Explanation: CKPT or CHKPT was specified on the SORT, MERGE, or OPTION control statement, IGNCKPT was specified at installation time, and the Blockset technique was selected. The Blockset technique does not support the automatic checkpoint/restart facility, so the CKPT or CHKPT option was ignored.

System Action: The program continues, but no checkpoints are taken.

Programmer Response: If checkpoints must be taken, specify NOBLKSET on the OPTION control statement.

ICE124A SUM FIELD OVERLAPS RECORD DESCRIPTOR WORD

Explanation: Critical. A sum field is specified in a SUM control statement which overlaps the record descriptor word (RDW) of the variable records being processed.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that the sum fields in the FIELDS operand of the SUM control statement do not overlap the RDW.

ICE125A SUM FIELD OVERLAPS SUM FIELD

Explanation: Critical. A sum field is specified in a SUM control statement which overlaps another sum field.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that the sum fields in the FIELDS operand of the SUM control statement do not overlap.

ICE126A REFORMATTING FIELDS ARE INCONSISTENT

Explanation: Critical. One of the following inconsistencies has been detected between the reformat fields specified in the INREC and OUTREC statement and the record format (p,m,a are referred to as an input field; nX and nZ are referred to as a separation field):

1. The last position value is specified without a corresponding length value when fixed-length records are being processed. This usage is only allowed with variable-length records.

2. A single input field containing only bytes from the RDW is specified when variable-length records are being processed. At least one separation field byte or one input field byte must be included in addition to the RDW.

3. The first field is a separation field or an input field which does not contain the RDW when variable-length records are being processed. The first field must contain the RDW.

4. The last position value is specified with a corresponding length value for INREC, but not for OUTREC, or vice versa. INREC and OUTREC must both include or exclude variable data at the end of the record.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Make sure that field values in the FIELDS operand of the INREC or OUTREC control statements are specified correctly.

ICE128I OPTIONS:
SIZE=n,MAXLIM=n,MINLIM=n,
EQUALS=x,LIST=x,ERET=a,
MSGDDN=b

ICE129I OPTIONS:
VIO=x,EXCPVR=x,RESDNT=c,
SMF=e,WRKSEC=x,OUTSEC=x,
VERIFY=x,CHALT=x,DYNALOC=d

ICE130I OPTIONS:
RESALL=n,RESINV=n,SVC=n,
CHECK=x,WRKREL=x,OUTREL=x,
CKPT=x,STIMER=x,COBEXIT=f

ICE131I OPTIONS:
TMAXLIM=n,ARESALL=n,
ARESINV=n,OVERRGN=n

ICE132I OPTIONS: VLSHRT=x

Explanation: Messages ICE128I through ICE132I indicate the options in effect for a sort application. When DFSORT is invoked by JCL, these options may come from the EXEC PARM field, the SYSIN data set, or the installation defaults. When DFSORT is invoked dynamically, these options may come from the SORTCNTL data set, the invocation parameter list, or the installation defaults. DFSORT may also change the options due to conflicts between specifications or for performance reasons. For information on where the options may be specified and on the order of override, see Appendix E, "Data Format Examples" on page 297. Message values are as follows:

x Y for YES; N for NO
n a decimal value
a RC16 or ABEND
b dynamic invocation message data set ddname
c ALL, MON, or NONE
d N for NO, or (y,n) where y is the dynamic allocation device name and n is a decimal value
e SHORT, FULL, or NO
f COB1 or COB2

System Action: None.

Programmer Response: None.

Note: Message ICE131I is printed only for a Blockset sort. However, the TMAXLIM, ARESALL, and ARESINV values are shown as zero except when running under MVS/XA with DEBUG BUFFERS=ANY (the default) in effect.

ICE134I NUMBER OF BYTES SORTED: n

Explanation: n is the total number of bytes "sorted," that is, the number of bytes in records for which control field processing must be performed. The following insert/delete/alter processing is performed before control field processing and is taken into account when determining n: SORTIN, SKIPREC, STOPAFT, E15, and/or INCLUDE/OMIT. Other processing that may affect the number of bytes actually manipulated by DFSORT (for example, INREC) is done after

control field processing, and is not taken into account when determining n. The maximum value that can be displayed for n is 4294967295. Above this value, n is not accurate.

System Action: None.

Programmer Response: None.

ICE136I ddname SPACE REALLOCATED IN CYLINDERS

Explanation: DFSORT reallocated the indicated work data set in cylinders to achieve better performance.

System Action: None.

Programmer Response: To optimize performance for future runs, change your JCL to use cylinder allocation for the indicated work data set.

ICE137A VSAM INPUT ERROR i(xxx) text from VSAM SYNAD

ICE137A (CONT.) text from VSAM SYNAD continued

Explanation: Critical. i is replaced by either P (physical) or L (logical), describing the type of error encountered. *xxx* is the VSAM feedback code from a GET macro, in decimal, and it is followed by the VSAM SYNAD message.

System Action: The program terminates.

Programmer Response: Refer to *VSAM Programmer's Guide* or *VSAM Reference* for the meaning of the return code, and if possible take appropriate action.

ICE138A VSAM OUTPUT ERROR i(xxx) text from VSAM SYNAD

ICE138A (CONT.) text from VSAM SYNAD continued

Explanation: Critical. i is replaced by either P (physical) or L (logical), describing the type of error encountered. *xxx* is the VSAM feedback code from a PUT macro, in decimal. *yyyyyyyy* is the VSAM SYNAD message.

System Action: The program terminates.

Programmer Response: Refer to *VSAM Programmer's Guide* or *VSAM Reference* for the meaning of the return code, and if possible take appropriate action.

ICE141A SPANNED RECORD ON ddname COULD NOT BE ASSEMBLED

Explanation: Critical. A spanned record on the indicated data set could not be properly assembled. This message is generated for one of the following reasons:

- A segment length greater than LRECL.
- A segment length less than 4 bytes.
- The order of the segment codes is incorrect. (Beginning=1, continuation=3, ending=2.)
- Total length of segments greater than LRECL.

System Action: The program terminates.

Programmer Response: Check the indicated data set for incorrect spanned records.

ICE142I ddname NOT FOUND - SYSOUT USED

Explanation: A DD statement for ddname (the specified alternate message data set) was not found. SYSOUT was used instead.

System Action: Processing continues; messages are written to SYSOUT.

Programmer Response: If you want to use an alternate message data set, provide a DD statement for ddname.

ICE143I t a TECHNIQUE SELECTED

Explanation: 't' indicates the DFSORT technique chosen for the run, and 'a' indicates the application chosen for the DFSORT technique. Message values are as follows:

t BLOCKSET (disk sort, standard merge, or copy)
PEERAGE (disk sort)
VALE (disk sort)
CONVENTIONAL (tape sort or nonstandard merge)

a SORT
MERGE
COPY

System Action: None.

Programmer Response: None.

ICE144A GENERATED SUM ROUTINE GREATER THAN 4K BYTES

Explanation: Critical. The routine generated by DFSORT to perform the summary function specified in a SUM statement has a total length greater than the maximum of 4096 bytes.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Reduce the number of summary fields.

ICE146I END OF STATEMENTS FROM xxxxCNTL—PARAMETER LIST STATEMENTS FOLLOW

Explanation: When a data set specified with a DD statement for xxxxCNTL is present, this message separates the listing of its statements from the listing of the statements specified in the parameter list. The statements for xxxxCNTL (if any) and the parameter list (if any) are listed only if LIST is in effect (see message ICE128I).

A statement other than OPTION or DEBUG in the xxxxCNTL data set completely overrides the same or corresponding statement in the parameter list.

A DEBUG statement in the xxxxCNTL data set overrides only the same or corresponding operands in a DEBUG statement in the parameter list. This selective override does not affect the other operands in a DEBUG statement in the xxxxCNTL data set or parameter list.

An OPTION statement in the xxxxCNTL data set overrides only the same or corresponding operands in an OPTION, SORT, or MERGE statement in the parameter list. This selective override does not affect the other operands in an OPTION, SORT, or MERGE statement in the xxxxCNTL data set or parameter list. (Note that OPTION statement operands LIST, NOLIST, MSGDDN, MSGPRT, SORTDD, SORTIN, and SORTOUT are not used if specified in the xxxxCNTL data set.)

For complete details on the order of override, see Appendix B.

System Action: None.

Programmer Response: None.

ICE147A OPTION STATEMENT OPERAND SORTIN OR SORTOUT NOT ALLOWED

Explanation: Critical. The OPTION statement operand SORTOUT was specified for a Conventional merge, or the operand SORTIN or SORTOUT was specified for a tape work data set sort.

System Action: The program terminates.

Programmer Response: Remove the SORTIN or SORTOUT operand. You may use the SORTDD operand of the OPTION statement to provide alternate ddnames for SORTIN and SORTOUT.

ICE148A ddname CONCATENATION NOT ALLOWED

Explanation: Critical. SORTINnn or SORTWKnn data sets were concatenated.

System Action: The program terminates.

Programmer Response: Use separate DD statements for SORTINnn or SORTWKnn data sets.

ICE150I VLSHRT NOT USED

Explanation: The VLSHRT option is in effect, and one or more of the following situations exists:

- An INCLUDE, OMIT, INREC, OUTREC, or SUM statement is specified
- More than one control field is specified, and Blockset has not been selected
- Blockset has been selected for a merge application
- The application is a copy (VLSHRT is not meaningful for copy)
- Blockset has not been selected, and EQUALS is in effect

VLSHRT cannot be used under these circumstances.

System Action: Processing continues; VLSHRT is not used.

Programmer Response: None, unless message ICE015A is received, in which case you should respecify your fields to be within the shortest record in the input data set(s). Alternatively, if the application is merge, the NOBLKSET option on the OPTION control statement can be used

ICE151A GENERATED INCLUDE/OMIT ROUTINE GREATER THAN 4K BYTES

Explanation: Critical. The routine generated by DFSORT to perform the function specified in an INCLUDE/OMIT statement has a total length greater than the maximum of 4096 bytes.

System Action: Termination when all control statement scanning is complete.

Programmer Response: Reduce the number of INCLUDE/OMIT fields.

ICE152I OVERFLOW DURING SUMMATION

Explanation: Overflow occurred for one or more pairs of summary fields during the run.

System Action: Summation continues for pairs of records for which overflow does not occur. Summation is not performed for pairs of records for which overflow would occur resulting in more than one record with the same control field.

Programmer Response: Redesign the records so that summary fields do not overflow, or, if possible, increase the size of the summary field(s) using INREC (see INREC Example 2).

ICE153A COBOL EXIT NOT SUPPORTED FOR TECHNIQUE USED

Explanation: Critical. An E15 or E35 exit written in COBOL was specified, but is not supported for the technique used (tape work data set sort or conventional merge).

System Action: The program terminates.

Programmer Response: Rerun job with SORTDIAG DD statement to get reason for the revert from message ICE800I. Remove cause of revert or remove the COBOL exit.

ICE154A STOPAFT NOT SUPPORTED FOR TECHNIQUE USED

Explanation: Critical. STOPAFT is in effect but is not supported for the technique used (tape work dataset sort).

System Action: The program terminates.

Programmer Response: Use disk work dataset, if possible, or remove the STOPAFT option from the OPTION statement.

ICE155I STOPAFT OR SKIPREC NOT APPLICABLE TO MERGE

Explanation: A MERGE control statement has been specified and STOPAFT and/or SKIPREC is in effect. STOPAFT and/or SKIPREC cannot be used with MERGE, so it is ignored.

System Action: Processing continues; STOPAFT and/or SKIPREC is ignored.

Programmer Response: None.

ICE156I MAIN STORAGE ABOVE 16-MEGABYTE = (y,z)

Explanation: Information on the amount of storage available above 16-megabyte virtual for a MVS/XA system.

y is the upper limit of main storage available to DFSORT above 16-megabyte virtual.

z is the actual amount of main storage available to DFSORT above 16-megabyte virtual, after DFSORT has released the ARESALL and ARESINV space.

System Action: None.

Programmer Response: None.

Note: This message is printed for a Blockset sort under MVS/XA. The values are shown as zero if DEBUG BUFFERS=BELOW is in effect.

ICE157I EXEC PARM E15=COB OR E35=COB AND NO MODS EXIT

Explanation: E15=COB has been specified without a corresponding E15 operand on the MODS statement and/or E35 has been specified without a corresponding E35 operand on the MODS statement.

System Action: Processing continues. The EXEC parameter E15=COB or E35=COB is ignored.

Programmer Response: Specify an E15 and/or E35 operand on a MODS statement or remove the EXEC parameter.

ICE158A SYSOUT DD STATEMENT MISSING

Explanation: Critical. Issued only to the master console to indicate that a message data set was required, but not specified. Corresponds to DFSORT return code 20.

System Action: The program terminates.

Programmer Response: Specify a DD statement for the message data set, using the ddname specified by MSGDDN (if any) or SYSOUT.

ICE159A MODULE ICECOB2 NOT FOUND

Explanation: Critical. Module ICECOB2 was not installed as part of DFSORT. An E15 or E35 exit routine written in COBOL is to be used and COBEXIT=COB2 is in effect. Module ICECOB2 is required for COBEXIT=COB2.

System Action: The program terminates.

Programmer Response: If IBM VS COBOL II has become available on your system since this release of DFSORT was installed, ask your system programmer to update the DFSORT installation.

If IBM VS COBOL II is not available on your system, COBEXIT-COB2 cannot be used.

ICE160A COPY FUNCTION COULD NOT BE USED - REASON CODE IS nn

Explanation: Critical. The COPY function is specified but cannot be processed. Reason code values are as follows:

1. BDAM input and/or output data set
2. SORTOUT is tape with DISP=OLD or DISP=MOD, and LRECL, RECFM, or BLKSIZE is not specified in DD statement.
3. System error attempting to open a data set
4. An input and/or output data set resides on an unsupported device
5. ASCII tapes with the following parameters:

   OPTCD=Q & RECFM=D & BUFOFF¬=L

   or

   OPTCD=Q & RECFM¬=D & BUFOFF¬=0
6. System error attempting to read DSCB for an input data set

7. System error attempting to read DSCB for an output data set

System Action: Termination when all control statement scanning is complete.

Programmer Response: Remove the indicated condition.

ICE161A COBEXIT=COB2 AND COBOL E15 AND E35 EXITS WERE FOUND

Explanation: Critical. When COBEXIT=COB2 is in effect, separately compiled COBOL E15 and E35 exits are not allowed together for COPY processing.

System Action: Termination when all control statement scanning is complete.

Programmer Response: When COBEXIT=COB2 is in effect, use either a COBOL E15 exit or a COBOL E35 exit for COPY applications, but not both. If your exits are actually written in OS VS COBOL, make sure that COBEXIT=COB2 is not in effect.

Diagnostic Messages for Debugging

Note: The following diagnostic messages can occur when the SORTDIAG DD statement is present.

ICE800I BLOCKSET TECHNIQUE COULD NOT BE USED—REASON CODE IS nn

Explanation: The primary DFSORT technique, Blockset, could not be used for this application. nn is the revert reason code. See the following list to determine the reason associated with this code; then use the index of this manual to locate more information about the reason.

1 Critical message issued

5 SORTIN JFCB - BDAM specified in DD Statement

6 SORTIN DSCB - BDAM specified in data set label

7 SORTOUT JFCB - BDAM specified in DD Statement

8 Old tape SORTOUT

10 DUMMY or SPOOL SORTWKxx

12 DCB ABEND exit taken

14 Compare or sum length too long

16 Unsupported SORTIN and/or SORTOUT device

17 Unsupported SORTWKxx device (e.g. tape)

22 ASCII tapes with following parameters:
OPTCD=Q & RECFM=D & BUFOFF¬=L

or

OPTCD=Q & RECFM¬=D & BUFOFF¬=0

23 OBTAIN failed for SORTIN DS

24 OBTAIN failed for SORTOUT DS

28 E18 or E38 with tape SORTIN(XX)

47 NOBLKSET specified

48 CKPT and IGNCKPT=NO specified

58 Dynamic tape SORTWKnn data set

63 Internal control word longer than maximum allowed

64 Internal control word longer than maximum allowed

65 Work data set not specified

68 Inadequate MAIN/BLOCKSIZE ratio

72 Many SORTWKs and reallocation required

73 The dynamically allocated work data set does not allow seek head. Use SORTWKnn DD statement to specify work data set.

74 E61 specified with VLSHRT

System Action: None.

Programmer Response: None.

ICE802I t TECHNIQUE IN CONTROL

Explanation: Indicates the DFSORT technique that is currently in control. t is BLOCKSET (disk sort, standard merge, or copy), PEERVALE (disk sort), or CONVENTIONAL (nonstandard merge or tape work data set sort). This technique may not be the technique ultimately selected (see message ICE143I).

System Action: None.

Programmer Response: None.

ICE803I TOTAL DATA SET TRACKS ALLOCATED: a TRACKS USED: b

Explanation: Gives work data set space usage information for the sort run. Message values are as follows:

a Total primary and secondary space allocated (in tracks)

b Total primary and secondary space used (in tracks)

System Action: None.

Programmer Response: None.

ICE804I ddname EXCP COUNT: n

Explanation: Lists the total number of EXCPs performed to the specified input, output, or work data set.

System Action: None.

Programmer Response: None.

Note: If this message is printed for a data set for which the EXCP access method is not used, the count will be 0, to indicate that it is not meaningful.

ICE805I JOBNAME: jobname , STEPNAME: stepname

Explanation: Indicates jobname and stepname for DFSORT run.

System Action: None.

Programmer Response: None.

ICE806I DYNAMIC ALLOCATION REQUIRED a DATA SET(S), EACH WITH b BLOCKS OF c BYTES

Explanation: Gives required dynamic allocation work data set space for the sort run when DYNALLOC is in effect. Message values are as follows:

a Number of work data sets to be dynamically allocated

b Number of primary blocks for each work data set

c Size of blocks

System Action: None.

Programmer Response: None.

ICE807I 370-XA SORTING INSTRUCTIONS (ARE | ARE NOT) BEING USED

Explanation: Indicates whether the sorting instruction algorithm was used.

System Action: None.

Programmer Response: None.

Note: This message is only printed for a Blockset sort under MVS/XA.

ICE808I PHASE p: RSA ALLOCATED (BELOW | ABOVE | ABOVE AND BELOW) 16-MEGABYTE VIRTUAL

Explanation: This message indicates where the record save area (RSA), used by DFSORT to hold records in storage, was allocated during the run.

p is the phase number (1=input, 2=key, 3=output).

System Action: None.

Programmer Response: None.

Note: This message is only printed for a Blockset sort under MVS/XA.

ICE809I PHASE p: c BUFFERS ALLOCATED (BELOW | ABOVE | ABOVE AND BELOW) 16-MEGABYTE VIRTUAL

Explanation: This message indicates where the SORTIN (input), SORTOUT (output), or SORTWK (work) buffers were allocated during the run.

p is the phase number (1=input, 2=key, 3=output).

c is SORTIN, SORTOUT, or SORTWK and the message would appear once for each type of buffer that was allocated.

System Action: None.

Programmer Response: None.

Note: This message is printed for a Blockset sort under MVS/XA, if the buffers were allocated. For example, if all the input was through an E15 user

exit, no input buffers would be allocated and the message would not be printed for SORTIN.

The following messages (ICE820-ICE990) are diagnostic messages for use by IBM field support personnel. They do not require your response as a DFSORT user.

ICE820I RL=a B=b IL=c IS=d IB=e RM=f EM=g BA=h IX=j OX=k

ICE821I BN=a X=b TO=c SN=d G=e

ICE822I BN=a X=b G=c PN=d BT=e TO=f

ICE823I BN=a X=b G=c TO=d BT=e

ICE824I PE=a RP=b CX=c CO=d CO=e CR=f G=g WB=h

ICE825I GP=a SA=b X=c

ICE826I BN=a X=b RM=c

ICE891I a WMAIN, b CMAIN

ICE892I a RIN b BLI c BLO d RUN e BUN f CPU

ICE893I a XIN b WIN c GIN

ICE894I a STR b MOR c IPB d OPB e CYL

ICE895I a MUNIT b SUNIT c OUNIT

ICE896I a SET b DEXTOT c BLK d CSZ

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

ICE989I CLOCK - xx,xx,xx

ICE990I NO OF STRINGS PROD BY PH1 xxxxxxx

Option[1]	E18	E19	E38	E39
SYNAD	x	x	x	x
EXLST	x[2]	x	x	x
EROPT	x		x	
EODAD	x			
BSAM 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 applications.

Index

D

E

F

H

I

J

L

M

N

O

P

Q

T

U

V

DFSORT Application
Programming: Guide
SC33-4035-11

Reader's Comment Form

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should 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.

Please use pressure sensitive or other gummed tape to seal this form.

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

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®

DFSORT Application Programming: Guide

SC33-4035-11

File No. S370-33

Printed in U.S.A.

IBM®