IBM

DFSORT

SC33-4035-14

Application Programming: Guide

Release 11

DFSORT

SC33-4035-14

Application Programming: Guide

Release 11

Fifteenth Edition (December 1988)

This edition replaces and makes obsolete the previous edition, SC33-4035-13.

This edition applies to Release 11 of IBM DFSORT, Program Number 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 Changes" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any 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, 4300, and 9370 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 licensed program in this publication is not intended to state or imply that only IBM's program 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 Reader's Comment Form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publishing, P. O. Box 49023, San Jose, California, U.S.A. 95161-9023. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks Used In This Document

The following names have been adopted by IBM for trademark use and will be used throughout this publication:

Enterprise Systems Architecture/370™

hiperspace™

Hipersorting™

MVS/ESA™

MVS/XA™

MVS/SP™

About This Book

To the Reader

DFSORT Application Programming: Guide contains all the information you need to use DFSORT on a continuing basis. DFSORT is a program you use to sort, merge, or copy information. DFSORT processes your information by manipulating the individual records in your data set.

- When you *sort* records, you arrange them in a particular sequence, choosing an order more useful to you than the original one.
- When you *merge* records, you combine the contents of two or more previously-sorted data sets into one.
- When you *copy* records, you make an exact duplicate of each record in your data set.

You can use many options with these basic procedures to get the exact results you want.

The information in this book is designed to help you program DFSORT applications. The information is not to be used for any other purpose.

Who Should Read This Book

This book is a reference guide for anyone who needs to sort, merge, or copy records using DFSORT (Data Facility Sort) Licensed Program 5740-SM1. It contains everything you need to know to use DFSORT, and is arranged so you can locate specific information quickly and easily. It is not designed to teach you how to use DFSORT, but is targeted at programmers who already have a basic understanding of the program and need a task-oriented guide to its functions and options.

If you are a new user, then *Getting Started with DFSORT* is the first book you should read. *Getting Started with DFSORT* is a self-study guide that tells you what you need to know to begin using DFSORT quickly, with step-by-step examples and illustrations.

How to Use This Book

DFSORT Application Programming: Guide is a task-oriented reference manual. To retrieve information quickly, use the Table of Contents to locate headings about the procedures you want to use, or the problems you want to solve.

Read the next section, titled "What This Book Covers," for summaries of chapters and appendixes.

Use the Index to locate information about specific topics such as restrictions or commands, about general topics such as tasks or procedures, and about anything that might be covered in several places in the book.

Finally, remember that *DFSORT Application Programming: Guide* is part of a more extensive DFSORT library. Six additional books can help you use DFSORT more effectively:

Task	DFSORT Publication
Evaluating DFSORT	*DFSORT General Information*, GC33-4033
	DFSORT Benchmark Guide, GG24-3019
Installing DFSORT	*DFSORT Installation and Customization*, SC33-4034
Self-study	*Getting Started with DFSORT*, SC26-4109
Quick reference	*DFSORT Reference Summary*, SX33-8001
Diagnosing failures	*DFSORT Diagnosis Guide*, LY27-9531
Interpreting messages	*DFSORT Messages and Codes*, SC26-4525

See "Reading List" on page 389, for other related documents you can use to take advantage of all the options and facilities of the program.

What This Book Covers

This book is a task-oriented reference manual. The various sections present related information together, grouped by what you want to do. The first four chapters of the book explain what you need to know to invoke and use DFSORT's primary record processing functions. The remaining chapters explain more specialized features, and how to use DFSORT most effectively. The appendixes provide specific information about a variety of topics.

- Chapter 1, "Introducing DFSORT" on page 1, presents an overview of DFSORT, explaining what you can do with the program and how you invoke DFSORT processing. It discusses how DFSORT works, data set formats and limitations, and explains the defaults that might have been modified during installation at your site.
- Chapter 2, "Executing DFSORT with Job Control Language" on page 15, explains how to use job control language (JCL) to execute your DFSORT jobs. It explains how to code JOB, EXEC, and DD statements, and how you can use cataloged procedures and EXEC PARM options to simplify your JCL and override DFSORT defaults set during installation.
- Chapter 3, "Using DFSORT Program Control Statements" on page 53, presents the DFSORT control statements you use to sort, merge, and copy data. It explains how to filter your data so you work only with the records you need, and how to edit data by reformatting and summarizing records. It explains how to write statements that direct DFSORT to use your own routines during execution.
- Chapter 4, "Using Panels to Create and Submit DFSORT Jobs" on page 145, explains DFSORT Panels, a facility you use to construct, save, and submit DFSORT jobs in an on-line environment. This chapter also describes how you can use the on-line HELP facility to get information about DFSORT and using DFSORT Panels.
- Chapter 5, "Using Your Own Exit Routines" on page 181, describes how to use DFSORT's program exits to call routines of your own during program execution. You can write your own routines to delete, insert, alter, and

summarize records, and you can incorporate your own error recovery routines.

- Chapter 6, "Invoking DFSORT from a Program" on page 231, describes how you use a system macro instruction to initiate DFSORT processing from your own assembler program. It also lists specific restrictions on invoking DFSORT from PL/I and COBOL.
- Chapter 7, "Improving Efficiency" on page 247, recommends ways you can maximize DFSORT processing efficiency. This chapter covers the entire spectrum of improvements you can make, from which operating system is most appropriate, to designing individual jobs for efficient processing at your site.
- Chapter 8, "Sample Job Streams" on page 265, contains annotated example jobstreams for sorting, merging and copying records.
- Appendix A, "Calculating Storage Requirements" on page 311, explains main storage considerations, and how to estimate the amount of intermediate storage you might require when sorting data.
- Appendix B, "Using Extended Function Support (EFS)" on page 315, explains how to use the Extended Function Support (EFS) interface to tailor control statements, to handle user-defined data types and collating sequences, and to have DFSORT issue customized informational messages during execution.
- Appendix C, "Specification/Override of DFSORT Options" on page 353, is a series of tables you can use to find the order of override for corresponding options that are specified in different sources.
- Appendix D, "Data Format Examples" on page 373, gives examples of the assembled data formats used with IBM System 370.
- Appendix E, "EBCDIC and ISCII/ASCII Collating Sequences" on page 377, lists the collating sequences from low to high order for EBCDIC and ISCII/ASCII characters.
- Appendix F, "DFSORT Abend Processing" on page 383, describes the ESTAE recovery routine for processing abends, and the Checkpoint/Restart facility.

Summary of Changes

Release 11, December 1988

Performance enhancements to the Blockset technique include:

- Hipersorting, a new DFSORT capability that uses hiperspace available with MVS/ESA on Enterprise Systems Architecture/370.
- New internal algorithms and more extensive use of System/370-XA sorting instructions for MVS/XA and MVS/ESA (fixed-length records sorting only).
- Use of EXCPVR when sorting and copying data sets for MVS/XA and MVS/ESA.
- More efficient allocation of dynamic work data sets.

Functional enhancements include:

- New installation and run-time parameters are available to modify the use of:
  - Hipersorting
  - EXCPVR
  - Control interval access for VSAM data sets
  - DFSORT's ESTAE recovery routine.
- PARM options and DFSORT control statements can now be supplied to DFSORT from a single source data set for JCL and program-invoked DFSORT jobs.
- The INREC and OUTREC statements have been enhanced:
  - Character and hexadecimal string constants are now supported as separators.
  - Column alignment is now supported for separators and input fields.
  - The upper limit for the separation repetition factor is raised from 256 to 4095.
- The Conventional technique modules can be installed in a system library to eliminate the need for the SORTLIB DD statement.
- Duplicate DDNAMEs are ignored after the first-specified DDNAME.
- DDNAMEs SORTWK0 through SORTWK9 are now accepted and treated as duplicates of SORTWK00 through SORTWK09. DDNAMEs SORTIN0 through SORTIN9 are now accepted and treated as duplicates of SORTIN00 through SORTIN09.
- DFSORT no longer requires that the SORTINnn data set with the largest block size and record length be specified first for Blockset merge jobs.
- The maximum length for variable blocked spanned records is increased to 32767.
- Multivolume SORTWKnn data sets are now allowed, but only the first volume of each data set is used.

- Ability to handle more complex SUM applications with Blockset and Peerage/Vale.

Additional improvements to DFSORT include:

- The MINLIM default is raised from 200K to 440K.
- The minimum number of data sets Blockset uses for dynamic allocation is changed from 1 to 2.

Contents

Chapter 1. Introducing DFSORT

This chapter introduces IBM OS/VS DFSORT Licensed Program 5740-SM1, or DFSORT. DFSORT is a program you use to sort, merge, and copy information.

- When you *sort* records, you arrange them in a particular sequence, choosing an order more useful to you than the original one.
- When you *merge* records, you combine the contents of two or more previously-sorted data sets into one.
- When you *copy* records, you make an exact duplicate of each record in your data set.

(Merging records first requires that the input data sets be identically sorted for the information you will use to merge them, and in the same order required for output. You can merge from 2 to 16 different data sets at a time).

In addition to the three basic functions, you can perform other processing simultaneously:

You can control which records to keep in the final output data set of a DFSORT run by using INCLUDE and OMIT statements in your job. These statements work like filters, testing each record against criteria that you supply, and retaining only the ones you want for the output data set. For example, you might choose to work only with those records that have a value of "Kuala Lumpur" in the field reserved for office location. Or perhaps you want to leave out any record dated after 1987 if it also contains a value greater than 20 for the number of employees.

You can edit and reformat your records before or after other processing by using INREC and OUTREC statements. Use INREC and OUTREC to delete fields from your records, to rearrange the order of the fields within records, and to insert separators, such as blanks, zeros, or constants, before, between, or after fields. For example, you might want to create a data set containing financial data but without any of the names that were there originally. You are probably interested in improving efficiency; you can use INREC to remove unnecessary fields from your records before other processing, and save the CPU time required to manipulate them.

You can sum numeric information from many records into one with the SUM statement. For example, if you want to know the total amount of a yearly payroll, you can add the values for a field containing salaries from the records of all your employees.

You can control DFSORT functions with other control statements, by specifying alternate collating sequences, invoking user exit routines, overriding installation defaults, and so on.

You can direct DFSORT to pass control during execution to routines you design and write yourself. For example, you can write user exit routines to summarize, insert, delete, shorten, or otherwise alter records during processing (although you should remember that DFSORT already provides extensive editing capabilities through the INCLUDE, OMIT, INREC, OUTREC, and SUM control statements). You can write your own routines to correct I/O errors that DFSORT cannot handle, or to perform any necessary abnormal end-of-task operation before DFSORT terminates.

You can write an EFS (Extended Function Support) program to intercept DFSORT control statements and PARM options for modification prior to use by DFSORT.

How You Can Invoke DFSORT

You can invoke DFSORT processing in a number of ways, described in this book at the pages listed below:

- With an EXEC job control statement in the input stream using the name of the program or the name of a cataloged procedure. See Chapter 2, "Executing DFSORT with Job Control Language" on page 15.
- With a program written in basic assembler language using a system macro instruction. See Chapter 6, "Invoking DFSORT from a Program" on page 231.
- With programs written in either COBOL or PL/I with a special facility of the language. See the programmer's guide describing the compiler version available at your location.
- With interactive panels supported under ISPF and ISMF. See Chapter 4, "Using Panels to Create and Submit DFSORT Jobs" on page 145.

In this book. the term *JCL invoked* means that the DFSORT program is initiated by an EXEC job control statement. The term *dynamically invoked* means that the DFSORT program is initiated from another program written in assembler, COBOL, or PL/I.

How DFSORT Works

This section lists the operating systems supported by DFSORT, and how DFSORT uses control fields and collating sequences to sort, merge, and copy your records.

Operating Systems

DFSORT operates under your operating system, and must be initiated according to the appropriate conventions. The operating systems this release supports are:

- OS/VS2 MVS Release 3.8
- MVS/SP Version 1 Release 3 with MVS/370
- MVS/SP Version 2 Release 1 with MVS/XA
- MVS/SP Version 3 Release 1 with MVS/ESA

and subsequent releases.

DFSORT can execute under MVS/370 running as a guest operating system under VM/SP, VM/SP High Performance Option (HPO), VM/XA Migration Aid, VM/370, or VM/XA System Facility (SF). DFSORT can execute under MVS/XA or MVS/ESA running as a guest operating system under VM/XA Migration Aid, or VM/XA System Facility (SF).

DFSORT is compatible with all of the IBM processors supported by MVS/370, MVS/XA, or MVS/ESA, in addition to any device they support for program residence. It also operates with any device QSAM or VSAM uses for input or output.

If you use MVS/ESA, DFSORT can use Hipersorting, which uses hiperspace available on Enterprise Systems Architecture/370 to improve performance.

If you use MVS/ESA or MVS/XA, DFSORT can:

- Place certain modules and buffers above 16-megabyte virtual, leaving more space below 16-megabyte virtual for user programs
- Use System/370 Extended Architecture Sorting Instructions to improve performance when sorting fixed-length records
- Use EXCPVR to improve performance when sorting and copying.

On all MVS systems, DFSORT can:

- Replace the IEBGENER system utility with the more efficient DFSORT ICEGENER facility
- Dynamically allocate the required intermediate work space for sorting.

Control Fields and Collating Sequences

You use DFSORT to process your information by defining *control fields* to identify the information you want to sort or merge. When you think of the contents of your data sets, you probably think of names, dates, account numbers, or similar pieces of information useful to you. For example, when you sort your data sets, you might think of arranging your records in alphabetical order, by family name. By using the *byte position* and *length* (in bytes) of the portion of each record containing a family name, you can define it as a control field to manipulate with DFSORT.

DFSORT uses the control fields you define as keys in processing. A *key* is a concept, such as family name, that you have in mind when you design a record processing strategy for a particular application. A control field, on the other hand, is a discrete portion of a record that contains the text or symbols corresponding to that information, in a form that can be used by DFSORT to identify and sort or merge the records. For all practical purposes, you can think of keys as equivalent to the control fields DFSORT uses in processing.

When you want to arrange your records in a specific order, you specify one or more control fields of your records to use as keys. The sequence in which you list the control fields becomes the order of priority DFSORT uses to arrange your records. The first control field you specify is called the *major control field.* Subsequent control fields are called *minor control fields*, as in first, second, and third minor control fields, and so on. If two or more records have identical values for the first control field, they are arranged according to the values in the second. Records with identical values for the first and second are arranged according to the third, and so on, until a difference is found, or no more control fields are available.

Records with identical values for all the control fields specified retain their original input order or are arranged randomly, depending upon which of the two options, EQUALS or NOEQUALS, is in effect. You can direct DFSORT to retain the original input order for records with identical values for all control fields by specifying EQUALS.

Control fields may overlap, or be contained within other control fields (such as a three-digit area code, within a 10-digit telephone number). They do not need to be contiguous, but must be located within the first 4092 bytes of the record. DFSORT interprets the collected control fields of each record, arranged in order of priority, as a single *control word*, up to a maximum of 4092 bytes in length (see Figure 1).

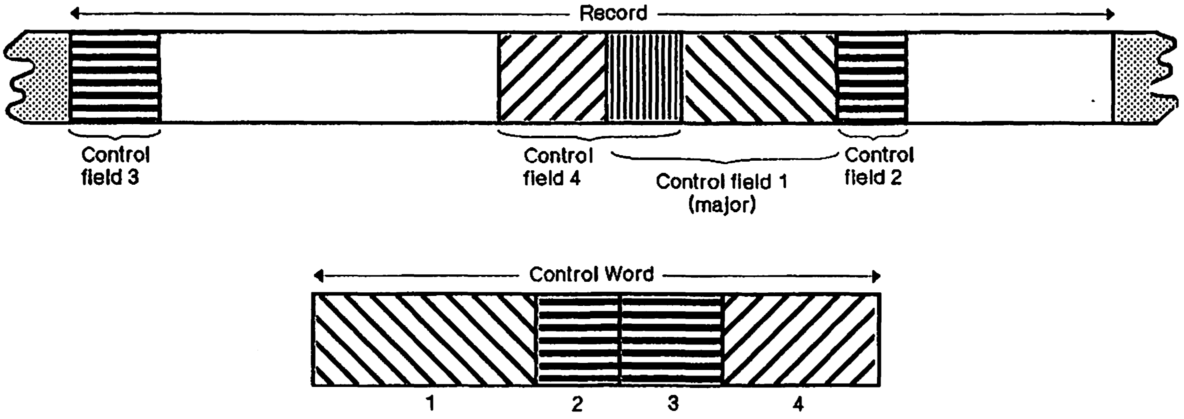


Figure 1. Control Fields

DFSORT contains several standard *collating sequences* you can use to determine how to arrange your records. Conceptually, a collating sequence is a specific arrangement of character priority used to determine which of two values in the same control field of two different records should come first. DFSORT uses EBCDIC, the standard IBM collating sequence, or the ISCII/ASCII collating sequence in sorting or merging records.

The collating sequence for character data and binary data is absolute; 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; each quantity is interpreted as having an algebraic sign.

You can modify the standard EBCDIC sequence to collate differently, if, for example, you want to allow alphabetic collation of national characters. An alternate collating sequence can be defined during installation with the ICEMAC ALTSEQ option, or you can define it yourself at execution with the ALTSEQ program control statement. You can also specify a modified collating sequence with an E61 user exit or with an Extended Function Support (EFS) program.

Program Execution

Unless you use DFSORT Panels to prepare and submit your job, (see Chapter 4, "Using Panels to Create and Submit DFSORT Jobs" on page 145), you must prepare job control language (JCL) statements and DFSORT program control statements to invoke DFSORT processing. JCL statements (see Chapter 2, "Executing DFSORT with Job Control Language" on page 15) are processed by your operating system. They describe your data sets to the operating system, and initiate DFSORT execution. DFSORT program control statements (see Chapter 3, "Using DFSORT Program Control Statements" on page 53) are processed by the DFSORT program. They describe the functions you want to perform, and invoke the processing you specify.

A sort usually requires intermediate storage as working space during program execution. Unless you specify dynamic allocation with the DYNALLOC facility, you must use JCL data definition (DD) statements to indicate the intermediate storage device or devices to use, and the amount of work space required for any sort run. Methods for determining how much space to allocate are explained in Appendix A, "Calculating Storage Requirements" on page 311. Merge or copy jobs do not require intermediate storage.

Figure 2 illustrates the processing order for record handling exits, statements, and options. Use this diagram with the text following it to understand the order DFSORT uses to execute your job.

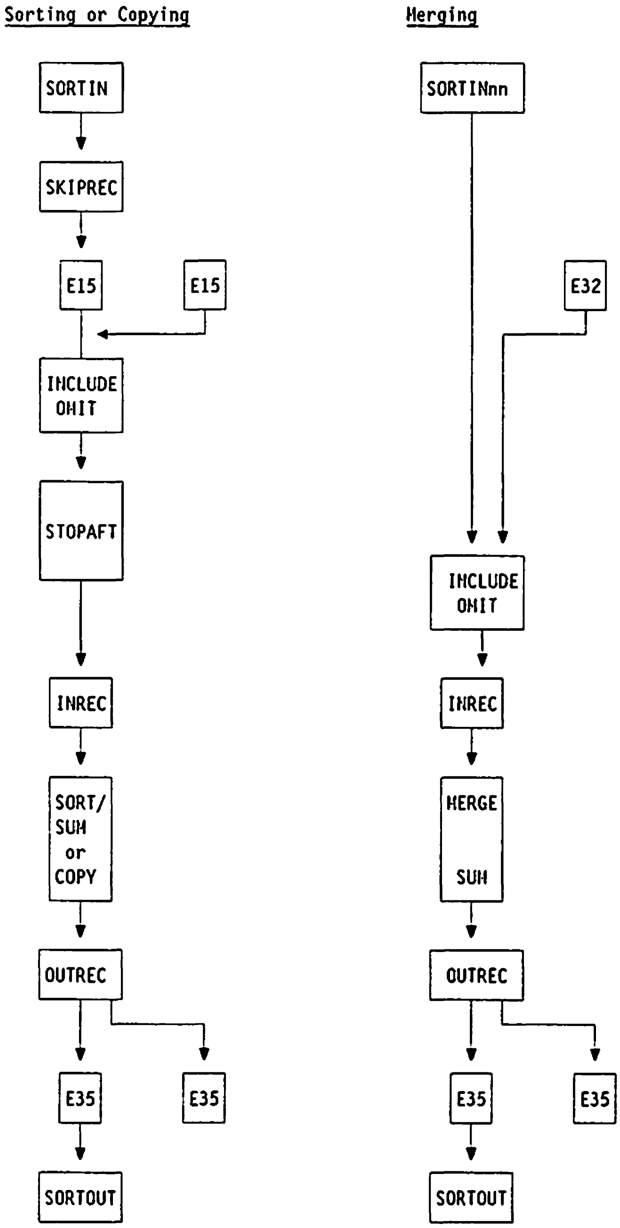


Figure 2. Record Processing Order

As shown in Figure 2, DFSORT processing follows this order:

1. DFSORT first checks for whether you supplied a SORTIN data set for SORT and COPY jobs, and SORTINnn data sets for MERGE jobs. If so, DFSORT reads the input records from them.

   - If no SORTIN data set is present for a SORT or COPY job, you must use an E15 exit to insert all the records. (This is also the case if you invoke DFSORT from a program with the address of an E15 exit in the parameter list, because SORTIN will be ignored.) DFSORT can use a COBOL E15 routine if you specified the E15 exit in the MODS statement.

   - If no SORTINnn data sets are present for a MERGE job, you must use an E32 exit to insert all the records.

2. If input records for SORT or COPY jobs were read from a SORTIN data set, DFSORT performs processing specified with the SKIPREC option. DFSORT deletes records until the SKIPREC count is satisfied. Eliminating records before SORT or COPY processing gives better performance.

3. If the input records for a SORT or COPY job were read from a SORTIN data set, DFSORT checks whether you specified a user exit routine at an E15 exit. If so, DFSORT transfers control to the user exit routine. You can use a COBOL E15 routine if the E15 exit is specified in the MODS statement. The E15 routine can insert, delete, or reformat records.

4. DFSORT performs processing specified on an INCLUDE or OMIT statement. If you used an E15 user exit routine to modify the record format, then the INCLUDE/OMIT control field definitions you specify must apply to the current format rather than to the original format. If you use the INCLUDE or OMIT statements to delete unnecessary records before SORT, MERGE, or COPY processing, your jobs will run more efficiently.

5. For SORT or COPY jobs, DFSORT performs processing specified with the STOPAFT option. Record input stops after the maximum number of records (n) you specify have been accepted. DFSORT accepts records for processing if they are:

   - Read from SORTIN or inserted by E15
   - Not deleted by SKIPREC
   - Not deleted by E15
   - Not deleted by INCLUDE/OMIT.

6. DFSORT performs processing specified in an INREC statement. If you changed record format before this step, the INREC control and separation field definitions you specify must apply to the current format rather than to the original format. Use INREC to shorten records before further processing for better performance.

7. DFSORT performs processing specified in the SORT, MERGE, or OPTION COPY statement.

   - For SORT, all input records are processed before any output record is processed.

   - For COPY or MERGE, an output record is processed after an input record is processed.

   - For SORT or MERGE, if a SUM statement is present, DFSORT processes it during the SORT or MERGE processing. DFSORT summarizes the records and deletes duplicates as soon as possible for better performance. If you made any changes to the record format prior to this step, the SORT or MERGE, and SUM field definitions you specify must apply to the current format rather than to the original format.

8. DFSORT performs processing specified in an OUTREC statement. If you changed record format prior to this step, the OUTREC control or separation field definitions must apply to the current format rather than to the original format.
9. If an E35 exit is present, DFSORT transfers control to your user exit routine after all of the statement processing is completed. If you changed record format, the E35 exit receives the records in the current format rather than in the original format. You can use a COBOL E35 routine if you specify the E35 exit in the MODS statement. You can use the E35 exit routine to add, delete, or reformat records.

   If a SORTOUT data set is not present, the E35 exit must dispose of all the records, because DFSORT treats 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.)
10. DFSORT writes your records to the SORTOUT data set, if present.

Data Set Considerations

You must define any data sets you provide for DFSORT according to the conventions your operating system requires. You can use the label checking facilities of the operating system during DFSORT execution. See *OS/VS2 MVS Supervisor Services and Macro Instructions* for details.

Unless you use DFSORT Panels to create and submit your jobs, you must describe all data sets (except those allocated with the DYNALLOC parameter) in job control data definition (DD) statements. You must place the DD statements in the operating system input stream with the job step that allocates DFSORT execution.

DFSORT Panels operates in two modes, foreground and background. Background mode creates DFSORT jobs containing the job control language (including DD statements) already coded in the DFSORT Panels user profile. This JCL is the same as that which you code yourself. Foreground mode uses CLIST processing instead of JCL, so if you choose this technique you do not need JCL at all. See Chapter 4, "Using Panels to Create and Submit DFSORT Jobs" on page 145 for more information.

When You Sort or Copy Records

Input to a sort or copy run can be a blocked or unblocked QSAM or VSAM data set, containing fixed or variable-length records. You can concatenate QSAM input data sets even if they are on dissimilar devices. See "SORTIN DD Statement" on page 42 for the restrictions that apply.

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

When You Merge Records

Input to a merge run can be up to 16 blocked or unblocked QSAM or VSAM data sets, containing fixed or variable-length records. Your input data sets can be either QSAM or VSAM, but not both. The records in all input data sets must already be sorted in the same order as that required for output.

Output from a merge run can be a blocked or unblocked QSAM or VSAM data set, regardless of whether the input is QSAM or VSAM. The output data set however, must be the same type (fixed or variable) as the input data sets.

Data Set Notes and Limitations

General Considerations

Your records can be EBCDIC, ISCII/ASCII, Japanese, and data types you define yourself. To process Japanese data types with DFSORT, you can use the IBM Double Byte Character Set Ordering Support Program (DBCS Ordering), Licensed Program 5665-360, Release 2.0.

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

The maximum record length DFSORT can handle depends on the amount of main storage available. Record length can never exceed the maximum record length you specify. Variable-length records are limited to 32756 bytes. Fixed-length records are limited to 32760 bytes. Variable block-spanned records are limited to 32767 bytes.

The number of records that can be sorted using a given amount of storage is reduced by:

- Processing control fields of different formats
- Large numbers of control fields
- Large numbers of intermediate data sets.

Providing an Extended Function Support program with an EFS01 exit routine may limit the record length that can be used when processing variable-length records.

Minimum block length for tape work data sets is 18 bytes; the minimum record length is 14 bytes.

The Blockset technique pads or truncates fixed-length records when the output data set record length is different from that of the input data set. Record padding and truncation can occur with Blockset sort and copy operations. Records are not padded when using Blockset merge or other sort techniques (unless an E15 or E35 exit, or an INREC or OUTREC operation is present). If you use user exits to modify records, you must ensure that the exit routines pad and truncate correctly. DFSORT pads records on the right, with binary zeros.

For more information about Blockset, see "Specify Efficient Sort/Merge Techniques" on page 249.

QSAM Considerations

- If you use DSN=NULLFILE on your DD statement for an input data set, a system restriction prevents DFSORT from using the EXCP access method.
- You can use empty input data sets.
- If any of the input data sets are on tape without standard labels, you must specify DCB parameters on their DD statements.
- ISO/ANSI Version 1 tape files can only be used as input; never as output.

VSAM Considerations

- 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.
- You can use empty input data sets in sort or copy jobs, but not for merge jobs. If an input data set to a merge job is empty, VSAM returns an open error code (160) and DFSORT terminates.
- Data sets cannot be concatenated.
- If output is a keyed-sequential data set (KSDS), the key must be the first control field (or the key fields must be in the same order as the first 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 5, "Using Your Own Exit Routines" on page 181.
- You must build the VSAM exit list with 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 bytes greater than the maximum record size defined in the cluster. Variable-length records have a record descriptor word (RDW) field 4 bytes long at the beginning of each record, but VSAM records do not. The record size defined in the VSAM cluster is therefore 4 bytes less than the non-VSAM LRECL. DFSORT adds 4 bytes for the RDW when processing the record. If you use VSAM for both input and output, these 4 bytes are removed. 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
```

- If a non-empty VSAM SORTOUT data set was defined without the reuse option, VSAM does not open the data set and issues OPEN error IEC161I RC84 error code 232(E8):

```
Reset was specified for a nonreusable data set and the data set
is not empty.
```

After receiving the OPEN return code, DFSORT reopens the data set for update and one of two things happens:

- For an ESDS, DFSORT adds records at the end of the data set.
- For a KSDS, DFSORT inserts records in key order.

Installation Defaults

Some DFSORT default values depend on specifications your system programmer set with the ICEMAC macro when DFSORT was installed. i2.installation defaults This book assumes DFSORT was installed at your site with the defaults it was delivered with. *DFSORT Installation and Customization* contains planning considerations and general information about installing DFSORT. Step-by-step installation procedures are listed in the *DFSORT Program Directory*.

The following parameters and functions can be set during installation. See your system programmer if you need to know what values were used.

Parameters **Function**

ABCODE Specifies the ABEND code used when DFSORT abends for a critical error.

ALTSEQ Alters the normal EBCDIC collating sequence.

ARESALL Specifies, for MVS/XA or MVS/ESA, the number of bytes reserved above 16-megabyte virtual for system use.

ARESINV Specifies, for MVS/XA or MVS/ESA, the number of bytes reserved above 16-megabyte virtual for the invoking program when DFSORT is dynamically invoked.

CHALT Translates format CH as well as format AQ, or translates format AQ only.

CHECK Determines whether record count checking is suppressed for applications that use the E35 user exit routine without a SORTOUT data set.

CINV Determines whether DFSORT can use control interval access for VSAM data sets.

COBEXIT Determines whether the E15 and E35 routines are executed with the VS COBOL II libraries.

DYNALOC Specifies the default values for device name and number of work data sets to be dynamically allocated. These default values are used in conjunction with the ICEMAC option DYNAUTO=YES and execution option DYNALLOC.

DYNAUTO Determines whether work data sets are dynamically allocated automatically when no SORTWKnn DD statement is present.

EFS Specifies the name of a user-written Extended Function Support program to be called by DFSORT.

EQUALS Preserves the input order of equally collating records.

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

ESTAE Determines whether DFSORT should delete its ESTAE recovery routine early or use it for the entire run.

EXCPVR Determines whether DFSORT should use EXCPVR when reading and writing SORTIN, SORTOUT, and SORTWKnn data sets.

EXITCK Determines whether DFSORT should terminate or continue when it receives certain invalid return codes from E15 or E35 exit routines.

GENER Specifies the name of the IEBGENER system utility module to be used by ICEGENER (DFSORT's facility for IEBGENER jobs).

HIPRMAX Specifies the amount of hiperspace to use for Hipersorting on an MVS/ESA system.

IEXIT Determines whether DFSORT is to pass control to your site's ICEIEXIT routine.

IGNCKPT Determines 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 Determines whether the specified JCL defaults are to be used when DFSORT is JCL invoked or dynamically invoked. One of these must immediately follow ICEMAC and be followed, in turn, by other options, if desired.

LIST Determines whether to print DFSORT program control statements.

LISTX Determines whether to print control statements that have been returned by an Extended Function Support program.

MAXLIM Sets an upper limit to the amount of address space available when SIZE/MAINSIZE=MAX.

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

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.

NOMSGDD Determines whether DFSORT should terminate or continue when the message data set is required, but not present.

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

OUTSEC Determines 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.

PARMDDN Specifies an alternate DDNAME for the DFSORT DFSPARM data set.

RESALL Reserves storage for system and application use.

RESINV Reserves space for programs invoking DFSORT.

SIZE Sets maximum amount of main storage.

SMF Produces SMF records.

SORTLIB Determines whether DFSORT searches a system or private library for the modules used with a tape work data set sort or Conventional merge.

STIMER Determines whether DFSORT can use the STIMER macro. If DFSORT does not use the STIMER macro, processor timing data will not appear in SMF records or in the ICETEXIT statistics.

SVC Specifies a user SVC number for DFSORT and allows an installation to use two different releases of DFSORT at the same time by using two different DFSORT SVC modules in conjunction with SVC 109.

TEXIT Determines whether DFSORT passes control to your site's ICETEXIT routine.

TMAXLIM Specifies, for MVS/XA or MVS/ESA, an upper limit to the total amount of main storage above and below 16-megabyte virtual available to DFSORT.

VERIFY Verifies sequence of output records.

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

VLSHRT Allows DFSORT to continue sorting or merging when it encounters a variable length record not long enough to contain all specified control fields.

WRKREL Determines whether unused temporary SORTWK data set space is released.

WRKSEC Determines whether DFSORT uses automatic secondary allocation for temporary SORTWK data sets.

ZDPRINT Determines whether DFSORT produces printable numbers from positive ZD fields that result from summarization; that is, whether the zone of the last digit should be changed from a hexadecimal C to a hexadecimal F.

Tables showing all the possible sources of specification and order of override for each option are shown in Appendix C, "Specification/Override of DFSORT Options" on page 353.

DFSORT Messages and Return Codes

You can determine, during installation or execution, whether DFSORT writes messages to the message data set, to the master console, or both. You can also direct an Extended Function Support program to write messages to the message data set.

Messages written to the message data set can be either critical error messages, informational error messages, or diagnostic messages, as determined during installation or execution.

Messages written to the master console can be either critical error messages or informational error messages, as determined during installation.

See *DFSORT Messages and Codes* for complete information about DFSORT messages and a discussion of DFSORT return codes.

Chapter 2. Executing DFSORT with Job Control Language

Your operating system uses the job control language (JCL) you supply with your DFSORT program control statements to:

- Identify you as an authorized user
- Allocate the necessary resources to execute your job
- Execute your job
- Return information to you about the results
- Terminate your job.

Unless you create your jobs with the interactive DFSORT Panels facility (see Chapter 4, "Using Panels to Create and Submit DFSORT Jobs" on page 145), you must supply job control language statements with every DFSORT job you submit.

Required JCL includes a JOB statement, an EXEC statement, and several DD statements. The statements you need and their exact form depend upon whether you:

- Invoke DFSORT with an EXEC statement in the input job stream, or with a system macro instruction within another program.
- Choose to use EXEC statement cataloged procedures to invoke DFSORT.
- Choose to specify various DFSORT control statements or PARM options.
- Want to use program exits to activate routines of your own.
- Want to use dynamic link-editing.
- Want to see diagnostic messages.

DFSORT Panels offers you an alternative to coding JCL directly. When you use panels to prepare a job to be executed or saved in a data set, much of the required JCL can be supplied automatically from the contents of the DFSORT User Profile. DFSORT jobs you prepare for submission in foreground under TSO use CLIST processing rather than JCL. See Chapter 4, "Using Panels to Create and Submit DFSORT Jobs" on page 145 for details on using DFSORT Panels.

The JCL statements and their functions are listed below. Details on coding the individual statements are presented in subsequent sections.

JCL Statement	Description
//JOBLIB DD	Defines your program link library if it is not already known to the system.
//STEPLIB DD	Same as //JOBLIB DD
//SORTLIB DD	Defines the data set that contains special load modules, if it is not already known to the system.
//SYSOUT DD[1]	Defines the message data set.
//SORTIN DD[1]	Defines the input data set for a sort or copy.
//SORTINnn DD[1]	Defines the input data sets for a merge.
//SORTOUT DD[1]	Defines the output data set for a sort, merge, or copy.
//SORTWKnn DD[1]	Defines intermediate storage data sets for a sort.

//DFSPARM DD[1] Contains DFSORT PARM options and program control statements.

//SYSIN DD Contains DFSORT program control statements.

//SORTCNTL DD[1] Same as //SYSIN DD

//SORTDIAG DD Specifies that all messages and program control statements be printed.

//SORTCKPT DD Defines the data set for checkpoint records.

//SYSUDUMP DD Defines the data set for output from a system ABEND dump routine.

//SYSABEND DD Same as //SYSUDUMP DD

//SORTSNAP DD Defines the snap dump data set dynamically allocated by DFSORT.

//ddname Defines the data set containing exit routines (as specified in the MODS program control statement).

The following DD statements are only necessary for dynamic link-editing of exit routines.

//SYSPRINT DD Defines the message data set for the linkage editor.

//SYSUT1 DD Defines the intermediate storage data set for the linkage editor.

//SYSLIN DD Defines the data set for control information for the linkage editor.

//SYSLMOD DD Defines the data set for output from the linkage editor.

//SORTMODS DD Defines the temporary partitioned data set for exit routines from SYSIN.

[1] These are the default DDNAMEs with which DFSORT was delivered. SYSOUT and DFSPARM might have been changed during DFSORT installation. You can change all of the indicated DDNAMEs yourself at run time. For override information, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

Notational Conventions

The syntax diagrams in this book are designed to make coding DFSORT program control statements simple and unambiguous. The lines and arrows represent a path or flowchart that connects operators, parameters, and delimiters in the order and syntax in which they must appear in your completed statement. Construct a statement by tracing a path through the appropriate diagram that includes all the parameters you need, and code them in the order that the diagram requires you to follow. Any path through the diagram gives you a correctly coded statement, if you observe these conventions:

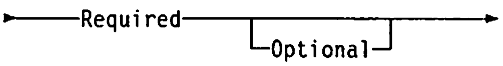
- Read the syntax diagrams from left to right and from top to bottom.
- Begin coding your statement at the spot marked with the double arrowhead.

▶▶────

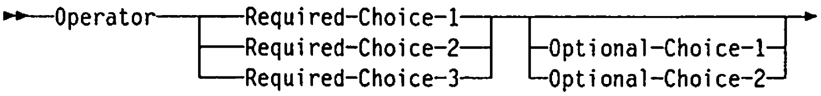
- A single arrowhead at the end of a line indicates that the diagram continues on the next line or at an indicated spot.

A continuation line begins with a single arrowhead.

- Strings in upper-case letters, and punctuation (parentheses, apostrophes, and so on), must be coded exactly as shown.
  - Semicolons are interchangeable with commas in program control statements and the EXEC PARM string. For clarity, only commas are shown in this book.
- Strings in all lowercase letters represent information that you supply.
- Required parameters appear on the same horizontal line (the main path) as the operator, while optional parameters appear in a branch below the main path.

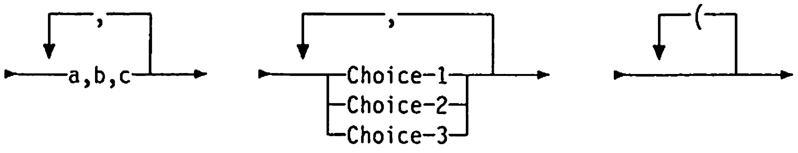


- Where you can make one choice between two or more parameters, the alternatives are stacked vertically.



If one choice within the stack lies on the main path (as in the example above, left), you *must* specify one of the alternatives. If the stack is placed below the main path (as in the example above, right), then selections are optional, and you can choose either one or none of them.

- The repeat symbol shows where you can return to an earlier position in the syntax diagram to specify a parameter more than once (see the example below, left), to specify more than one choice at a time from the same stack (see the example below, middle), or to nest parentheses (see the example below, right).



Do not interpret a repeat symbol to mean that you can specify incompatible parameters. For instance, do not specify both **ABEND** and **NOABEND** in the same **EXEC** statement, or attempt to nest parentheses incorrectly.

Use any punctuation or delimiters that appear within the repeat symbol to separate repeated items.

- A double arrowhead at the end of a line indicates the end of the syntax diagram.

```
──►◄
```

Using the JOB Statement

The JOB statement is the first JCL statement of your job. It must contain a valid job name in the name field, and the word JOB in the operation field. All parameters in the operand field are optional, although your site might have made information such as account number and the name of the programmer mandatory:

```
//jobname JOB accounting information, programmer's name, and so on
```

Using the EXEC Statement

The EXEC statement is 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 can also specify DFSORT options on the EXEC statement.

The format of the EXEC statement is:

```
►►─//stepname EXEC─┬──PGM=┬SORT──┬──┬┬────────────────────┬──────────────►
                   │      └ICEMAN┘  ││                    │
                   ├──PROC=┬SORT─┬──┤│     ┌─,──┐         │
                   │       └SORTD┘  ││     ▼    │         │
                   └─┬SORT──┬───────┘└,PARM='─options┴'───┘
                     └SORTD┘

►─────────────────────────────────┬──────────────────────┬─────────────►◄
                                  │ ┌──────────────────┐ │
                                  │ ▼                  │ │
                                  └─,─other-parameters─┴─┘
```

If you use a cataloged procedure (discussed in detail below), specify PROC=SORT or PROC=SORTD. You can omit PROC= and specify simply SORT or SORTD. However, PROC= can remind you that a cataloged procedure is being used.

If you do not use a cataloged procedure, use PGM= either with the actual name of the sort module (ICEMAN) or with its alias, SORT. Be sure that the alias has not been changed at your site.

Specifying EXEC Statement Cataloged Procedures

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 called the procedure library. Two cataloged procedures are supplied with the program: SORT and SORTD. Specify them in the first parameter of the EXEC statement by PROC=SORT, PROC=SORTD, or simply SORT or SORTD.

SORT Cataloged Procedure

You can use the supplied SORT cataloged procedure when you include user routines that require link-editing. Using 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 a private data set containing the modules needed for a sort using tape work files or a merge using 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.

If the modules were installed in a system library and ICEMAC SORTLIB=SYSTEM is used, then the SORTLIB DD statement is unnecessary and is ignored.

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 a private 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.

If the modules were installed in a system library and ICEMAC SORTLIB=SYSTEM is used, then the SORTLIB DD statement is unnecessary and is ignored.

30 Directs messages to system output class A.

Specifying EXEC Statement PARM Options

When you invoke DFSORT with JCL, you can specify some DFSORT options on the PARM parameter of the EXEC statement. These options include EFS, LIST, NOLIST, LISTX, NOLISTX, MSGPRT, and MSGDDN, which are ignored if specified in an OPTION statement in SYSIN. Full override and applicability details are listed in Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: If you use the DFSPARM DD statement instead, you can specify both EXEC PARM options and DFSORT control statements in a single source data set that overrides all other sources. See "DFSPARM DD Statement" on page 49.

```
              ┌─────────────────────────────,──────┐
              ↓                                    │
►►──,PARM='─┬─┬─ABEND───┬──────────────────────────┴─'──────────►◄
            │ └─NOABEND─┘                          │
            ├─ARESALL=─┬─n──┬──────────────────────┤
            │          └─nK─┘                      │
            ├─BSAM─────────────────────────────────┤
            ├─┬─CINV───┬───────────────────────────┤
            │ └─NOCINV─┘                           │
            ├─DYNALLOC─┬────────────────────┬──────┤
            │          └─=─┬─d──────────────┘      │
            │              ├─(d)────┤              │
            │              ├─(,n)───┤              │
            │              ├─(d,n)──┤              │
            │              ├─OFF────┤              │
            │              └─(OFF)──┘              │
            ├─EFS=─┬─name─┬────────────────────────┤
            │      └─NONE─┘                        │
            ├─┬─EQUALS───┬─────────────────────────┤
            │ └─NOEQUALS─┘                         │
            ├─EXCPVR=─┬─ALL───┬────────────────────┤
            │         ├─NOWRK─┤                    │
            │         └─NONE──┘                    │
            ├─E15=COB──────────────────────────────┤
            ├─E35=COB──────────────────────────────┤
            ├─FILSZ=─┬─x──┬────────────────────────┤
            │        └─Ex─┘                        │
            ├─HIPRMAX=─┬─OPTIMAL─┬─────────────────┤
            │          └─n───────┘                 │
            ├─┬─LIST───┬───────────────────────────┤
            │ └─NOLIST─┘                           │
            ├─┬─LISTX───┬──────────────────────────┤
            │ └─NOLISTX─┘                          │
            ├─MSGDDN=ddname────────────────────────┤
            ├─MSGPRT=─┬─ALL──────┬─────────────────┤
            │         ├─CRITICAL─┤                 │
            │         └─NONE─────┘                 │
            ├─┬─OUTREL───┬─────────────────────────┤
            │ └─NOOUTREL─┘                         │
            ├─RESALL=─┬─n──┬───────────────────────┤
            │         └─nK─┘                       │
            ├─SIZE=─┬─n──────┬─────────────────────┤
            │       ├─nK─────┤                     │
            │       ├─MAX────┤                     │
            │       ├─MAX-m──┤                     │
            │       └─MAX-mK─┘                     │
            ├─SKIPREC=z────────────────────────────┤
            ├─┬─STIMER───┬─────────────────────────┤
            │ └─NOSTIMER─┘                         │
            ├─STOPAFT=n────────────────────────────┤
            ├─┬─WRKREL───┬─────────────────────────┤
            │ └─NOWRKREL─┘                         │
            └─┬─WRKSEC───┬─────────────────────────┘
              └─NOWRKSEC─┘
```

ABEND or NOABEND

```
►►──────────┬ABEND──┬──────────────────────────────────────►◄
            └NOABEND┘
```

Temporarily overrides the ERET installation option, which specifies whether DFSORT abends or terminates with a return code of 16 if your sort, copy, or merge is unsuccessful.

ABEND
specifies that if your sort, copy, or merge is unsuccessful, it abends with a user completion code equal to the appropriate message number, or with a user-defined number between 1 and 99, as set during installation with the ICEMAC option ABCODE = n.

When DEBUG ABEND is in effect, a user abend code of zero might be issued when a tape work data set sort or Conventional merge is unsuccessful.

NOABEND
specifies that an unsuccessful sort, copy, or merge terminates with a return code of 16.

Note: RC16 = ABE and NORC16 can be used instead of ABEND and NOABEND, respectively.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options."

ARESALL

```
►►──────────ARESALL=─┬─n──┬──────────────────────────────►◄
                     └─nK─┘
```

For MVS/XA and MVS/ESA, temporarily overrides the ARESALL installation option, which specifies the number of bytes to be reserved above 16-megabyte virtual for system use. For more information, see the discussion of the ARESALL parameter in "OPTION Control Statement" on page 97.

n
n specifies the number of bytes of storage to be reserved

Limit: 8 digits.

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

Limit: 5 digits.

Note: RESERVEX can be used instead of ARESALL.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options."

BSAM

```
▸▸────────BSAM──────────────────────────────────────────▸◂
```

For disk processing the EXCP access method is normally used for SORTIN and SORTOUT. If you encounter a problem related to the use of EXCP, you can temporarily bypass it by specifying BSAM.

This option is ignored if VSAM SORTIN or SORTOUT data sets are specified.

Default: None; optional. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

CINV or NOCINV

```
▸▸──────────┬─CINV───┬──────────────────────────────────▸◂
            └─NOCINV─┘
```

Temporarily overrides the CINV installation option, which specifies whether DFSORT can use control interval access for VSAM data sets. For more information, see the explanation of the CINV parameter in "OPTION Control Statement" on page 97.

CINV
: directs DFSORT to use control interval access when possible for VSAM data sets.

NOCINV
: directs DFSORT not to use control interval access.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

DYNALLOC

```
►►──────────DYNALLOC─┬──────────────┬──────────────────────────►◄
                     └=─┬─d──────┬──┘
                        ├─(d)────┤
                        ├─(,n)───┤
                        └─(d,n)──┘
```

Use this parameter to assign DFSORT the task of dynamically allocating needed work space. You do not need to calculate and use JCL to specify the amount of intermediate work space needed by the program. DFSORT uses the dynamic allocation facility of the operating system to allocate work space to get the best possible performance for the newest application.

For more information, see the discussion of DYNALLOC in "OPTION Control Statement" on page 97.

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 maximum number of requested work data sets. The maximum value of n is 16; if you specify more than 16, 16 is used. If you specify 1 and the Blockset technique is selected, 2 is used.

Default: None; optional. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

DYNALLOC=OFF

```
►►──────────DYNALLOC=─┬─(OFF)─┬──────────────────────────────►◄
                      └─OFF───┘
```

Directs DFSORT *not* to allocate intermediate workspace dynamically, overriding the ICEMAC installation option DYNAUTO=YES, or the DYNALLOC parameter (without OFF) specified at execution. For more information, see the discussion of the DYNALLOC option in "OPTION Control Statement" on page 97.

OFF
directs DFSORT not to allocate intermediate workspace dynamically.

Default: None; optional. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

EFS

```
►►────────EFS=┬─name─┬──────────────────────────────►◄
              └─NONE─┘
```

Temporarily overrides the EFS installation option, which specifies whether DFSORT is to pass control to an Extended Function Support program. See Appendix B, "Using Extended Function Support (EFS)" on page 315 for more information on EFS

name
the name of the EFS program that will be called to interface with DFSORT.

NONE
means no call will be made to the EFS program.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

EQUALS or NOEQUALS

```
►►────────┬EQUALS───┬───────────────────────────────►◄
          └NOEQUALS─┘
```

Temporarily overrides the EQUALS installation option, which specifies whether the original sequence of identical collating records for a sort or a merge should be preserved from input to output. For more information, see the discussion of the EQUALS option in "OPTION Control Statement" on page 97.

EQUALS
specifies that the original sequence must be preserved.

NOEQUALS
specifies that the original sequence need not be preserved.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

EXCPVR

```
►►──EXCPVR=┬ALL───┬─────────────────────────────────►◄
           ├NOWRK┤
           └NONE──┘
```

For MVS/XA and MVS/ESA, temporarily overrides the EXCPVR installation option, which specifies the data sets for which DFSORT can use EXCPVR when sorting or copying fixed- and variable-length records with the BLOCKSET technique. For more information, see the explanation of the EXCPVR parameter in "OPTION Control Statement" on page 97.

Note: The EXCPVR option is not supported for MVS/370.

ALL
specifies that DFSORT can use EXCPVR for SORTIN, SORTOUT, and SORTWKnn data sets.

NOWRK
specifies that DFSORT can use EXCPVR for the SORTIN and SORTOUT data sets only. DFSORT will not use EXCPVR for SORTWKnn if NOWRK is specified.

NONE
specifies that DFSORT cannot use EXCPVR for SORTIN, SORTOUT, or SORTWKnn data sets.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

E15=COB

```
►►──────────E15=COB──────────────────────────────────────►◄
```

Specifies that your E15 routine is written in COBOL, and temporarily overrides the MODS statement for E15. If you specify E15=COB but do not identify an E15 module with a MODS statement, the E15=COB is ignored.

Default: None; optional. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

E35=COB

```
►►──────────E35=COB──────────────────────────────────────►◄
```

Specifies that your E35 routine is written in COBOL, and temporarily overrides the MODS statement for E35. If you specify E35=COB but do not identify an E35 module with a MODS statement, the E35=COB is ignored.

Default: None; optional. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

FILSZ

```
►►──────────FILSZ=─┬─x──┬──────────────────────────────────────────────►◄
                   └─Ex─┘
```

This parameter specifies (1) an exact number of records for a sort or merge, or (2) an estimated number of records for a sort. Using *exact* values forces 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, it uses the estimated FILSZ value to aid optimization of intermediate storage; otherwise, it does not use the value for this purpose. For variable-length records, DFSORT uses the length of the longest input record to determine the total amount of work space to allocate. DFSORT uses the estimated FILSZ (if specified) for optimization under these circumstances:

- An E15 exit routine is used.
- Input data sets are multivolume or on tape.
- Input data sets are concatenated and Blockset is not selected.

x

specifies 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.

Ex

specifies an estimated number of records to be sorted. This value must be preceded by a letter E, and be large enough to include the SORTIN data set, and any records you might add at exit E15.

For example, if you estimate your total data set size to be 5000 records, specify FILSZ=E5000.

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

Default: None; optional. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

HIPRMAX

```
►►──HIPRMAX=─┬─OPTIMAL─┬──────────────────────────────────────────────►◄
             └─n───────┘
```

For MVS/ESA, temporarily overrides the HIPRMAX installation option, which specifies the maximum amount of hiperspace to be committed for Hipersorting. For more information on this parameter, see the explanation of the HIPRMAX parameter ("OPTION Control Statement" on page 97).

OPTIMAL
specifies that DFSORT determines dynamically the maximum amount of hiperspace to be committed for Hipersorting. DFSORT determines the paging activity at the start of the run and selects the maximum value that causes minimal system impact.

n
specifies the maximum megabytes of hiperspace to be committed for Hipersorting. n must be a value between 0 and 9999. The actual amount of hiperspace used will not exceed the HIPRMAX value, but might be less if DFSORT determines that using the maximum will increase system paging significantly. If n is 0 (zero), Hipersorting is not used.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

LIST or NOLIST

```
►►──────────┬─LIST───┬──────────────────────────────────────────►◄
            └─NOLIST─┘
```

Temporarily overrides the LIST installation option, which specifies whether DFSORT program control statements should be written to the message data set. See *DFSORT Messages and Codes* for full details on use of the message data set.

LIST
specifies that all DFSORT control statements are printed on the message data set.

NOLIST
specifies that DFSORT control statements are not printed.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

LISTX or NOLISTX

```
►►──────────┬─LISTX───┬─────────────────────────────────────────►◄
            └─NOLISTX─┘
```

Temporarily overrides the LISTX installation option, which specifies whether DFSORT writes to the message data set the program control statements returned by an EFS program. See *DFSORT Messages and Codes* for full details on use of the message data set.

LISTX

means that control statements returned by an EFS program are to be printed to the message data set.

NOLISTX

means that control statements returned by an EFS program are not to be printed to the message data set.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- If EFS=NONE is in effect after final override rules have been applied, NOLISTX will be set in effect.
- LISTX and NOLISTX can be used independently of LIST and NOLIST.
- For more information on printing EFS control statements, see *DFSORT Messages and Codes.*

MSGDDN

```
►►──────────MSGDDN=ddname────────────────────────────────────────────►◄
```

Temporarily overrides the MSGDDN installation option, which specifies an alternate DDNAME for the message data set. See the explanation of the MSGDDN parameter in "OPTION Control Statement" on page 97 for more information.

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, SYSOUT is used instead. For details on using the message data set, see *DFSORT Messages and Codes.*

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: MSGDD can be used instead of MSGDDN.

MSGPRT

```
►►──────────MSGPRT=┬ALL──────┬───────────────────────────────────────►◄
                   ├CRITICAL┤
                   └NONE─────┘
```

Temporarily overrides the MSGPRT installation option, which specifies the class of messages to be written to the message data set. See *DFSORT Messages and Codes* 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.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: The forms FLAG(I)|FLAG(U)|NOFLAG, and MSG={NO|AP|AC|CC|CP|PC} are also accepted. The following table lists the equivalent MSGPRT/MSGCON specifications for these alternate forms:

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

OUTREL or NOOUTREL

```
►►──────────┬─OUTREL───┬───────────────────────────────────────────►◄
            └─NOOUTREL─┘
```

Temporarily overrides the OUTREL installation option, which specifies whether unused temporary SORTOUT data set space is to be released.

OUTREL
means 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.

Note: RLSOUT and NORLSOUT can be used instead of OUTREL and NOOUTREL, respectively.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

RESALL

```
►►──────RESALL=─┬─n──┬──────────────────────────────►◄
                └─nK─┘
```

Temporarily overrides the RESALL installation option, which specifies the number of bytes to be reserved in a REGION for system use. For more information, see the explanation of the RESALL parameter ("OPTION Control Statement" on page 97).

n

is an integer specifying 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.

nK

nK specifies that n times 1024 bytes of storage are to be reserved, when SIZE/MAINSIZE=MAX is in effect. If you specify less than 4K, 4K is used.

Limit: 5 digits.

Note: RESERVE can be used instead of RESALL.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

SIZE

```
►►──────SIZE=─┬─n───────┬───────────────────────────►◄
              ├─nK──────┤
              ├─MAX─────┤
              ├─MAX-m───┤
              └─MAX-mK──┘
```

Temporarily overrides the SIZE installation option, which specifies the amount of main storage available to DFSORT. See the explanation of the MAINSIZE parameter in "OPTION Control Statement" on page 97.

n

n is an integer specifying the number of bytes of main storage to be allocated.

Limit: 8 digits.

nk

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

Limit: 5 digits.

MAX

instructs DFSORT 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 or MVS/ESA system) set when DFSORT was installed.

MAX-m

specifies the RESALL value (m), in bytes. 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 or MVS/ESA system) *minus* the amount of storage reserved for system and application use (RESALL).

Limit for m: 8 digits.

MAX-mK

specifies the RESALL value (m times 1024) in kilobytes. 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 or MVS/ESA system) *minus* the amount of storage reserved for system and application use (RESALL).

Limit for m: 5 digits.

Specifying a SIZE value equal to your installation's MAXLIM value is equivalent to specifying SIZE=MAX. If the SIZE or MAINSIZE passed to DFSORT is equal to the MAXLIM, the MAXLIM value will be used and DFSORT will set MAX in effect.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: The forms SIZE(option), CORE=option, and CORE(option) can be used instead of RESALL=option.

SKIPREC

```
►►──────────SKIPREC=z──────────────────────────────────────────►◄
```

Specifies the number of records (z) you want to skip before starting to sort or copy the input data set. SKIPREC is typically used to bypass records not processed from the previous DFSORT job. For more information, see the discussion of the SKIPREC option in "OPTION Control Statement" on page 97.

Default: None; optional. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

STIMER or NOSTIMER

```
►►──────┬─STIMER───┬──────────────────────────────────►◄
        └─NOSTIMER─┘
```

Temporarily overrides the STIMER option, which specifies whether DFSORT can use the STIMER macro.

STIMER
means that an STIMER will be issued and processor time data will appear in SMF records and ICETEXIT statistics.

NOSTIMER
means that no STIMER will be issued and processor time will not appear in SMF records or ICETEXIT statistics.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: If your exit or exits take checkpoints, then an STIMER must not be issued.

STOPAFT

```
►►──────STOPAFT=n─────────────────────────────────────►◄
```

Specifies the maximum number of records (n) 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). For more information, see the discussion of the STOPAFT option in "OPTION Control Statement" on page 97.

Default: None; optional. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: If you specify (1) FILSZ=x in the EXEC PARM, or (2) SIZE=x or FILSZ=x on the OPTION or SORT statement, and the number of records accepted for processing does not equal x, then DFSORT issues an error message and terminates.

WRKREL

```
►►──────┬─WRKREL───┬──────────────────────────────────►◄
        └─NOWRKREL─┘
```

Temporarily overrides the WRKREL installation option, which specifies whether unused temporary SORTWKnn data set space will be released.

WRKREL
specifies that unused space will be released.

NOWRKREL
specifies that unused space will not be released.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- If you have dedicated certain volumes for SORTWKnn data sets, and you do not want unused temporary space to be released, you should specify NOWRKREL.
- RELEASE=ON and RELEASE=OFF can be used instead of WRKREL and NOWRKREL, respectively.

WRKSEC

```
►►──────────┬─WRKSEC───┬──────────────────────────────────────────────────►◄
            └─NOWRKSEC─┘
```

Temporarily overrides the WRKSEC installation option, which specifies whether DFSORT uses automatic secondary allocation for temporary SORTWKnn data sets.

WRKSEC
specifies that automatic secondary allocation for temporary SORTWKnn data sets will be used, and that 25 percent of the primary allocation will be used as the secondary allocation.

NOWRKSEC
specifies that automatic secondary allocation for temporary SORTWKnn data sets will not be used.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: SECOND=ON and SECOND=OFF can be used instead of WRKSEC and NOWRKSEC, respectively.

Using DD Statements

A DFSORT job always requires DD statements after the EXEC statement. DD statements fall into two categories:

- System DD statements (discussed in detail in "System DD Statements" on page 39).
- Program DD statements (discussed in detail in "Program DD Statements" on page 40).

System DD statements (and some program DD statements) are usually supplied automatically when you use a cataloged procedure. Others you must always supply yourself.

The DD statement parameters, the conditions under which they are required, and the default values, are summarized in Table 1. The subparameters of the DCB parameter (a DD statement parameter) are described similarly in Figure 3 on page 38.

Note:

- 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 "Data Set Notes and Limitations" on page 9.
- When using DFSORT applications, FREE=CLOSE cannot be used on any DD statements.

Table 1 (Page 1 of 2). DD Statement Parameters Used by DFSORT

Parameter	When Required	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 3 on page 38.)
UNIT	When the input data set is neither cataloged nor passed or when the data set is being created.	Specifies (symbolically or actually) the type and quantity of I/O units required by the data set.	
SPACE	When the DD statement defines a new data set on direct access.	Specifies the amount of space needed to contain the data set.	
VOLUME or VOL	When the input data set is neither cataloged nor passed, for multireel input or when the output data set is on direct access and is to be kept or cataloged.	Specifies information used to identify the volume or volumes occupied by the data set.	
LABEL	When the default value is not applicable.	Specifies information about labeling and retention for the data set.	The system assumes standard labeling.

Table 1 (Page 2 of 2). DD Statement Parameters Used by DFSORT

Parameter	When Required	Parameter Values	Default Value
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.

Duplicate DDNAMEs

If you specify a particular DDNAME (such as SORTIN) more than once within the same step, DFSORT uses the first DDNAME and ignores subsequent duplicates. Processing continues normally.

In addition:

- SORTWK0, SORTWK1...SORTWK9 can be specified *instead of* SORTWK00, SORTWK01...SORTWK09, respectively. If you specify both SORTWKx and SORTWK0x in the same job step, DFSORT treats them as duplicates, and ignores each usage after the first. For example, SORTWK2 and SORTWK02 are treated as duplicates. (Only SORTWK2 is used.)

  Note:

  For a tape work data set sort, SORTWKx will not be recognized because of the existing restriction which allows only SORTWK01, SORTWK02...SORTWK32. However, duplicates of these accepted DDNAMEs will be ignored.

- SORTIN0, SORTIN1...SORTIN9 can be specified *instead of* SORTIN00, SORTIN01...SORTIN09, respectively. If you specify both SORTINx and SORTIN0x in the same job step, DFSORT treats them as duplicates, and ignores each usage after the first. For example, SORTIN2 and SORTIN02 are treated as duplicates. (Only SORTIN2 is used.)

  Note:

  For a conventional merge, SORTINx will not be recognized because of the existing restriction which allows only SORTIN01, SORTIN02...SORTIN16. However, duplicates of these accepted DDNAMEs will be ignored.

Shared Tape Units

The following pairs of DFSORT data sets can be assigned to a single tape unit:

- 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 can include the parameter: UNIT=AFF=SORTIN in the DD statement for SORTWK01. The AFF subparameter causes the system to place the data set on the same unit as the dataset with the DDNAME following the subparameter (SORTIN, in this case).

In the same way, you can 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 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 SORTOUT data set, the same as the first SORTIN, or appropriate SORTINnn data set.[3] • No default if input on unlabeled tape, or BLP or NSL specified.
LRECL[1]		Specifies the maximum length (in bytes) of the logical records in the data set.	
BLKSIZE[2]		Specifies the maximum length (in bytes) of the physical records in the data set.	
OPTCD	When processing data in 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] For padding and truncating fixed-length records, see "Data Set Notes and Limitations" in the Table of Contents.

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

[3] Blockset merge uses the largest LRECL and BLKSIZE found among the input data sets. Conventional merge uses the LRECL and BLKSIZE from SORTIN01.

Figure 3. DCB Subparameters Used by DFSORT

System DD Statements

If you choose not to use the SORT or SORTD cataloged procedures to invoke DFSORT, you might need to supply system DD statements in your input job stream. (See also the following section for DD statements dedicated to DFSORT, such as SORTIN). The DD statements contained in the cataloged procedure (or provided by you) are:

//JOBLIB DD or

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

//SYSIN DD contains DFSORT control statements, comment statements, blank statements and remarks when DFSORT is invoked with JCL rather than by another program. It can also contain user exit routines, in object deck format, to be link-edited by DFSORT.

- If you use DFSPARM, then SYSIN is not necessary unless your job requires link-editing.
- The SYSIN 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.

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

If you use the DFSPARM DD statement instead, you can specify both EXEC PARM options and DFSORT control statements in a single source data set that overrides all other sources. See "DFSPARM DD Statement" on page 49.

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, SORTCNTL, or DFSPARM, SYSIN 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 are using no DDNAME other than SYSOUT, the use of EXHIBIT or DISPLAY in your COBOL program can produce 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 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 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

Even if you use the SORT or SORTD cataloged procedure to invoke DFSORT, you might need to supply additional dedicated DD statements: The following list summarizes each of these statements, and a more detailed explanation of each one follows.

//SORTLIB DD defines the data set that contains special load modules for DFSORT, and can usually be omitted.

//SORTIN DD defines the input data set for a sorting or copying application. Cannot be used for a merging application or if SORTINnn is used.

//SORTINnn DD defines the input data sets for a merging application. Cannot be used for a sorting application or if SORTIN is used.

//SORTWKnn DD defines intermediate storage data sets. Usually needed for a sorting application unless dynamic allocation is requested. Will not be used for a copying or merging application (but may be opened, if present).

//SORTOUT DD defines the output data set for a sorting, merging, or copying application.

//SORTCKPT DD defines a data set for checkpoint records. Required only if you are using the checkpoint facility.

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

//DFSPARM DD defines the data set from which both additional or changed DFSORT program control statements and EXEC statement parameters can be read, when DFSORT is JCL-invoked or dynamically invoked.

//SORTDKnn DD defines the data set used for a VIO SORTWKnn allocation by DFSORT if it is dynamically reallocated; SORTDKnn must 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.

//SORTSNAP DD defines the snap dump data set dynamically allocated by DFSORT. SORTSNAP must never be specified in the job stream.

SORTLIB DD Statement

The SORTLIB DD statement can usually be omitted. This statement describes the data set that contains special DFSORT load modules.

When Required: A SORTLIB DD statement is required only for:

- Sort applications using tape work data sets
- Merge applications for which Blockset cannot be used (see message ICE800I).

The ICEMAC SORTLIB option determines whether DFSORT searches a system library or private library for the load modules required by tape work data set sorts and Conventional merges. If the load modules were installed in a system library, the SORTLIB DD statement is unnecessary, and is ignored.

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, if the volume is not already mounted, writes a mounting message to the operator.

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 Table 1 on page 36. The subparameters of the DCB parameter are described similarly in Figure 3 on page 38. 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 both provide an E15 exit that supplies all input to DFSORT, and 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 empty and null non-VSAM data sets for sorting and copying (be sure to apply DCB parameters). DFSORT also accepts empty VSAM data sets for sorting or copying. For non-VSAM data sets, DFSORT examines the DS1LSTAR field in the format-1 DSCB to determine whether the data set is empty or null. If DS1LSTAR is zero, DFSORT treats the data set as empty or null.

Note that a null data set is one that has been newly created, but never successfully closed. Null data sets cannot be processed successfully for a tape work data set sort.

DFSORT accepts an empty (not a null) QSAM data set for sorting or copying (be sure to supply DCB parameters). DFSORT accepts an empty VSAM data set for sorting or copying. 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 "Data Set Notes and Limitations" on page 9 for additional considerations.

The following rules apply to concatenated data sets:

- RECFM must be the same for all data sets in the concatenation, except that FB and FBS can be mixed.
- BLKSIZE can vary, but the data set with the largest block size must be specified on the first DD statement of the concatenation.

- With fixed-length records, LRECL must be the same for all data sets. With variable-length records, LRECL can vary, but the largest size must be specified for the data set described on the first DD statement.
- If the data sets are on unlike devices you cannot use the EXLST parameter at exit E18.
- If BSAM is used, all null data sets must precede all non-null data sets.

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, if the volume is not already mounted, writes a mounting message to the operator.

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 a disk or on a standard-labeled tape, 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.

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 non-VSAM or VSAM, but not both. Empty and null non-VSAM data sets are accepted. An empty VSAM data set causes a VSAM open error (code 160), and DFSORT terminates. For non-VSAM data sets, DFSORT examines the DS1LSTAR field in the format-1

DSCB to determine whether the data set is null or empty. If DS1LSTAR is zero, DFSORT treats the data set as null or empty. Note that a null data set is one that has been newly created, but never successfully closed. Null data sets cannot be processed successfully by the Conventional merge technique.

Data sets can be multivolume, but not concatenated.

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

BLKSIZE can vary, but for a Conventional merge, SORTIN01 must specify the largest blocksize.

With fixed-length records, LRECL must be the same for all data sets. With variable-length records, LRECL can vary, but for a Conventional merge,the LRECL from SORTIN01 is used.

See "Data Set Notes and Limitations" on page 9 for additional considerations.

Coding Notes

- You can merge up to 16 data sets. (Blockset might allow more, depending on available storage.)
- If Blockset merge is used, nn can be any integer from 00 (the initial zero is optional) to 99, in any order. Remember that Blockset merge treats DDNAMEs of the form SORTINx and SORTIN0x as duplicates, and ignores any occurrences after the first. For example, if DFSORT reads a DD statement as SORTIN4 DD... and subsequently reads another as SORTIN04 DD..., the latter DD statement is ignored.
- If Conventional merge is used, nn can range from 01 to 16. The first number you use must be 01, and the remainder must follow in numeric order. No numbers can be skipped. Remember that Conventional merge ignores DDNAMEs in the form "SORTINx".
- 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
- Dynamic allocation has been requested.

For information on how to calculate the amount of storage needed, see "Intermediate Storage" on page 311.

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 311. Disk types can be mixed.

Tape must be nine-track unless input is on seven-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.
  - With disk work areas, nn can be any decimal number from 00 (the initial zero is optional) through 99 and numbers can be in any order; however, if more than 32 are specified, only the first 32 are used. Remember that DFSORT treats DDNAMEs of the form SORTWKx and SORTWK0x as duplicates, and ignores any occurrences after the first. For example, if DFSORT reads a DD statement as SORTWK4 DD... and subsequently reads another as SORTWK04 DD..., the latter DD statement is ignored.

- 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.
- Do not include parameters relating to ISCII/ASCII data, 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 for temporary SORTWKnn data sets, automatic secondary allocation will be used unless NOWRKSEC is in effect. (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.)
- DFSORT uses *only* the space on the first volume specified for a multivolume data set. Space on the second and subsequent volumes is not used. Multivolume SORTWKnn data sets are therefore effectively treated as single-volume SORTWKnn data sets.
- If primary space is fragmented, then all but the first fragment are handled as secondary space.

Virtual I/O: If a SORTWKnn data set is specified on a virtual device:

- With VIO=NO, DFSORT carries out dynamic reallocation using the DDNAME SORTDKnn on a real device with the same device type as the virtual device. If a real device corresponding to the virtual device is not available in the system, DFSORT terminates.
- With VIO=YES, the virtual device is used; performance might be 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 "Data Set Notes and Limitations" on page 9 and "VSAM Considerations" on page 10.

Note:

- If LABEL=RETPD is specified in the SORTOUT DD statement for a standard labeled tape, the DCB parameters must also be specified. If the DCB parameters are not specified, the tape might be opened twice.
- 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 might be degraded.
- If no secondary allocation is requested for a temporary or new SORTOUT data set, automatic secondary allocation will be used unless NOOUTSEC is in effect.

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 can be allocated on any device that operates with the Basic Sequential Access Method (BSAM). Processing must be restarted only 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 311.

If you want to use the checkpoint/restart facility, refer to "Checkpoint/Restart" on page 383.

SORTCNTL DD Statement

You can use the SORTCNTL data set to read changed and additional DFSORT control statements, comment statements, blank statements, and remarks, when DFSORT is invoked from another program (written, for example, in COBOL or PL/I).

- The SORTCNTL data set usually resides in the input stream, but 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.
- SORTCNTL cannot be concatenated data sets.
- When DFSORT is invoked from a PL/I program, the SORTCNTL or DFSPARM data set must not be used to supply a new RECORD control statement.

Example 10. SORTCNTL DD Statement

```
//SORTCNTL DD  *
```

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

If your program invokes DFSORT more than once, you can direct DFSORT to read different versions of the SORTCNTL data set at each call. See the explanation of the SORTDD parameter of the OPTION program control statement on page 114.

Note: If you use the DFSPARM DD statement instead, you can specify both EXEC PARM options and DFSORT control statements in a single source data set that overrides all other sources. See "DFSPARM DD Statement." For override rules, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

DFSPARM DD Statement

The DFSPARM DD statement can be used to read changed or additional DFSORT program control statements and EXEC statement PARM options from a single DD source. Because statements in the DFSPARM data set are read whether DFSORT is dynamically invoked or invoked with JCL, you can specify EXEC PARM options when invoking DFSORT from another program (unlike SORTCNTL). DFSPARM accepts all DFSORT program control statements and all EXEC statement PARM options (including those ignored by SYSIN and SORTCNTL) and any equivalent options specified on a DFSORT OPTION statement.

DFSPARM also accepts comment statements, blank statements, and remarks.

Full override and applicability details are listed in Appendix C, "Specification/Override of DFSORT Options" on page 353.

- If you use DFSPARM, SYSIN is not necessary unless your job requires link-editing.
- The DFSPARM data set usually resides in the input stream, but 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.
- DFSPARM cannot be concatenated data sets.
- When DFSORT is invoked from a PL/I program, the SORTCNTL or DFSPARM data set must not be used to supply a new RECORD control statement.

Note: The DDNAME "DFSPARM" is used throughout this book to refer to this data set source for EXEC PARM options and DFSORT program control statements. When your system programmers installed DFSORT, they might have changed this name to one more appropriate for your site with the PARMDDN option of the ICEMAC installation macro. Verify the correct DDNAME before attempting to use the features available with DFSPARM.

General Coding Notes: Coding of parameters in the DFSPARM DD statement follows the same rules used for the JCL EXEC statement PARM options, and the program control statements specified in SYSIN or SORTCNTL, with these exceptions:

- Labels are not allowed.
- PARM options and program control statements cannot be mixed on the same line, but can be specified in any order on different lines.
- PARM options must be specified without the PARM= keyword and without quote marks.
- Commas (or semicolons) are accepted but not required to continue PARM options to another line.
- Leading blanks are not required for PARM options, but at least one leading blank is required for program control statements.

Example 11. DFSPARM DD Statement

```
//DFSPARM  DD  *
  SORT FIELDS=(1,2,CH,A),STOPAFT=300
ABEND
  OPTION SORTIN=DATAIN
  STOPAFT=500
```

In this example the DFSPARM DD data set passes a DFSORT SORT statement, the ABEND and STOPAFT parameters equivalent to specifying PARM='ABEND,STOPAFT=500' in a JCL EXEC statement, and a DFSORT OPTION statement.

Notes:

1. Neither of the DFSORT program control statements appear on the same line as the ABEND or STOPAFT PARM options.
2. The PARM option STOPAFT=500 overrides the SORT control statement option STOPAFT=300.

Example 12. DFSPARM DD Statement

```
//DFSPARM DD *
  SORT FIELDS=(5,2,CH,D),SKIPREC=10
  STOPAFT=100,BSAM,SKIPREC=5
  OPTION SORTIN=DATAIN,SKIPREC=20
```

In this example, the DFSPARM DD data set contains a SORT program control statement, three PARM options on one line, and an OPTION program control statement.

Note: Because PARM options override program control statements, DFSORT uses SKIPREC=5, and ignores the other SKIPREC specifications.

For information on the parameters used in the DFSPARM DD statement, the conditions under which they are required, and any default values assumed if a parameter is omitted, see "Specifying EXEC Statement PARM Options" on page 21 and Chapter 3, "Using DFSORT Program Control Statements" on page 53.

SORTDKnn DD Statement

SORTWKnn data sets can be assigned to VIO. If the ICEMAC parameter VIO is specified or defaults to NO, VIO SORTWKnn data sets are deallocated and reallocated by DFSORT with the ddname SORTDKnn. The DDNAME 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 job, a SYSOUT DD statement or a DDNAME DD statement (where DDNAME is the alternate message data set DDNAME specified during installation or execution) 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.

Example 13. SORTDIAG DD Statement

```
//SORTDIAG DD  DUMMY
```

SORTSNAP DD Statement

The SORTSNAP DD statement defines the data set where ESTAE recovery routine snap dumps, and snap dumps requested before or after a call to an EFS program, are printed. SORTSNAP is dynamically allocated by DFSORT whenever it is required. The DDNAME, SORTSNAP, is reserved for DFSORT.

Chapter 3. Using DFSORT Program Control Statements

Program control statements direct DFSORT processing of your records. Some program control statements are required, while others are optional. You use the control statements to:

- Indicate whether you want to sort, merge, or copy records
- Describe the control fields you want to use
- Indicate program exits for transferring control to your own routines
- Describe DFSORT functions you want to invoke
- Describe your input and output files
- Indicate various options you want to use while processing.

You can supply program control statements to DFSORT from:

- A SYSIN data set
- A SORTCNTL data set
- A DFSPARM data set
- A 24-Bit parameter list
- An extended parameter list.

See Appendix C, "Specification/Override of DFSORT Options" on page 353 for an explanation of when you might want to use each source.

DFSORT Panels offers you an alternative to coding program control statements directly. When you use panels to prepare a job to be run or saved in a data set, you can create the necessary statements in correct syntax by entering information and commands on-line. See Chapter 4, "Using Panels to Create and Submit DFSORT Jobs" on page 145 for details on using DFSORT Panels.

This chapter begins with a summary of DFSORT program control statements and coding rules, and then provides, in alphabetical order, a detailed description of each statement.

Control Statement Summary

Control Statements

Describing your primary task (sorting, merging, or copying):

The only required program control statement in a DFSORT job is a SORT, MERGE, or OPTION statement that specifies whether you want to sort, merge, or copy records. (Copying can be specified on any of the three statements).

SORT Describes your control fields if you are coding a sort job, or can be used to specify a copy job. Indicates whether you want ascending or descending order.

MERGE Describes your control fields if you are coding a merge job, or can be used to specify a copy job. Indicates whether you want ascending or descending order.

OPTION Overrides installation defaults (such as EQUALS, CHALT, and CHECK) and supplies optional information (such as DYNALLOC and SKIPREC). Can be used to specify a copy job.

Including or omitting records from the output data set:

INCLUDE Specifies that only records whose fields meet certain criteria are included.

OMIT Specifies that any records whose fields meet certain criteria are deleted.

Reformatting and editing records: You can modify individual records by deleting and reordering fields and inserting blanks, zeros, or constants.

INREC Specifies how records are reformatted before they are sorted, copied, or merged.

OUTREC Specifies how records are reformatted after they are sorted, copied, or merged.

Invoking additional functions and options: You can use the remaining control statements to perform a variety of tasks.

ALTSEQ Modifies the standard IBM EBCDIC collating sequence. The modified sequence is used for any control field whose format is specified as AQ.

DEBUG Used for diagnostic purposes.

END Causes DFSORT to discontinue reading SYSIN, SORTCNTL, or DFSPARM.

MODS Required only if you include user exit routines in a DFSORT job. See Chapter 5, "Using Your Own Exit Routines" on page 181 for information about user exit routines.

RECORD Supplies 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 and output are VSAM data sets.

SUM Specifies that numeric summary fields in records with equal control fields are summed in one record, and that the other records are deleted.

General Coding Rules

See "Inserting Comment Statements" on page 58 for an explanation of how to use comment statements, blank statements, and remarks. DFSORT program control statements and EXEC PARM options can also be specified together in a user-defined DD data set. See "DFSPARM DD Statement" on page 49 for special coding conventions that apply to this DD source.

All other DFSORT control statements have the same general format, shown in Figure 4 on page 56. The illustrated format does *not* apply to control statements you supply in a parameter list. See Chapter 6, "Invoking DFSORT from a Program" on page 231 for information on the special rules that apply.

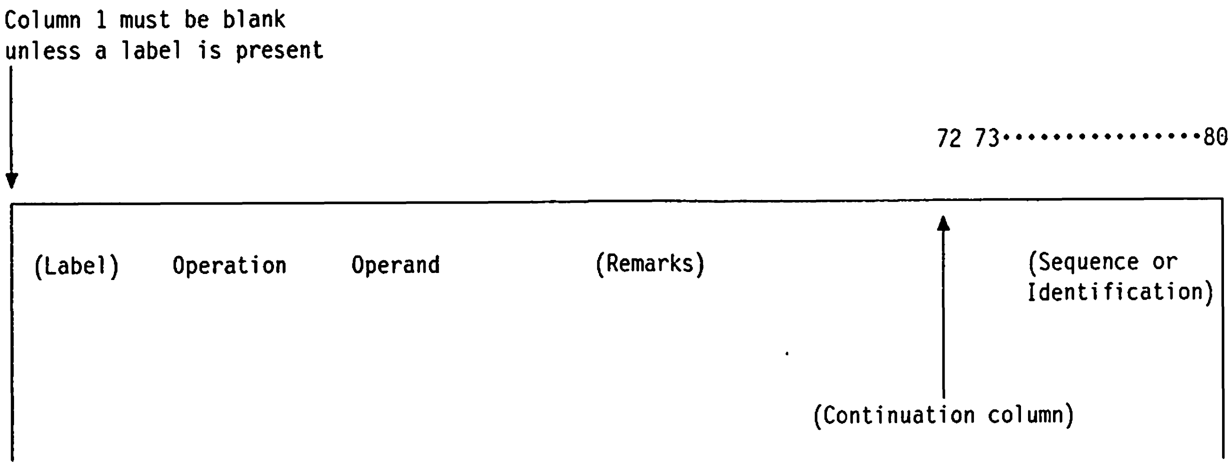


Figure 4. Control Statement Format

The control statements are free-form; that is, the operation definer, operand(s), and comment field can 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.

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

Operation Field: This field can appear anywhere between column 2 and column 71 of the first line. It contains a word (for example, SORT or MERGE) that identifies the statement type to the program. 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 or semicolons. This field must follow the operation field, and be separated from it by at least one blank. No blanks are allowed within the parameters, but a blank is required at the end of all parameters. If the statement occupies more than one line, the operand 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 can 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
```

Remark Field: This field can contain any information. 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 or semicolon 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 can be used for any purpose.

Continuation Lines

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

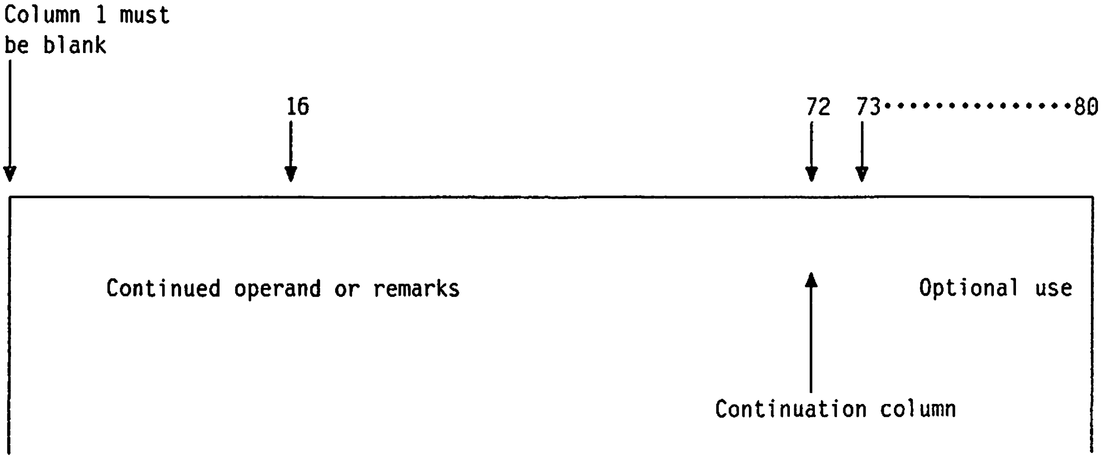


Figure 5. Continuation Line Format

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 can 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 or a semicolon, the continuation column (72) can be left blank, and the continuation can begin in any column from 2 through 71. If the comma or semicolon 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 or semicolon, 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:

```
 1             16                                                      72
 ↓              ↓                                                       ↓
 SORT FIELDS=(5,8,A,20,2,D),
  FORMAT=CH
 OPTION SKIPREC=2,LIST,  SKIP 2 RECORDS——LIST CONTROL STATEMENTS——
        DYNALLOC         USE DYNAMIC ALLOCATION
 INCLUDE COND=(1,10,CH,EQ,C'STOCKHOLM',AND,21,8,ZD,GT,+500,OR,31,4,CH,N*
               E,C'HERR')
```

Inserting Comment Statements

- Specify comment statements by coding an asterisk (*) in column 1. A comment statement is printed along with other DFSORT program control statements, but is not otherwise processed.
- A statement with blanks in columns 1 through 71 is treated as a comment statement.
- Comment statements are allowed only in the DFSPARM, SYSIN, and SORTCNTL data sets.

Coding Restrictions

The following rules apply to control statement preparation:

- Labels, operation definers, and operands must be in uppercase EBCDIC.
- Column 1 of each control statement can be used only for a label or for a comment statement that begins with an asterisk in column 1.
- Labels must begin in column 1, and conform to operating system requirements for statement labels.
- The entire 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.
- Blanks are not allowed in operands. Anything following a blank is considered part of the remark field.
- Values can contain no more than eight alphameric characters (except for estimated data set size, which can contain nine characters).
- Commas, semicolons, and blanks can be used only as delimiters. They can be used in values only if the values are constants.
- Each type of program control statement can appear only once within a single source (for example, the SYSIN data set).

EFS Restrictions When an EFS Program Is in Effect

In addition to the items above, the following restrictions apply to control statement preparation for an EFS program.

- Non-DFSORT operation definers can be up to 8 bytes long
- An operation definer with no operands is allowed only if it:
  - Is supplied through SYSIN, SORTCNTL, or DFSPARM
  - Is the only operation definer on a line; column 72 must contain a blank.

Using Control Statements from Other IBM Programs

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 DFSORT now uses the OPTION control statement, OPTION control statements in any job streams from other IBM sort programs cause DFSORT to terminate, unless the parameters from the other program conform to the DFSORT OPTION control statement parameters.

DFSORT 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, DFSORT does not accept other programs' MODS control statements, with the exception of those used by the IBM Sort/Merge Program 360S-SM-023, and Licensed Program Sort/Merge 5734-SM1.

Note that applications using the 360S-SM-023 and 5734-SM1 programs can be successfully run using DFSORT.

ALTSEQ Control Statement

```
                    ┌─,──┐
                    ▼    │
►►──ALTSEQ CODE=──(──fftt─┴─)──────────────────────────►◄
```

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

When you supply an ALTSEQ statement, DFSORT applies the modified collating sequence to 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 site, if there is one. Otherwise, DFSORT uses the standard EBCDIC collating sequence.

ff

represents, in hexadecimal, the EBCDIC collating position of the character whose position is to be changed.

tt

represents, in hexadecimal, the EBCDIC collating position to which the character is to be moved.

The order in which the parameters are specified is not important.

Note:

- If CHALT is in effect, control fields with format CH are translated by the ALTSEQ table in addition to those with format AQ.
- Use of ALTSEQ can degrade performance.

Default: Usually the installation option, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

ALTSEQ Statement Examples

ALTSEQ Example 1

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

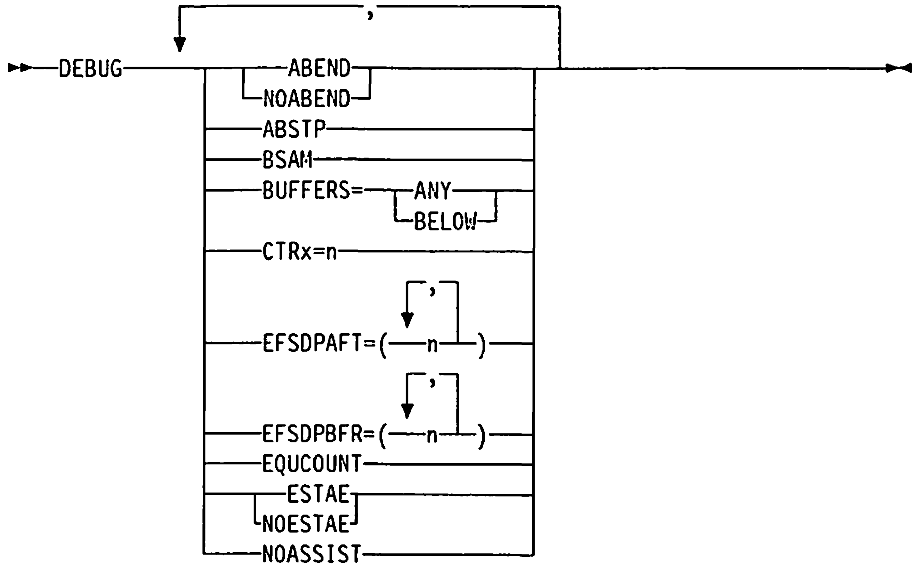
The character represented by X'5B'($ or national character) is to collate after 'Z' (at position X'EA').

ALTSEQ Example 2

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

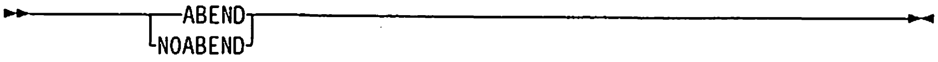
The numerals 0 through 9 are to collate before uppercase letters (but after lowercase letters).

DEBUG Control Statement



The statement is not intended for regular use; only ABEND, NOABEND, BSAM, and EQUCOUNT are of general interest. For a tape work sort or a Conventional merge, only the ABEND or NOABEND parameters of the DEBUG statement are used. For more information about problem diagnosis, see *DFSORT Diagnosis Guide*.

ABEND or NOABEND



Temporarily overrides the ERET installation option, which specifies whether DFSORT abends or terminates with a return code of 16 if your sort, copy, or merge is unsuccessful.

ABEND
specifies that if your sort, copy, or merge is unsuccessful, it abends with a user completion code equal to the appropriate message number, or with a user-defined number between 1 and 99, as set during installation with the ICEMAC option ABCODE=n.

When DEBUG ABEND is in effect, a user abend code of zero might be issued when a tape work data set sort or Conventional merge is unsuccessful.

NOABEND
specifies that an unsuccessful sort, copy, or merge terminates with a return code of 16.

Note: RC16=ABE and NORC16 can be used instead of ABEND and NOABEND, respectively.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

ABSTP

```
►►──────────ABSTP──────────────────────────────────────────►◄
```

ABSTP prevents loss of needed information in a dump when Blockset terminates. This option overrides ERET, ABEND, and NOABEND. If the DFSORT application is unsuccessful, an abend is forced with a completion code equal to the appropriate message number, or with the user ABEND code set during installation with the ICEMAC option ABCODE=MSG or ABCODE=n. The message is *not* written if NOESTAE is in effect.

Default: None; optional

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

BSAM

```
►►──────────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. BSAM is ignored for VSAM SORTIN and/or SORTOUT data sets.

Note: Use of this option might degrade performance.

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

BUFFERS

```
►►──────────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 C, "Specification/Override of DFSORT Options" on page 353.

CTRx

```
►►──────CTRx=n──────────────────────────────────────►◄
```

DFSORT keeps a count of the input and output records, and abends with code 0C1 when the count reaches n.

The numbers that can be assigned to x are:

2 Count of input records being moved from the input buffer (not used for a copy)

3 Count of output records being moved to the output buffer (not used for a copy or merge)

4 Count of input records inserted by E15 (not used for Blockset)

5 Count of output records deleted by E35 (not used for Blockset)

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

EFSDPAFT

```
                       ┌,┐
                       ▼ │
►►──────EFSDPAFT=──n─┴──────────────────────────────►◄
```

EFSDPAFT causes a SNAP dump after a Major Call to an EFS program. Any combination of the numbers can be specified.

The numbers have the following meanings:

2 Take the SNAP dump after Major Call 2 to the EFS program

3 Take the SNAP dump after Major Call 3 to the EFS program

4 Take the SNAP dump after Major Call 4 to the EFS program

5 Take the SNAP dump after Major Call 5 to the EFS program

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

EFSDPBFR

```
                       ┌,┐
                       ▼ │
►►──────EFSDPBFR=──n─┴──────────────────────────────►◄
```

This parameter causes a SNAP dump before a Major Call to an EFS program. Any combination of the numbers can be specified.

The numbers have the following meanings:

2 Take the SNAP dump before Major Call 2 to the EFS program

3 Take the SNAP dump before Major Call 3 to the EFS program

4 Take the SNAP dump before Major Call 4 to the EFS program

5 Take the SNAP dump before Major Call 5 to the EFS program

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

ESTAE or NOESTAE

```
►►──────────┬─ESTAE───┬─────────────────────────────────────────►◄
            └─NOESTAE─┘
```

Temporarily overrides the ESTAE installation option, which determines whether DFSORT should delete its ESTAE recovery routine early, or use it for the entire run.

DFSORT normally establishes an ESTAE recovery routine at the beginning of a run. If an abend occurs and the ESTAE option is in effect, the system passes control to the recovery routine. The routine terminates the run after attempting to:

- Print additional abend information
- Continue a sort, merge, or copy job after successful output
- Call the EFS program at Major Calls 4 and 5 for cleanup and housekeeping
- Write an SMF record
- Call the ICETEXIT termination exit.

If an abend occurs and the ESTAE option is not in effect, these functions might not be performed.

ESTAE
specifies that DFSORT can use its ESTAE recovery routine for the entire run.

NOESTAE
specifies that DFSORT is to delete its ESTAE recovery routine at a point early in its processing. If DFSORT terminates or abends before this point is reached, it will not delete its ESTAE recovery routine; that is, NOESTAE will not be in effect.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options."

Note: See Appendix F, "DFSORT Abend Processing" on page 383 for more information on the DFSORT ESTAE recovery routine.

EQUCOUNT

```
►►——————EQUCOUNT——————————————————————————————————————————►◄
```

Use this parameter to determine the number of records having equal keys which have been sorted by the Blockset technique (printed in message ICE184I). For variable-length records, EQUCOUNT cannot be used without at least one intermediate data set.

Use of EQUCOUNT will degrade performance.

Default: None; optional

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

NOASSIST

```
►►——————NOASSIST——————————————————————————————————————————►◄
```

DFSORT uses System/370-XA Sorting Instructions on MVS/XA and MVS/ESA, 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 C, "Specification/Override of DFSORT Options" on page 353.

DEBUG Statement Examples

DEBUG Example 1

```
//SYSIN DD *
  SORT FIELDS=(1,6,A,28,5,D),FORMAT=CH
  DEBUG EQUCOUNT
```

Issue message ICE184I to indicate how many records with equal keys were sorted by the Blockset technique.

END Control Statement

```
►►—END————————————————————————————————————————————————►◄
```

The END statement is required if you want DFSORT to discontinue reading SYSIN, DFSPARM, 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 1

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

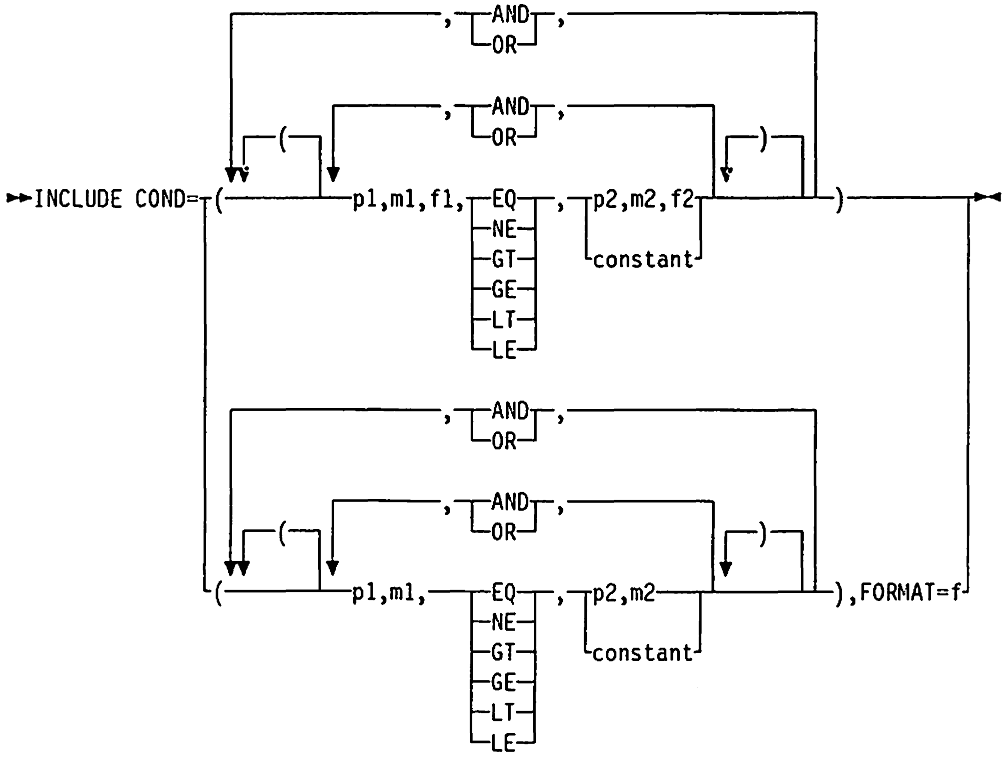
Because the OPTION statement appears after the END statement, it cannot be read.

END Example 2: 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



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 can 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.

By nesting relational conditions within parentheses, you can create logical expressions of higher complexity. Nesting of parentheses is limited only by the amount of available storage.

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┬,relational condition2──┴─┘
                                    └OR─┘
```

Default: None; must be specified.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

FORMAT

```
►►──FORMAT=f──────────────────────────────────────────────────►◄
```

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

Default: None. Must be specified if not included in the COND parameter.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Relational Condition

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

- 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.)
- The symbols & (AND) and | (OR) can be used instead of the words.

Relational Condition Format

Two formats for the relational condition can be used:

```
►►──(p1,m1,f1,┬─EQ─┬,┬p2,m2,f2┬)─────────────────────────────►◄
              ├─NE─┤ └constant┘
              ├─GT─┤
              ├─GE─┤
              ├─LT─┤
              └─LE─┘
```

Or,

```
►►──(p1,m1,─┬─EQ─┬─,─┬─p2,m2────┬─),FORMAT=f──────────────────►◄
            ├─NE─┤   └─constant─┘
            ├─GT─┤
            ├─GE─┤
            ├─LT─┤
            └─LE─┘
```

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: These variables specify a field in the input record to be compared either to 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 can extend beyond byte 4092.
- m1 specifies the length of the field. Acceptable lengths for different formats are in Table 2.
- f1 specifies the format of the data in the field. Permissible formats are given in Table 2.

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

Table 2 (Page 1 of 2). Permissible Format Codes

Format	Length	Description
CH	1-256	Character EBCDIC, unsigned.[2]
AQ	1-256	Character EBCDIC, alternate collating sequence.
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.

Table 2 (Page 2 of 2). Permissible Format Codes

Format	Length	Description
BI	1-256	Binary, unsigned.
AC	1-256	ISCII/ASCII character, unsigned.
CSL or LS	2-256	EBCDIC numeric, leading separate sign.
CST or TS	2-256	EBCDIC numeric, trailing separate sign.
CLO or OL	1-256	EBCDIC numeric, leading overpunch sign.
CTO or OT	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.
D2	1-256	User-defined data type. (Requires an EFS program.)

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 6 on page 72.

AC, ASL, and AST formats sequence EBCDIC data using the ISCII/ ASCII collating sequence.

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	D2
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	
D2														X

Figure 6. Permissible Field-to-Field Comparisons for INCLUDE/OMIT

Note to Figure 6: D2 field formats are user-defined.

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 7 on page 73.

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
D2	x	x	x

Figure 7. Permissible Field-to-Constant Comparisons for INCLUDE/OMIT

Note to Figure 7: D2 field formats are user-defined.

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

[±]*n*

When the decimal constant, n, is compared with a field of FI format, it cannot 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). You can specify up to 256 characters.

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. You can specify up to 256 pairs 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 identifier
	'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 a character string
X'00'	For a hexadecimal string

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

INCLUDE/OMIT Statement Notes

- The size of the routine generated by DFSORT to handle the INCLUDE/OMIT function depends 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.
- Floating point fields cannot be referenced in INCLUDE or OMIT statements.
- Any selection can be performed with either an INCLUDE or an OMIT statement. INCLUDE and OMIT are mutually exclusive.
- 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 can be changed by using parentheses inside the COND expression.
- If any changes are made to record formats by exits E15 or E32, the INCLUDE or OMIT statement must apply to the newest formats.
- 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 8 on page 76 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 8 will 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 8. Logic Table for INCLUDE/OMIT

INCLUDE Statement Examples

INCLUDE Example 1

```
INCLUDE   COND=(5,8,GT,13,8,|,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 nine-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. ("OMIT Statement Example" on page 95 illustrates how parentheses can be used to change the order of evaluation.) Also note that ending a line with a comma or semicolon followed by a blank indicates that the parameters continue on the next line, starting in any position from columns 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 includes only 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─────┬┴──)──────────────────►◄
                      └─c:─┘  └─p,m─┬────┬┘
                                    └─,a─┘
```

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 only those fields, in the order in which you have specified them, and aligned on the boundaries or in the columns you have indicated.

You can also insert blanks, binary zeros, character strings, and hexadecimal strings as separators before, between, and after the input fields in the reformatted input records.

For information concerning the interaction of INREC and OUTREC, see also "Use Options that Enhance Performance" on page 251.

FIELDS

```
                     ┌──────────,──────────┐
                     ↓                     │
►►────────FIELDS=──(──┬────┬──┬─────s─────┬┴──)──────────────────►◄
                      └─c:─┘  └─p,m─┬────┬┘
                                    └─,a─┘
```

specifies the order and alignment of the input and separation fields in the reformatted input record.

c:

Indicates the column in which the first position of the associated input or separation field is to be aligned, relative to the start of the reformatted input record. Unused space preceding the specified column is padded with EBCDIC blanks. The following rules apply:

- c must be a number between 1 and 32000.
- c: must be followed by an input field or a separation field.
- c must not overlap the previous input field or separation field in the reformatted input record.
- For variable-length records, c: must not be specified before the first input field (the record descriptor word), nor after the variable part of the input record.
- The colon (:) is treated like the comma (,) or semicolon (;) for continuation to another line.

For example:

Table 3. Examples of Valid and Invalid Column Alignment

Validity	Specified	Result
Valid	33:C'State '	Columns 1-32 — blank Columns 33-40 — 'State '
Valid	20:5,4,30:10,8	Columns 1-19 — blank Columns 20-23 — input field (5,4) Columns 24-29 — blank Columns 30-37 — input field (10,8)
Invalid	0:5,4	Column value cannot be zero.
Invalid	:25Z	Column value must be specified
Invalid	32001:21,8	Invalid — column value must be less than 32001
Invalid	5:10:2,5	Column values cannot be adjacent
Invalid	20,10,6:C'AB'	Column value overlaps previous field

s

Indicates a separation field to be inserted into the reformatted input record in the position you code it relative to the other fields. It can be specified before or after any input field. Consecutive separation fields can be specified. For variable- length records, s must not be specified before the first input field (the record descriptor word), or after the variable part of the input record. Permissible values are:

nX Blank separation. n bytes of EBCDIC blanks (X'40') are inserted in the reformatted input records. n can be from 1 to 4095. If n is omitted, 1 is used.

For example:

Table 4. Examples of Valid and Invalid Blank Separation

Validity	Specified	Result
Valid	X or 1X	1 blank
Valid	4095X	4095 blanks
Invalid	5000X	Too many repetitions. Use two adjacent separation fields instead (2500X,2500X, for example)
Invalid	0X	0 is not allowed.

nZ Binary zero separation. n bytes of binary zeros (X'00') are inserted in the reformatted input records. n can be from 1 to 4095. If n is omitted, 1 is used.

For example:

Table 5. Examples of Valid and Invalid Binary Zero Separation

Validity	Specified	Result
Valid	Z or 1Z	1 binary zero
Valid	4095Z	4095 binary zeros
Invalid	4450Z	Too many repetitions. Use two adjacent separation fields instead (4400Z,50Z for example).
Invalid	0Z	0 is not allowed.

nC′xx...x′ Character string separation. n repetitions of the character string constant (C′xx...x′) are inserted into the reformatted input records. n can be from 1 to 4095. If n is omitted, 1 is used. x can be any EBCDIC character. You can specify from 1 to 256 characters.

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

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

For example:

Table 6. Examples of Valid and Invalid Character String Separation

Validity	Specified	Result	Length
Valid	C'John Doe'	John Doe	8
Valid	C'JOHN DOE'	JOHN DOE	8
Valid	C'$@#'	$@#	3
Valid	C'+0.193'	+0.193	6
Valid	4000C' '	8000 blanks	8000
Valid	20C'**FILLER**'	**FILLER** repeated 20 times	200
Invalid	C'''''	Apostrophes not paired	n/a
Invalid	'ABCDEF'	C identifier missing	n/a
Invalid	C'ABCDE	Apostrophe missing	n/a
Invalid	4450C'1'	Too many repetitions. Use two adjacent separation fields instead (4000C'1',450C'1', for example).	n/a
Invalid	0C'ABC'	0 is not allowed	n/a
Invalid	C''	No characters specified	n/a

nX′yy...yy′ Hexadecimal string separation. n repetitions of the hexadecimal string constant (X′yy...yy′) are inserted in the reformatted input records. n can be from 1 to 4095. If n is omitted, 1 is used.

The value yy represents any pair of hexadecimal digits. You can specify from 1 to 256 pairs of hexadecimal digits.

For example:

Table 7. Examples of Valid and Invalid Hexadecimal String Separation

Validity	Specified	Result	Length
Valid	X'FF'	FF	1
Valid	X'BF3C'	BF3C	2
Valid	3X'00000F'	00000F00000F00000F	9
Valid	4000X'FFFF'	FF repeated 8000 times	8000
Invalid	X'ABGD'	G is not a hexadecimal digit	n/a
Invalid	X'F1F'	Incomplete pair of digits	n/a
Invalid	'BF3C'	X identifier missing	n/a
Invalid	'F2F1'X	X in wrong place	n/a
Invalid	8000X'01'	Too many repetitions. Use two adjacent separation fields instead (4000X'01',4000X'01', for example).	n/a
Invalid	0X'23AB'	0 is not allowed	n/a
Invalid	X''	No hexadecimal digits specified	n/a

p,m,a

indicates an input field to be inserted into the reformatted input record in the position you code it relative to the other fields.

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 can 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 an integer number of bytes. See the note on page 82 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 two (that is, position 1, 3, 5, and so forth).

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

D Doubleword aligned. The displacement is a multiple of eight (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 C, "Specification/Override of DFSORT Options" on page 353.

INREC Statement Notes

- 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.

- When you specify INREC, you must be aware of the change in record size and layout of the resulting reformatted input records. You must also understand how reformatting of records affects sort performance, and how to use INREC or OUTREC to achieve the most efficient sort. (See also "OUTREC Control Statement" on page 121 and "Use Options that Enhance Performance" on page 251 for more details.)

- 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.

- 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.

  If the Blockset technique is not selected and the INREC/OUTREC length is less than the specified or defaulted SORTOUT LRECL, then you must pad the record to the correct length.

  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 bytes less than the LRECL when processing variable-length records. See "VSAM Considerations" on page 10 for more information.

- The variable part of the input record (that part beyond the minimum record length) can be included in the reformatted input record, and if included, must be the last part. In this case, a value must 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* must be omitted.

  If both INREC and OUTREC are 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.

- 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.

- Fields referenced in INREC statements can overlap each other and/or control fields.

- 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.

- In general, use INREC to reduce the length of input records as much as possible to achieve the most efficient processing. Use OUTREC to reformat the records for output. However, if overflow might occur during summation, INREC can 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.

- 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 must 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 might 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. The output records are variable-length, have 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,s─┴──),FORMAT=f─────┤ └─,─┬──CKPT───────────────┼──┘
                │                            │     ├─┬─EQUALS───┬────────┤
                │                            │     │ └─NOEQUALS─┘        │
                └──COPY──────────────────────┘     ├──FILES=n────────────┤
                                                   ├─┬─FILSZ=x─┬─────────┤
                                                   │ └─SIZE=y──┘         │
                                                   ├──SKIPREC=z──────────┤
                                                   └──STOPAFT=n──────────┘
```

The MERGE control statement must be used when a merge operation is to be performed. It can 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 parameters, 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 and STOPAFT options are 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 C, "Specification/Override of DFSORT Options" on page 353.

FIELDS

```
             ┌──,──┐
             ↓     │
►►──FIELDS=(─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 "SORT Control Statement" on page 132.

FIELDS=COPY

```
►►──FIELDS=COPY──────────────────────────────────────────────►◄
```

See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" on page 97.

FORMAT=f

```
►►──FORMAT=f─────────────────────────────────────────────────►◄
```

The FORMAT operand is used in the same way for a merge as for a sort. See "SORT Control Statement" on page 132.

CKPT

```
►►──CKPT─────────────────────────────────────────────────────►◄
```

See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" on page 97.

EQUALS or NOEQUALS

```
►►──┬─EQUALS──┬──────────────────────────────────────────────►◄
    └─NOEQUALS┘
```

See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" on page 97.

FILES

```
►►──FILES=n──────────────────────────────────────────────────►◄
```

specifies the number of input files for a merge when input is supplied through the E32 exit.

Default: None.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

FILSZ or SIZE

```
►►──┬FILSZ=x┬────────────────────────────────────────────────►◄
    └SIZE=y─┘
```

See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" on page 97.

SKIPREC

```
►►──SKIPREC=z──────────────────────────────────────────────────►◄
```

See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" on page 97.

STOPAFT

```
►►──STOPAFT=n──────────────────────────────────────────────────►◄
```

See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" on page 97.

Note: For a merge job, records deleted by an E35 exit routine are not sequence-checked. If you use an E35 exit routine without a SORTOUT data set, sequence-checking is not performed. In this case, you must ensure that input records are in correct sequence.

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

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

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 in descending order.

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─┘
```

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 5, "Using Your Own Exit Routines" on page 181.

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

exit

```
►►──────────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 can 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.

If you do not specify a value for s, DFSORT uses the following search order to find the library in which your routine is located:

1. The libraries identified by the STEPLIB DD statement
2. The libraries identified by the JOBLIB DD statement (if there is no STEPLIB DD statement)
3. The link library.

e

indicates the linkage editor requirements of your routine, or indicates your routine is written in COBOL. e can take the following values:

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) can 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.

Default: None. Must be specified. N is the default for the fourth parameter.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- The s parameter must be the same or omitted for each routine with N or C for the e parameter (library concatenation is allowed). These routines cannot be placed in SYSIN. Each such routine must be a load module.
- Each routine for which T or S is specified for the e parameter can 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.
- 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.
- 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.)

- If you code C for a conventional merge or a tape work data set sort, DFSORT issues message ICE153A and terminates.
- exit=(n,m) can be used to omit both the s and e parameters.
- exit=(n,m,,e) can be used to omit the s parameter, but not the e parameter.
- The s parameter must be specified for a conventional merge or tape work data set sort, or when S or T is specified for the e parameter.

For information on user exit routines in SYSIN, see "System DD Statements" on page 39.

For details on how to design your routines, refer to "Summary of Rules for User Exit Routines" on page 189.

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.

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,,C),
       E35=(COBOLE35,7000,EXITC,C)
```

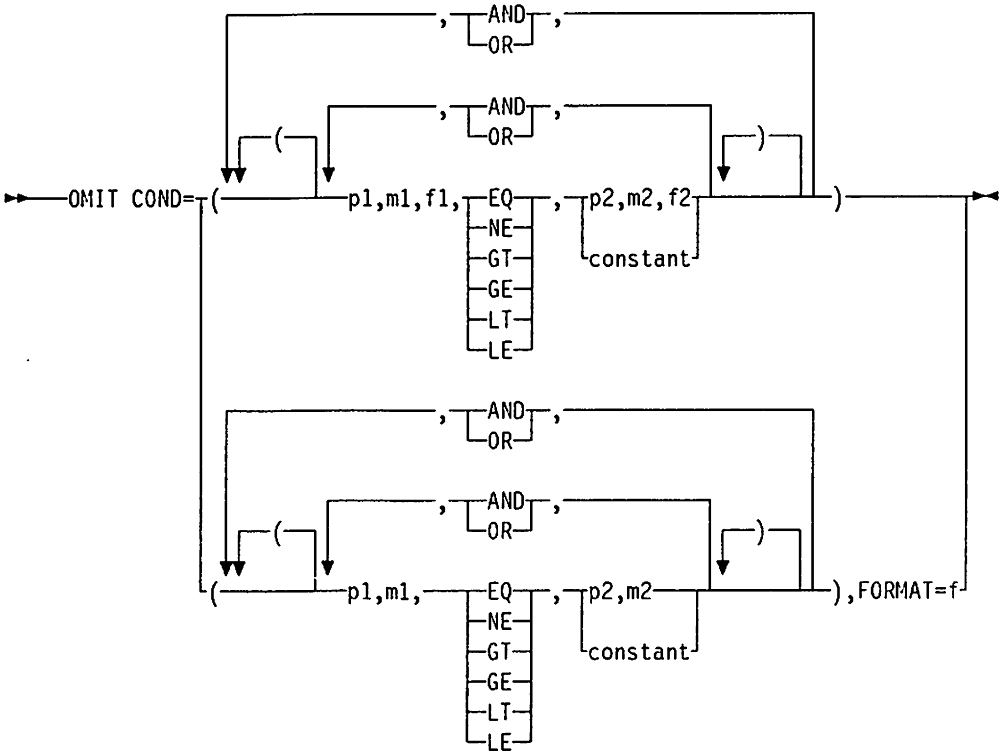
E15

At exit E15, DFSORT transfers control to your own routine. Your routine is written in COBOL and is in the STEPLIB/JOBLIB or link libraries. Its member name is COBOLE15 and it 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 it uses 7000 bytes.

OMIT Control Statement



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 can 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 68.

OMIT Statement Example

OMIT Example.

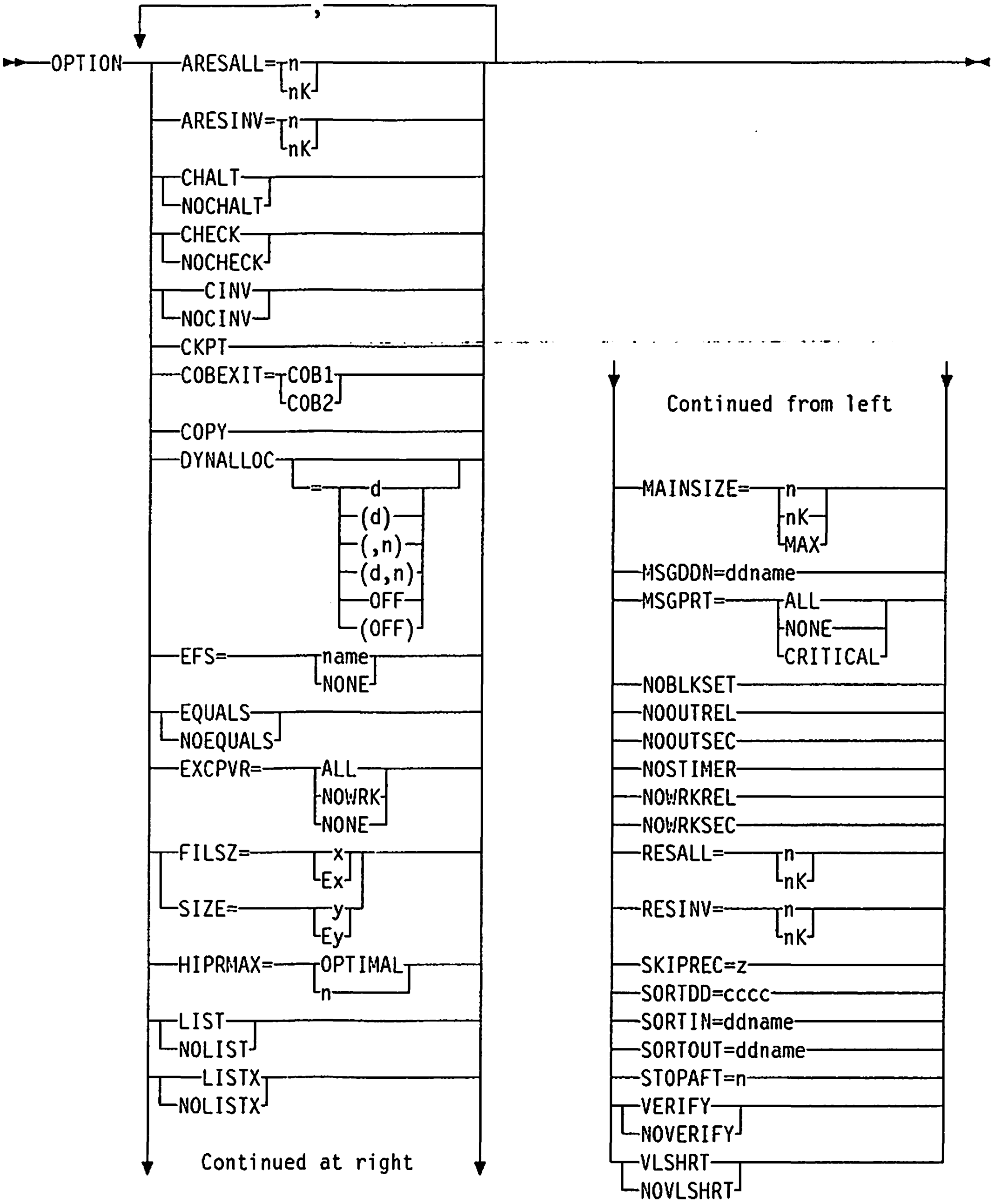
```
16                                                       72
|                                                         |
v                                                         v
OMIT COND=(1,10,CH,EQ,C'STOCKHOLM',&,(21,8,ZD,GT,+50000,|*
 ,31,4,CH,NE,C'HERR'))
```

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



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 C, "Specification/Override of DFSORT Options" on page 353.

OPTION parameters used by other IBM sort programs cause DFSORT to terminate unless they conform to the following parameters.

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

ARESALL

```
►►──ARESALL=─┬─n──┬──────────────────────────────────────────►◄
             └─nK─┘
```

For MVS/XA and MVS/ESA, temporarily overrides the ARESALL installation option, which specifies 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

specifies the number of bytes of main storage to be reserved.

Limit: 8 digits.

nK

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 C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

ARESINV

```
►►──ARESINV=─┬─n──┬──────────────────────────────────────────►◄
             └─nK─┘
```

For MVS/XA and MVS/ESA, temporarily overrides the ARESINV installation option, which specifies 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 invoking program itself. *ARESINV is used only when DFSORT is dynamically invoked.*

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

specifies the number of bytes of main storage to be reserved.

Limit: 8 digits.

nK

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 C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

CHALT or NOCHALT

```
►►──┬─CHALT──┬──────────────────────────────────────────►◄
    └─NOCHALT┘
```

Temporarily overrides the CHALT installation option, which 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 are not translated.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

CHECK or NOCHECK

```
►►──┬─CHECK──┬──────────────────────────────────────────►◄
    └─NOCHECK┘
```

Temporarily overrides the CHECK installation option, which 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 C, "Specification/Override of DFSORT Options" on page 353 full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

CINV or NOCINV

```
►►──────────┬─CINV───┬─────────────────────────────────►◄
            └─NOCINV─┘
```

Temporarily overrides the CINV installation option, which specifies whether DFSORT can use control interval access for VSAM data sets. The Blockset technique uses control interval access for VSAM input data sets, when possible, to improve performance.

CINV
directs DFSORT to use control interval access when possible for VSAM data sets.

NOCINV
directs DFSORT not to use control interval access.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

CKPT

```
►►──CKPT───────────────────────────────────────────────►◄
```

CKPT causes DFSORT to activate the checkpoint/restart facility of the operating system. See "Checkpoint/Restart" on page 383 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 48. In addition, you might need to specify more intermediate storage for a sort application. See "Intermediate Storage" on page 311.

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: CHKPT can be used instead of CKPT.

COBEXIT

```
►►──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 C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

COPY

```
►►──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 in copy jobs:

- BDAM data sets
- Dynamic link-editing.

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

DYNALLOC

```
►►──DYNALLOC─┬──────────────┬───────────────────────────────►◄
             └─=─┬──d────┬──┘
                 ├─(d)───┤
                 ├─(,n)──┤
                 ├─(d,n)─┤
                 ├──OFF──┤
                 └─(OFF)─┘
```

Use this parameter to assign DFSORT the task of dynamically allocating needed work space. You do not need to calculate and use JCL to specify the amount of intermediate work space needed by the program. DFSORT uses the dynamic allocation facility of the operating system to allocate work space to get the best possible performance for the newest application.

d

specifies the device type. You can 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 maximum number of requested work data sets. The maximum value of n is 16; if you specify more than 16, 16 is used. If you specify 1 and the Blockset technique is selected, 2 is used.

For disk work data sets, an estimate of the number of input records and FILE/SIZE are used as the basis for determining the total work space to allocate. If DFSORT cannot reasonably estimate the number of input records (for example, if the input data set is on tape) and FILSZ/SIZE is not specified (see FILSZ/SIZE on the OPTION statement for details), 6000 blocks are dynamically allocated. For variable-length records, DFSORT uses the length of the longest input record to determine the total amount of work space to allocate.

When using DYNALLOC to request dynamic allocation with multiple invocations of DFSORT within the same step, make sure that the first DYNALLOC request is large enough for all the sorts. Dynamically allocated work data sets are not deallocated until the step is finished to ensure that SMF logs data set usage correctly. As a result, in this situation, DFSORT reuses 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.

DYNALLOC is not used if SORTWKnn DD statements are provided.

If VIO=NO is in effect

- Work space is allocated on nontemporary data sets (DSNAME parameter specified).
- If the device (d) you specify is a virtual device, DFSORT will ignore VIO=NO and use the virtual device.

Default: None; optional.

- If DYNALLOC is not specified, and no SORTWKnn DD statements are present, program execution proceeds according to the specification set during installation with the ICEMAC DYNAUTO option (YES for dynamic allocation, NO for no dynamic allocation).
- If DYNALLOC is specified without d, the default for d is that specified (or defaulted) by the ICEMAC DYNALOC option during installation.
- If DYNALLOC is specified without n, the default for n is that specified (or defaulted) by the ICEMAC DYNALOC option during installation.

You can specify DYNALLOC without n, without d, or without both. If DYNALLOC is specified without n, and the default for the n value of the DYNALOC installation option is chosen, then at execution:

- If one of the Blockset techniques is chosen, two work data sets will be requested.
- If a technique other than Blockset is chosen, three work data sets will be requested.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

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

DYNALLOC=OFF

```
►►──DYNALLOC=┬─OFF──┬──────────────────────────────────────►◄
             └(OFF)─┘
```

Directs DFSORT *not* to allocate intermediate workspace dynamically, overriding the ICEMAC installation option DYNAUTO=YES, or the DYNALLOC parameter (without OFF) specified at execution. Use this option when you know that an in-core sort can be performed, and you want to suppress dynamic allocation of workspace.

OFF
directs DFSORT not to allocate intermediate workspace dynamically.

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

EFS

```
►►──EFS=┬name┬─────────────────────────────────────────────►◄
        └NONE┘
```

Temporarily overrides the EFS installation option, which specifies whether DFSORT is to pass control to an Extended Function Support (EFS) program. See Appendix B, "Using Extended Function Support (EFS)" on page 315 for more information.

name
the name of the EFS program that will be called to interface with DFSORT.

NONE
means no call will be made to the EFS program.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

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

EQUALS or NOEQUALS

```
►►───────────┬─EQUALS───┬──────────────────────────────────────►◄
             └─NOEQUALS─┘
```

Temporarily overrides the EQUALS installation option which specifies whether the original sequence of identical collating records for a sort or a merge should be preserved from input to output.

EQUALS
specifies that the original sequence must be preserved.

NOEQUALS
specifies that the original sequence need not be preserved.

For sort jobs, the sequence of the records in the output file 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.

For merge jobs, the sequence of the records in the output file depends upon the order of:

- The records from a SORTINnn file. Equally-collating records are output in the order of their file increments. For example, records from SORTIN01 are output before any equally collating records from SORTIN02.
- The records from an E32 user exit routine for the same file increment number. Equally-collating records from E32 are output in the order of their file increments. For example, records from m1 are output before any equally-collating records from m2.

Default: Usually the installation default. See Appendix C, "Specification/Override of DFSORT Options" on page 353 for more information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options."

Note:

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

EXCPVR

```
►►──EXCPVR=┬ALL───┬──────────────────────────────────►◄
           ├NOWRK─┤
           └NONE──┘
```

For MVS/XA and MVS/ESA, temporarily overrides the EXCPVR installation option, which specifies the data sets for which DFSORT can use EXCPVR when sorting or copying fixed- and variable-length records with the Blockset technique. EXCPVR can reduce the CPU time required for a run.

EXCPVR can only be used if the DFSORT SVC is installed.

Using EXCPVR=ALL provides the largest reduction in required CPU time, but can cause other jobs to be paged in and out. Using EXCPVR=NOWRK causes less page fixing than EXCPVR=ALL, but still improves DFSORT performance. Use EXCPVR=NONE if avoiding higher paging rates is more important than reducing CPU time.

Note: The EXCPVR option is not supported for MVS/370.

ALL
specifies that DFSORT can use EXCPVR for SORTIN, SORTOUT, and SORTWKnn data sets.

NOWRK
specifies that DFSORT can use EXCPVR for the SORTIN and SORTOUT data sets only. DFSORT will not use EXCPVR for SORTWKnn if NOWRK is specified.

NONE
specifies that DFSORT cannot use EXCPVR for SORTIN, SORTOUT, or SORTWKnn data sets.

Default: Usually the installation default, but see Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: EXCPVR=ALL is treated like EXCPVR=NOWRK for Hipersorting and copying. DFSORT does not use EXCPVR when it determines that the associated overhead might cause performance to be degraded.

FILSZ or SIZE

```
►►─┬─FILSZ=─┬─x──┬─┬──────────────────────────────►◄
   │        └─Ex─┘ │
   └─SIZE=──┬─y──┬─┘
            └─Ey─┘
```

This parameter specifies an exact number of records for a sort or merge job, or an estimated number of records for a sort job. 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, it uses the estimated FILSZ or SIZE value to aid optimization of intermediate storage; otherwise, it does not use the value for this purpose. For variable-length records, DFSORT uses the length of the longest input record to determine the total amount of workspace to allocate. Following are the circumstances under which DFSORT uses the estimated FILSZ or SIZE (if specified) for optimization:

- An E15 exit routine is used.
- Input data sets are multivolume or on tape.
- Input data sets are concatenated and Blockset is not selected.

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=Ex or SIZE=Ey

x or y is the estimated number of records to be sorted; it must be immediately preceded by the letter E. In either case, it must be large enough to include both the SORTIN data set and any records you add at exit E15.

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 to be sorted.

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

HIPRMAX

```
►►──HIPRMAX=┬OPTIMAL┬──────────────────────────────────►◄
            └n──────┘
```

For MVS/ESA, temporarily overrides the HIPRMAX installation option, which specifies the maximum amount of hiperspace to be committed for Hipersorting. A hiperspace is a high-performance data space that resides in expanded storage and is backed by auxiliary storage (if necessary). Because I/O processing is reduced for Hipersorting, elapsed time, EXCP counts, and channel usage are also reduced.

The amount of hiperspace available for a job is limited by the IEFUSI exit your site uses. When HIPRMAX=n is specified, DFSORT will not use more storage than the value of HIPRMAX. When HIPRMAX=OPTIMAL is specified, DFSORT selects the amount of hiperspace that will cause minimal impact on system paging activity.

If the amount of hiperspace available for Hipersorting is insufficient for temporary storage of the records, intermediate DASD storage will be used along with hiperspace. If the amount of hiperspace is too small to improve performance, Hipersorting will not be used. DYNAUTO=YES will always be in effect for Hipersorting.

Hipersorting might cause a small CPU time degradation. When CPU optimization is a concern, you can use HIPRMAX=0 to suppress Hipersorting.

OPTIMAL
specifies that DFSORT determines dynamically the maximum amount of hiperspace to be committed for Hipersorting. DFSORT determines the paging activity at the start of the run and selects the maximum value that causes minimal system impact.

n
specifies the maximum megabytes of hiperspace to be committed for Hipersorting. n must be a value between 0 and 9999. The actual amount of hiperspace used will not exceed the HIPRMAX value, but might be less if DFSORT determines that using the maximum will increase system paging significantly. If n is 0 (zero), Hipersorting is not used.

Default: Usually the default set during installation, but see Appendix C, "Specification/Override of DFSORT Options" on page 353 for override information.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

LIST or NOLIST

```
►►──┬─LIST───┬──────────────────────────────────────────►◄
    └─NOLIST─┘
```

Temporarily overrides the LIST installation option, which specifies whether DFSORT program control statements should be written to the message data set. See *DFSORT Messages and Codes* for details on use of the message data set.

LIST
means that DFSORT control statements are printed to the message data set.

NOLIST
means that DFSORT control statements are not printed to the message data set.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note: LIST or NOLIST are processed only if they are passed on the OPTION control statement in an extended parameter list or in DFSPARM.

LISTX or NOLISTX

```
►►──┬─LISTX───┬──────────────────────────────────────────►◄
    └─NOLISTX─┘
```

Temporarily overrides the LISTX installation option, which specifies whether DFSORT writes to the message data set program control statements that are returned by an EFS program. See *DFSORT Messages and Codes* for details on use of the message data set.

LISTX
means that control statements returned by an EFS program are to be printed to the message data set.

NOLISTX
means that control statements returned by an EFS program are not to be printed to the message data set.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- LISTX or NOLISTX are processed only if they are passed on the OPTION control statement in an extended parameter list or in DFSPARM.
- If EFS=NONE is in effect after final override rules have been applied, NOLISTX will be set in effect.
- LISTX and NOLISTX can be used independently of LIST and NOLIST.
- For more information on printing EFS control statements, see *DFSORT Messages and Codes.*

MAINSIZE

```
►►──MAINSIZE=─┬─n───┬────────────────────────────────────►◄
              ├─nK──┤
              └─MAX─┘
```

Temporarily overrides the SIZE installation option, 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 and MVS/ESA 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 cannot exceed MAINSIZE.

Specifying a MAINSIZE value equal to your installation's MAXLIM value is equivalent to specifying MAINSIZE=MAX. If the MAINSIZE or SIZE passed to DFSORT is equal to the MAXLIM, the MAXLIM value will be used and DFSORT will set "MAX" in effect.

For details on main storage allocation, see "Tuning Main Storage" on page 256 and "Main Storage" on page 311.

n

specifies the number of bytes of main storage to be allocated. You can specify a value greater than MAXLIM or TMAXLIM.

Limit: 8 digits

nK

specifies n times 1024 bytes of main storage to be allocated. You can 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 and MVS/ESA systems) value set when DFSORT was installed.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

MSGDDN

```
►►──MSGDDN=ddname──────────────────────────────────────────►◄
```

Temporarily overrides the MSGDDN installation option, 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 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 one through eight-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 *DFSORT Messages and Codes*.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

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

MSGPRT

```
►►──MSGPRT=─┬─ALL──────┬──────────────────────────────────►◄
            ├─CRITICAL─┤
            └─NONE─────┘
```

Temporarily overrides the MSGPRT installation option, which specifies the class of messages to be written to the message data set. For details on use of the message data set, see *DFSORT Messages and Codes*.

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.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

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

NOBLKSET

```
►►──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 C, "Specification/Override of DFSORT Options" on page 353.

NOOUTREL

```
►►──NOOUTREL──────────────────────────────────────────────►◄
```

Temporarily overrides the OUTREL installation option, which specifies whether 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 C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

NOOUTSEC

```
►►──NOOUTSEC──────────────────────────────────────────►◄
```

Temporarily overrides the OUTSEC installation option, which specifies whether automatic secondary allocation is to 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 C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

NOSTIMER

```
►►──NOSTIMER──────────────────────────────────────────►◄
```

Temporarily overrides the STIMER installation option, which specifies whether DFSORT can use the STIMER macro. NOSTIMER means that DFSORT does not use the STIMER macro; processor time data does not appear in SMF records or in statistics provided to the ICETEXIT termination installation exit.

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

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

NOWRKREL

```
►►──NOWRKREL──────────────────────────────────────────►◄
```

Temporarily overrides the WRKREL installation option, which specifies whether 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 C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

NOWRKSEC

```
►►──NOWRKREL──────────────────────────────────────────────────►◄
```

Temporarily overrides the WRKSEC installation option, which specifies whether automatic secondary allocation is to 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 C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

RESALL

```
►►──RESALL=─┬─n──┬────────────────────────────────────────────►◄
            └─nK─┘
```

Temporarily overrides the RESALL installation option, which specifies the number of bytes to be reserved in a REGION for system use. Usually, only 4K bytes (the standard default) of main storage must be available in a region for system use. However, in some cases, this might 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. RESALL is used only when MAINSIZE/SIZE=MAX is in effect.

For MVS/XA and MVS/ESA 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

specifies the number of bytes of storage to be reserved. If you specify less than 4096, 4096 is used.

Limit: 8 digits.

nK

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

Limit: 5 digits.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

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

RESINV

```
►►──RESINV=─┬─n──┬──────────────────────────────────────────►◄
            └─nK─┘
```

Temporarily overrides the RESINV installation option, which specifies the number of bytes to be reserved in a REGION for the invoking program. RESINV is used only when DFSORT is dynamically invoked and MAINSIZE/SIZE=MAX is in effect.

For MVS/XA or MVS/ESA 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

nK

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 C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

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

SKIPREC

```
►►──SKIPREC=z───────────────────────────────────────────────►◄
```

z is the number of records you want to skip before starting to sort or copy 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 job with an input data set that exceeds intermediate storage capacity usually terminates unsuccessfully. However, for a tape work data set sort, you can use a routine at E16 (as described in Chapter 5, "Using Your Own Exit Routines" on page 181) 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 bypass the previously-sorted records, sort only the remaining records, and then merge the output from different runs to complete the application.

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- SKIPREC applies only to records read from SORTIN (not from E15 routines). (See Figure 2 on page 6.)
- If SKIPREC=0 is in effect, SKIPREC is not used.

SORTDD

```
►►──SORTDD=cccc──────────────────────────────────────────────►◄
```

You should use this parameter to specify a four-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 four-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.

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

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- SORTDD is processed only if it is passed on the OPTION control statement in an extended parameter list, or in DFSPARM.
- If both SORTIN=ddname and SORTDD=cccc are specified, ddname is used for DFSORT input.
- If both SORTOUT=ddname and SORTDD=cccc are specified, ddname is used for DFSORT output.

SORTIN

```
►►──SORTIN=ddname────────────────────────────────────────────►◄
```

specifies 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.

The DDNAME can be 1 through 8 characters, but must be unique within the job step. Do *not* use DDNAMEs reserved for use by DFSORT (such as SYSIN).

Default: If this parameter is not specified, ddname defaults to SORTIN, unless SORTDD=cccc is specified, unless ccccIN is the default.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- SORTIN is processed only if it is passed on the OPTION control statement in an extended parameter list, or in DFSPARM.
- If both SORTIN=ddname and SORTDD=cccc are specified, ddname is used for the input file. The same DDNAME cannot be specified for SORTIN and SORTOUT.
- If SORTIN is used for a tape work data set sort, DFSORT terminates.

SORTOUT

```
►►──SORTOUT=ddname──────────────────────────────────────────►◄
```

specifies 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.

The DDNAME can be 1 through 8 characters, but must be unique within the job step. Do *not* use DDNAMEs reserved for use by DFSORT (such as SYSIN).

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

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- SORTOUT is processed only if it is passed on the OPTION control statement in an extended parameter list, or in DFSPARM.
- If both SORTOUT=ddname and SORTDD=cccc are specified, ddname is used for the output file. The same DDNAME cannot be specified for SORTIN and SORTOUT.
- If SORTOUT is specified for a conventional merge or for a tape work data set sort, DFSORT terminates.

STOPAFT

```
►►──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.

Default: None; optional.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- STOPAFT is not used for a tape work data set sort.
- If you specify the FILSZ=x PARM, or FILSZ=x or SIZE=x on the OPTION or SORT statement, and the number of records accepted for processing does not equal x, DFSORT issues message ICE047A and terminates.
- If STOPAFT=0 is in effect, STOPAFT is not used.

VERIFY or NOVERIFY

```
►►──┬─VERIFY──┬──────────────────────────────────────────►◄
    └─NOVERIFY┘
```

Temporarily overrides the VERIFY installation option, which specifies whether sequence checking of the final output records must be performed.

VERIFY
specifies that sequence checking is to be performed.

NOVERIFY
specifies that sequence checking is not to be performed.

Note: Use of VERIFY can degrade performance.

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

VLSHRT or NOVLSHRT

```
►►──┬─VLSHRT──┬──────────────────────────────────────────►◄
    └─NOVLSHRT┘
```

Temporarily overrides the VLSHRT installation option, which specifies whether DFSORT is to continue sorting or merging if a variable-length input record is found to be too short to contain all specified control fields. VLSHRT is not meaningful for fixed-length record processing.

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

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

Default: Usually the installation default, but refer to Appendix C, "Specification/Override of DFSORT Options" on page 353 for full override details.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- VLSHRT is not used if INCLUDE/OMIT, INREC, OUTREC, and/or SUM are specified, or if you have an EFS01 routine.
- 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.
- 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 because of the number and position of the control fields.
  - EQUALS is not used if VLSHRT is in effect.

OPTION Statement Examples

OPTION Statement Example 1. One Control Field and Related Options

```
SORT   FIELDS=(1,20,CH,A)
OPTION SIZE=50000,SKIPREC=5,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 in 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.

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

DYNALLOC
Two data sets (by default) are allocated on SYSDA (by default). The space on the data set is calculated using the 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 in 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 equally-collating records is preserved from input to output.

NOCHALT
Only AQ fields are translated through the ALTSEQ translate table. If CHALT=YES was specified during installation, then NOCHALT temporarily overrides it.

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 two work data sets (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 does not use the Blockset technique for a sort or merge.

OPTION Example 5. Using STOPAFT and COBEXIT

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

STOPAFT
DFSORT accepts 100 records before sorting or copying.

COBEXIT
E15 and 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 DFSPARM

```
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 DFSPARM. These options can also be passed in an extended parameter list, with the appropriate syntax changes.

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)
```

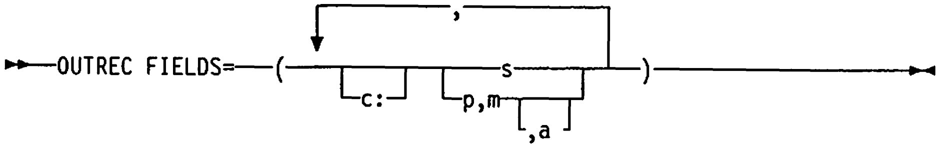
SORT
The sort statement is ignored because the COPY option has been specified.

COPY
The copy processing is always done on a record-by-record basis. Each record is therefore 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



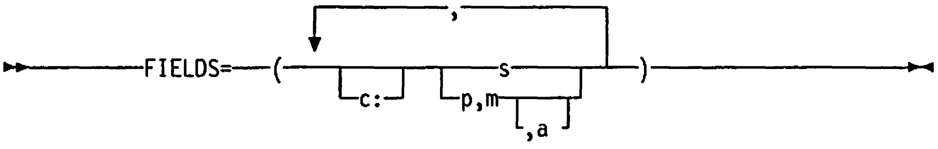
The OUTREC control statement allows you to reformat the input records before they are output; that is, to define which parts of the input record are to be included in the reformatted output record, in what order they are to appear, and how they are to be aligned.

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 or in the columns you have indicated.

You can also insert blanks, binary zeros, character strings, and hexadecimal strings as separators before, between, and after the input fields, in the reformatted output records.

For information concerning the interaction of INREC and OUTREC, see "Use Options that Enhance Performance" on page 251.

FIELDS



specifies the order and alignment of the input and separation fields in the reformatted output records.

c:

Indicates the column in which the first position of the associated input or separation field is to be aligned, relative to the start of the reformatted output record. Unused space preceding the specified column is padded with EBCDIC blanks. The following rules apply:

- c must be a number between 1 and 32000.
- c: must be followed by an input field or a separation field.
- c must not overlap the previous input field or separation field in the reformatted output record.
- For variable-length records, c: must not be specified before the first input field (the record descriptor word), nor after the variable part of the input record.
- The colon (:) is treated like the comma (,) or semicolon (;) for continuation to another line.

See Table 3 on page 79 for examples of valid and invalid column alignment.

s

Indicates a separation field to be inserted into the reformatted output record in the position you code it relative to the other fields. It can be specified before or after any input field. Consecutive separation fields may be specified. For variable length records, s must not be specified before the first input field (the record descriptor word), or after the variable part of the input record. Permissible values are:

nX Blank separation. n bytes of EBCDIC blanks (X'40') are inserted in the reformatted output records. n can be from 1 to 4095. If n is omitted, 1 is used.

See Table 4 on page 79 for examples of valid and invalid blank separation.

nZ Binary zero separation. n bytes of binary zeros (X'00') are inserted in the reformatted output records. n can be from 1 to 4095. If n is omitted, 1 is used.

See Table 5 on page 79 for examples of valid and invalid binary zero separation.

nC'xx...x' Character string separation. n repetitions of the character string constant (C'xx...x') are inserted into the reformatted output records. n can be from 1 to 4095. If n is omitted, 1 is used. x can be any EBCDIC character. You can specify 1 to 256 characters.

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

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

See Table 6 on page 80 for examples of valid and invalid character string separation.

nX'yy...yy' Hexadecimal string separation. n repetitions of the hexadecimal string constant (X'yy...yy') are inserted in the reformatted output records. n can be from 1 to 4095. If n is omitted, 1 is used.

The value yy represents any pair of hexadecimal digits. You can specify from 1 to 256 pairs of hexadecimal digits.

See Table 7 on page 80 for examples of valid and invalid hexadecimal string separation.

p,m,a

Indicates an input field to be inserted into the reformatted output record in the position you code it relative to the other fields.

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 the following "OUTREC Statement Notes" 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 124 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,ˊand 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 C, "Specification/Override of DFSORT Options" on page 353.

OUTREC Statement Notes

- If input records are reformatted by INREC or E15, OUTREC must refer to fields in the appropriate reformatted record (see the discussion of *p* on page 122).
- When you specify OUTREC, you must be aware of the change in record size and layout of the resulting reformatted output records. You must also understand how reformatting of records affects processing performance, and how to use INREC and/or OUTREC to achieve the most efficient processing. (See also "INREC Statement Notes" on page 81 and "Use Options that Enhance Performance" on page 251 for more details).
- 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.

[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.

If the Blockset technique is not selected and the INREC/OUTREC length is less than the specified or defaulted SORTOUT LRECL, then you must pad the record to the correct length.

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 bytes more than the LRECL when processing variable-length records. See "VSAM Considerations" on page 10 for more information.

- 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.

- The variable part of the input record (that part beyond the minimum record length) can be included in the reformatted output record as the last part. In this case, a value must 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* must 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 might result.

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

- Fields referenced in OUTREC statements can overlap each other and/or control fields.

- 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.

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

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

- When you specify OUTREC, VLSHRT is not used. If VLSHRT 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 is to contain 32 bytes beginning with byte 11 of the input record. This statement can be used only 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 is to 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=(7:20,4,C' FUTURE ',20,2,10,3,1Z,1,9,13,7,24,57,6X'FF')
```

This example illustrates how a fixed-length input data set can 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 must be specified as 103.

Position	**Contents**
1-6	EBCDIC blanks for column alignment
7-10	Input positions 20 through 23
11-18	Character string: C' FUTURE '
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	Hexadecimal string: X'FFFFFFFFFFFF'

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 can 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.

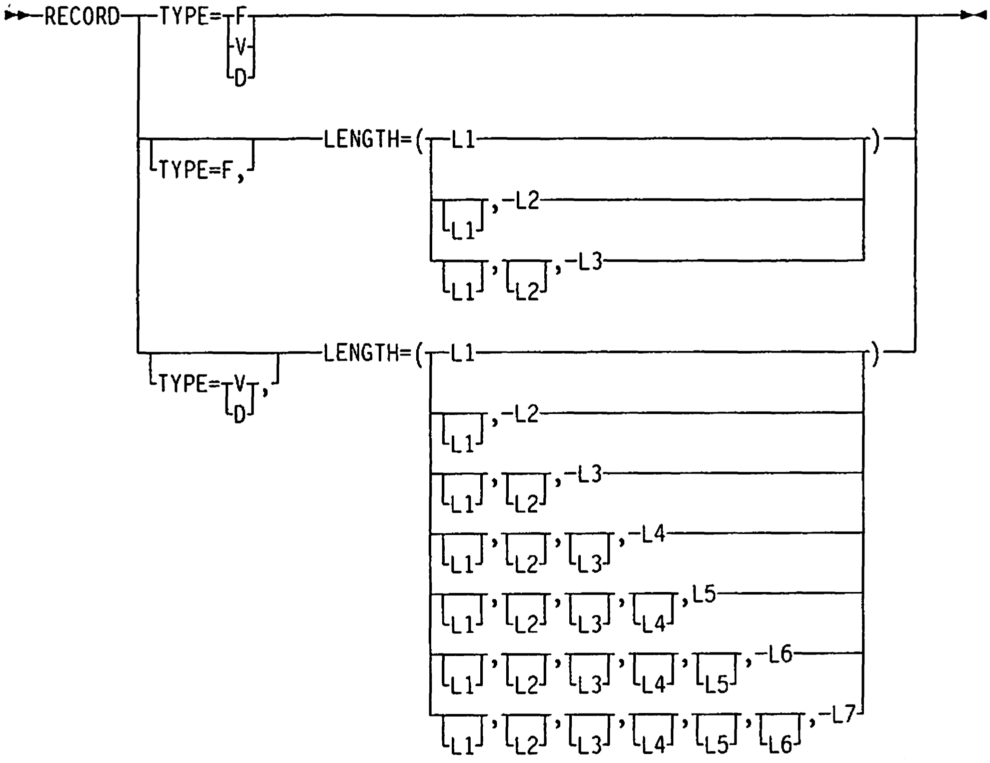
```
MERGE FIELDS=(28,4,BI,A)
OUTREC FIELDS=(1,4,5Z,5X,5,3,28,8,6Z)
```

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



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.
- All of the input is supplied through a user exit.
- Input and output record length information is unavailable.
- A sort is invoked from a program written in PL/I.
- VSAM input and VSAM output data sets are used.

Note: L4 through L7 apply only to variable-length records.

If you are working with VSAM input and non-VSAM output data sets, DFSORT requires the RECORD TYPE=x statement only for tape work data set sorts or Conventional merges. If you do not specify the record type for VSAM input and non-VSAM output, DFSORT continues processing using:

- The record type from the non-VSAM output data set.
- A record type of F (for *fixed*), if the record type is not available from the non-VSAM output data set.
- A record type of F (for *fixed*), if there is no output data set.

If you do specify a record type with a VSAM input data set, DFSORT uses your specified record type to process your data sets.

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

TYPE

```
►►──TYPE=x────────────────────────────────────────────────►◄
```

x can take the following values:

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

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

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

DFSORT accepts the alternate values FB, VB, and DB as equivalents for F, V, and D, respectively.

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 2, "Executing DFSORT with Job Control Language" on page 15), 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.

Default: Required when SORTIN or SORTINnn RECFM is unavailable; otherwise, defaults to SORTIN or SORTINnn record format.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

LENGTH

```
►►──────────LENGTH=(─┬─L1───────────────────────────────┬─)──────────►◄
                     │                                  │
                     ├─┬──┬─,─L2────────────────────────┤
                     │ └L1┘                             │
                     ├─┬──┬─,─┬──┬─,─L3─────────────────┤
                     │ └L1┘   └L2┘                      │
                     ├─┬──┬─,─┬──┬─,─┬──┬─,─L4──────────┤
                     │ └L1┘   └L2┘   └L3┘               │
                     ├─┬──┬─,─┬──┬─,─┬──┬─,─┬──┬─,L5────┤
                     │ └L1┘   └L2┘   └L3┘   └L4┘        │
                     ├─┬──┬─,─┬──┬─,─┬──┬─,─┬──┬─,─┬──┬─,─L6─┤
                     │ └L1┘   └L2┘   └L3┘   └L4┘   └L5┘      │
                     └─┬──┬─,─┬──┬─,─┬──┬─,─┬──┬─,─┬──┬─,─┬──┬─,─L7┘
                       └L1┘   └L2┘   └L3┘   └L4┘   └L5┘   └L6┘
```

This parameter is required when you change record lengths at one or more exits, or when the SORTIN or SORTINnn record length (LRECL or VSAM RECSZ) is unavailable.

Note: L4 through L7 apply only to variable-length records.

L1

Input record length, L1, is required only when the SORTIN or SORTINnn record length is unavailable. 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 bytes more than the value defined in the cluster. See "VSAM Considerations" on page 10 for more information.

Note that parentheses are optional if L1 alone is specified. If any of L2 through L7 are specified with or without L1, then parentheses are required.

Default: The SORTIN or SORTINnn record length.

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.

Default: Required when E15 changes the record length; otherwise defaults to L1.

L3

Output record length, L3, can usually be supplied by default: you need to specify L3 when the SORTOUT record length is unavailable, 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.

Default: SORTOUT record length, if available; otherwise defaults to L2.

L4

Specifying the minimum record length, L4, can help performance. However, if you specify too large a value, the program fails and issues message ICE015A.

Default: 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. L5 is not used for the Blockset technique.

Default: None; optional.

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

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

Note:

- You can drop values from the right. For example, LENGTH=(80,70,70,70).
- You can omit values from the middle or left, provided you indicate their omission by a comma or semicolon. For example, LENGTH=(,,,30,80).
- Parentheses are optional when L1 alone is specified. If any of L2 through L7 is specified, with or without L1, then parentheses are required.

RECORD Statement Examples

RECORD Example 1. Fixed-Length, Three Length Values

```
MODS E15=(INEX,1000,EXIT),E35=(OUTEX,2000,EXIT)
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)
```

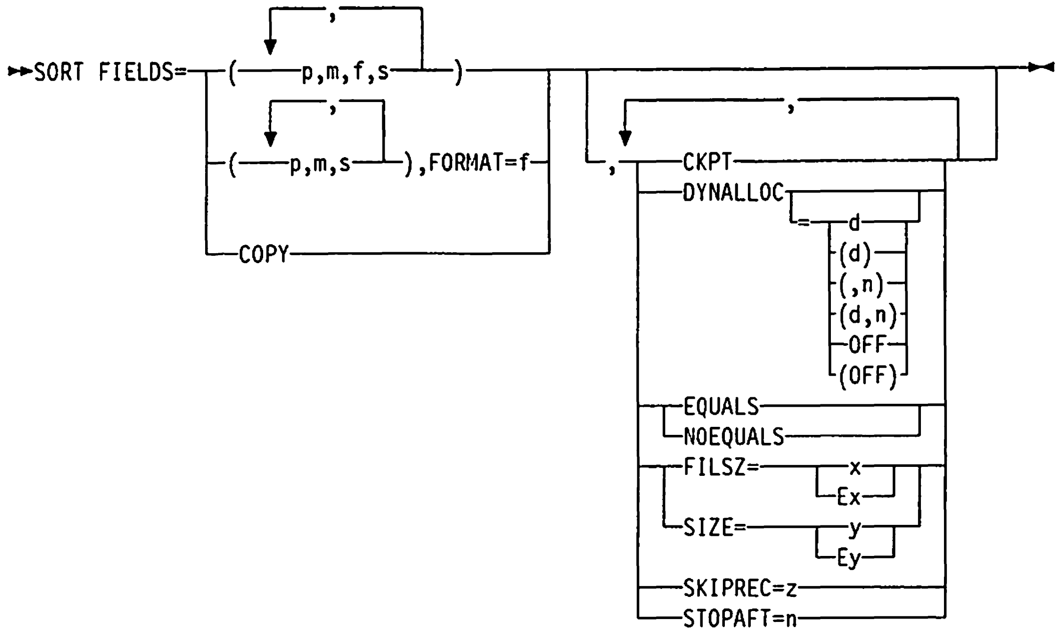
TYPE

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



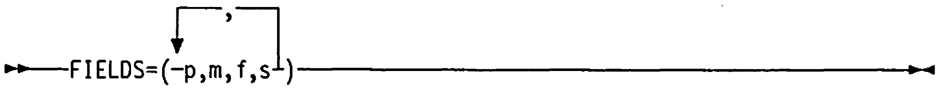
The SORT control statement must be used when a sorting application is to be performed; this statement describes the control fields in the input records on which the program sorts.

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 C, "Specification/Override of DFSORT Options" on page 353.

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

FIELDS



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

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). 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 (explained on page 136) 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. p, m, f, and s describe the control fields. The text that follows gives 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. The first 4 bytes contain the record descriptor word. 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 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:

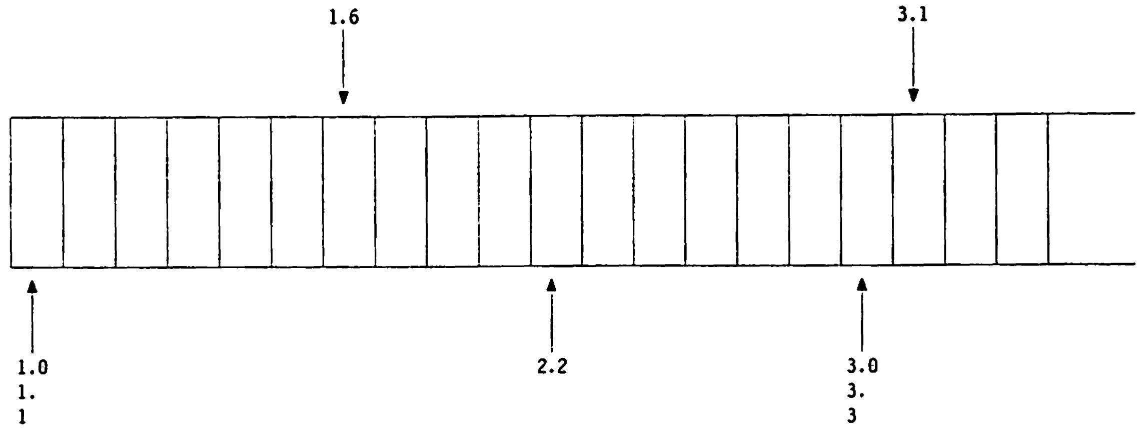
1. First, specify the byte location relative to the beginning of the record and follow it with a period.
2. Then, specify the bit location relative to the beginning of that byte. Remember that the first (high-order) bit of a byte is bit 0 (not bit 1); the remaining bits are numbered 1 through 7.

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
```

Other examples of this notation are:

[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.



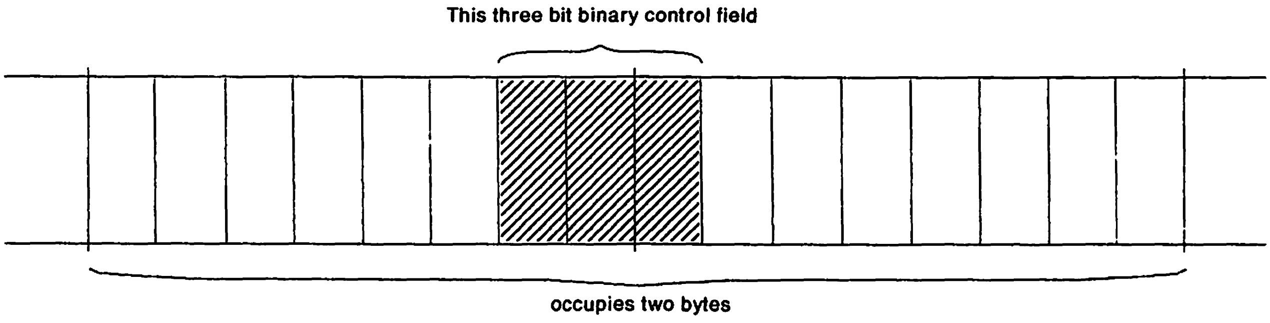
m

specifies the length of the control field. Values for all control fields except binary fields must be expressed in integer numbers of bytes. Binary fields can be expressed in the notation "bytes.bits". The length of a binary control field that is an integer value (d) 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 Table 8 on page 135.

Table 8. Control Field Formats and Lengths

Format	**Length**	**Description**
CH	1-4092	Character EBCDIC, unsigned. If CHALT is in effect, CH is treated the same as AQ.
AQ	1-256	Character EBCDIC, alternate collating sequence.
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 or LS	2-256	Signed numeric, leading separate sign.
CST or TS	2-256	Signed numeric, trailing separate sign.
CLO or OL	1-256	Signed numeric, leading overpunch sign.
CTO or OT	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.
D1	1-4092	User-defined data type; requires an EFS program.

The AC format sequences EBCDIC data using the ISCII/ASCII collating sequence.

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 during execution, 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 E, "EBCDIC and ISCII/ASCII Collating Sequences" on page 377 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 5, "Using Your Own Exit Routines" on page 181.

Default: None; must be specified.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

FORMAT

```
►►──FORMAT=f──────────────────────────────────────────────►◄
```

f can be used to specify the format of the data described in the FIELDS parameter, if you specify more than one control field and those control fields are not all of the same format. The possible values of f are listed in Table 8 on page 135.

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 C, "Specification/Override of DFSORT Options" on page 353.

FIELDS=COPY

```
►►──FIELDS=COPY───────────────────────────────────────────►◄
```

See the discussion of the COPY parameter on the OPTION statement, discussed in "OPTION Control Statement" on page 97.

[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.

CKPT

```
►►──CKPT──────────────────────────────────────────────►◄
```

See the discussion of this operand on the OPTION statement, discussed in "OPTION Control Statement" on page 97.

DYNALLOC

```
►►───DYNALLOC─┬─────────────┬─────────────────────────►◄
              └─=─┬─d─────┬─┘
                  ├─(d)───┤
                  ├─(,n)──┤
                  ├─(d,n)─┤
                  ├─OFF───┤
                  └─(OFF)─┘
```

See the discussion of this parameter on the OPTION statement.

EQUALS

```
►►──┬─EQUALS───┬──────────────────────────────────────►◄
    └─NOEQUALS─┘
```

See the discussion of this parameter on the OPTION statement, discussed in "OPTION Control Statement" on page 97.

FILSZ or SIZE

```
►►──┬─FILSZ=────┬─x──┬─┬──────────────────────────────►◄
    │           └─Ex─┘ │
    └─SIZE=─────┬─y──┬─┘
                └─Ey─┘
```

See the discussion of this parameter on the OPTION statement, discussed in "OPTION Control Statement" on page 97.

SKIPREC

```
►►──SKIPREC=z─────────────────────────────────────────►◄
```

See the discussion of this parameter on the OPTION statement, discussed in "OPTION Control Statement" on page 97.

STOPAFT

```
►►──STOPAFT=n──────────────────────────────────────────────►◄
```

See the discussion of this parameter on the OPTION statement, discussed in "OPTION Control Statement" on page 97.

SORT Statement Note

If the records are reformatted by INREC or E15, SORT must refer to the appropriate 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, and contains character data. It is to be sorted in ascending order.

FILSZ
The data set to be sorted contains exactly 29483 records.

SORT Example 2. Five Control Fields, 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),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, and contains character (EBCDIC) data. It is to be sorted in descending order.

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 in ascending order.

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, and contains binary data. It is to be sorted in ascending order.

The fifth control field begins on byte 452, is 8 bytes long, contains normalized floating-point data, which is to be sorted in 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.

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

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

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

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 in descending sequence.

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 in 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 in 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───┴─),FORMAT=f─┤
               │                       │
               └────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

```
                      ┌─,───┐
                      ↓     │
►►────FIELDS=────(────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 can 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:

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

[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.

NONE

eliminates records with duplicate keys. Only one record with each key is kept and no summation is performed.

Default: None; must be specified.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

FORMAT

```
►►──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. Must be specified if not included in the FIELDS parameter.

Applicable Functions: See Appendix C, "Specification/Override of DFSORT Options" on page 353.

SUM Statement Notes

- 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).
- The size of the routine generated by DFSORT to handle the SUM function depends 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.
- Summary fields must not be control fields. They must not overlap control fields, or each other, and must not overlap the RDW.
- Floating-point fields must not be summed.
- When records are summed, you can predict which record is to receive the sum (and be retained) and which record is to be deleted only when EQUALS is in effect and the BLOCKSET technique is used. In this case, the first record (based on the sequence described under EQUALS or NOEQUALS on page 104) is chosen to contain the sum.
- Fields other than summary fields remain unchanged and are taken from the record that receives the sum.
- If overflow occurs, the two records involved are left unsummed. That is, the contents of the records are left undisturbed, neither record is deleted, and the records are still available for summation. Overflow does not prevent further summary.
- DFSORT issues a message and terminates if a SUM statement is specified for a tape work data set sort or Conventional merge.
- 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 4. Using Panels to Create and Submit DFSORT Jobs

Overview of DFSORT Panels

If your site has installed DFSORT Panels under the Interactive System Productivity Facility (ISPF) or the Interactive Storage Management Facility (ISMF), you can perform many DFSORT tasks using interactive panels. You can use DFSORT Panels to sort, copy, and merge files in an online environment that lets you concentrate on performing the task without spending time in coding job control or program control statements.

You can access the DFSORT program in either foreground or background mode, depending on which environment best fits your task at the moment. With foreground execution, you submit your DFSORT job for immediate processing under TSO. With background execution, you can submit your job for batch processing while you perform other tasks, or you can save the completed application in a data set to edit or run later. The advantages of using DFSORT in panel mode are:

- You can create a job without coding JCL and DFSORT command syntax. You provide information about your application on the appropriate panels, and the panels build your DFSORT job automatically.
- You can use the online HELP facility to get information about DFSORT functions and options during your terminal session. HELP is keyed to individual DFSORT panels, so you can also use it as a tutorial. You can get explanations and examples of the panels while you work through them.
- The panels prime your entry fields with last-used values or with defaults, and you can create new jobs with similar control statements by simply paging through the panels. To change an existing value, you type over it. To reset to a default, type over it with blanks.

With DFSORT Panels, occasional users can process data sets effectively with minimal preparation. Experienced users can access DFSORT functions and options more quickly and efficiently.

If you are unfamiliar with either DFSORT or DFSORT Panels, consider reading the tutorial, *Getting Started with DFSORT*. It introduces both DFSORT and the interactive panels in a step-by-step format with illustrations and complete examples. While this chapter describes *all* the functions and options available to you with the panels, *Getting Started with DFSORT* begins with the basics and then progresses to advanced topics.

Before You Begin

You can use DFSORT Panels to specify most DFSORT functions and options directly by entering information in specific fields or by selecting codes on various panels. In some situations, you might want to create a DFSORT job with features that the panels do not directly support. For example, you might want to use:

- Tape input or output data sets
- Additional work data sets

- Specific DFSORT subparameters that do not appear in the panels (such as DEBUG ABEND)
- Nonstandard DFSORT control statements or keywords
- Data types unique to your system
- Data sets not catalogued or allocated on the system from which you access DFSORT Panels
- Other programs that invoke DFSORT.

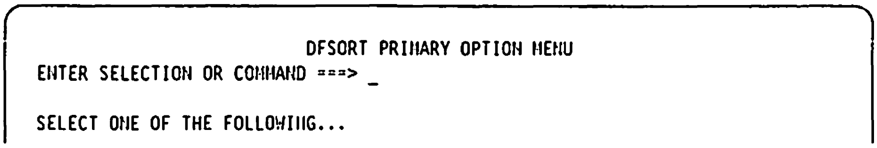
DFSORT Panels provides several solutions to these problems by letting you customize your jobs and the way you create them:

- You can use free form panels to insert control statements into your jobs exactly as you have coded them. If your application requires a DFSORT feature not listed in a panel, you can simply code the statement yourself, and direct DFSORT Panels to use it in your job.
- You can save your completed job in a data set, and edit or add statements later without going through the interactive panels. Using both your text editor and DFSORT Panels to build your jobs lets you choose for yourself the most efficient way to accomplish your tasks.
- You can create DFSORT DD and control statements for use by another program (such as COBOL) that invokes DFSORT. Just save the completed DFSORT job in a data set and copy the required statements to the other job.
- You can suppress some automatic procedures by changing the settings on the panels (for instance, to build a job that uses uncataloged or unallocated data set names, see pages 151 and 152).

Your system programmers can also tailor DFSORT Panels to your needs by adding or substituting their own panels for user-defined control statements with the User-Defined Panels Interface (UDPI). With UDPI, they can insert additional panels that let you create DFSORT jobs with specialized control statements unique to your needs. The *DFSORT Planning and Installation Guide* contains specific information on customizing DFSORT Panels with the User-Defined Panels Interface.

Getting Around in DFSORT Panels

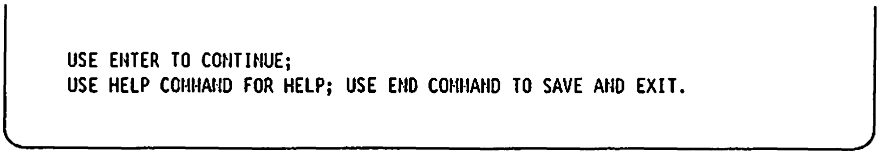
A panel is a formatted screen of information presented on a display terminal. Some panels provide information to you on request (such as HELP panels), while others require you to enter information. What you enter determines which panel you see next, or the function that is performed. Arrows mark the fields where you can enter information or specify choices, as shown in Figure 9.



```
                             DFSORT PRIMARY OPTION MENU
 ENTER SELECTION OR COMMAND ===> _

 SELECT ONE OF THE FOLLOWING...
```

Figure 9. Command Lines on the Panels Are Shown by Arrows

Instructions on the panels tell you what keys and commands to use to get the results you want, as shown in Figure 10.



```
USE ENTER TO CONTINUE;
USE HELP COMMAND FOR HELP; USE END COMMAND TO SAVE AND EXIT.
```

Figure 10. The Panels Explain the Keys and Commands You Use

The following commands control panel flow. Some commands are inactive in panels for which they are not appropriate.

ENTER key On some panels, the **ENTER** key saves your new values and takes you to the next panel. On others, the **ENTER** key verifies that your panel input is correct.

END On some panels, **END** saves your new values and exits to the previous panel. On others, **END** cancels your new values and exits to the previous panel. If you are using HELP, the **END** command takes you back to the panel from which you invoked HELP.

HELP Takes you to a HELP panel. HELP panels give you more information about a particular panel or function. HELP can also give you an explanation of DFSORT Panels messages.

RETURN Terminates the application you are currently performing and returns you to the DFSORT Primary Option Menu.

UP Scrolls the fields on the panel backward by the specified number of lines, or takes you to the previous panel in a series.

DOWN Scrolls the fields forward by the specified number of lines, or takes you to the next panel in a series.

CANCEL Exits the displayed panel without saving any changes. You can use **CANCEL** only on those panels that direct you to "USE END TO SAVE AND EXIT." For all other panels, **END** performs the **CANCEL** function.

REINIT Reinitializes all pages of a panel with default entries or blanks in all the fields. Use this command to start over again with all new entries. Like **CANCEL**, **REINIT** can be used only on panels that direct you to "USE END TO SAVE AND EXIT."

BROWSE Displays the control statement you generated or entered exactly as it would be passed to DFSORT. Enter the command from the SORT or MERGE Control Fields Entry panels or from the SORT or MERGE Free Form Statement Entry panels. Use **BROWSE** to verify that the control statement does what you intended so that you can correct any errors in your panel entries. You can also enter the **B** BROWSE STATEMENT select code on the Control Statements Selection panel to display control statements in the form in which they would be passed to the DFSORT program.

PROFILE Takes you to a menu that lets you view or update your DFSORT user profile. The profile controls the way DFSORT Panels constructs your applications. For information on what options are available and when you might want to change them, see the section "Changing Your DFSORT User Profile" on page 170. You can enter the PROFILE command from any SORT, COPY, or MERGE panel. You see the same panels by entering **0** on the DFSORT Primary Option Menu.

Using Panels to Create DFSORT Jobs

When you select a sort, copy, or merge task from the DFSORT Primary Option Menu, you see the first of a series of panels that let you build and submit or save your DFSORT job. Fill them in one by one with information about what you want to do. The panels for sorting, copying, and merging data sets are similar, and you specify input and output data sets, statements, options, and other information in similar ways. Most functions and options available when you code JCL and DFSORT statements yourself can also be specified using appropriate panels. In fact, if you have used DFSORT before, then it will be helpful to keep in mind that DFSORT Panels actually writes and submits or saves the same JCL and DFSORT control statements with which you are already familiar.

Using the panels makes it easy to modify DFSORT jobs after you have created them. Because the panels retain your final entries between sessions, you can use them again the next time you create a job. If you want to change some items and keep others, you can view and change individual statements one at a time, or choose to turn them off for some jobs and on for others. You can submit the job as soon as you complete it, in either foreground or background mode. You can also use the background execution environment to save the job stream in a data set to edit or submit later.

Because sorting, copying, and merging are separate applications in DFSORT Panels, you can specify different information to appear in the fields when you select different tasks. Only information specified in your DFSORT user profile is common to sort, copy, and merge jobs. Remember that before you can merge data sets, they must be sorted in the same way with respect to the position, length, format, and order that you specify in the MERGE control fields.

The DFSORT panels you work with vary with your task. DFSORT processes the panel entries and saves or submits your job as soon as you supply it with all the information you told it to expect. You do not need to use all of the panels for every job, nor do you always need to look at a panel to use it to construct a job. In fact, you can create and submit a fairly sophisticated DFSORT job after seeing only three panels. This means that the last panel you see before DFSORT processes your entries varies from job to job.

Beginning a DFSORT Session

DFSORT Panels is supported under the Interactive System Productivity Facility (ISPF). The way in which you invoke DFSORT Panels depends on how it was installed at your site. DFSORT Panels might be installed as an option on one or more of the following menus:

- The ISPF Master Application Menu or submenu

- The ISPF/PDF Primary Option Menu or submenu
- The ISMF Primary Option Menu.

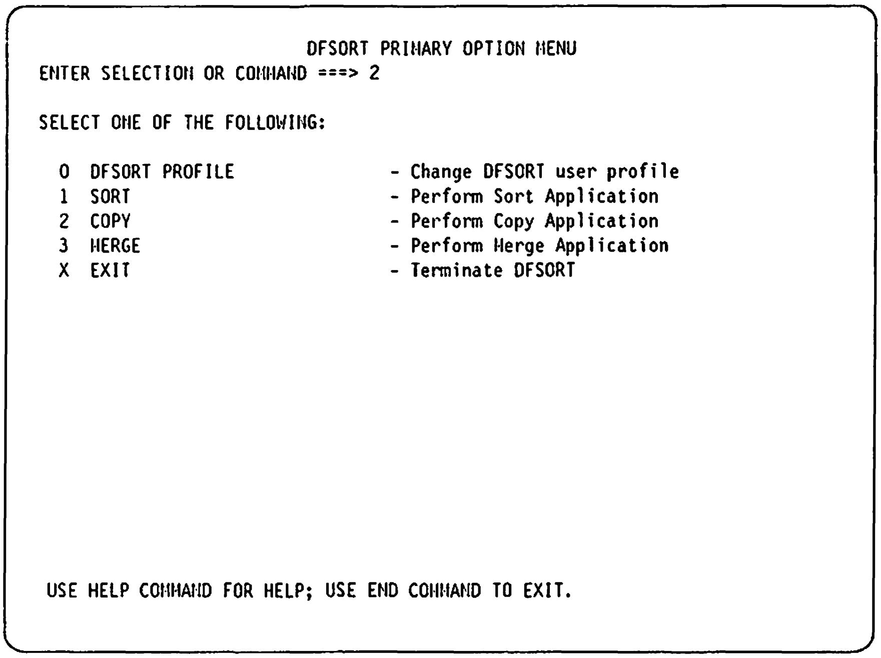
Alternatively, your site might use the ISPSORT command to invoke DFSORT Panels directly from TSO. If you do not know the actual method used to invoke DFSORT Panels, contact your information systems center for help.

If the Interactive Storage Management Facility (ISMF) is available at your site, you can use the SORTREC line operator from the ISMF Data Set Application to do a quick and simple sort on a data set. See "Using SORTREC for a Quick and Simple Sort Under ISMF" on page 174 for details on how to use this feature.

After you specify the appropriate codes to invoke DFSORT Panels, you can begin your session with the DFSORT Primary Option Menu.

Beginning Your Job: The DFSORT Primary Option Menu

To begin sorting, copying, or merging records with DFSORT Panels, specify your choice in the command line of this panel, as shown in Figure 11. You see this menu anytime you begin a DFSORT session, and you use it to terminate DFSORT when you are done. For instance, if you enter **1** (PERFORM SORT APPLICATION), you can begin to build a sort job with the panels that appear on your screen. To copy or merge files, select **2** or **3**, respectively.



```
                           DFSORT PRIMARY OPTION MENU
ENTER SELECTION OR COMMAND ===> 2

SELECT ONE OF THE FOLLOWING:

  0  DFSORT PROFILE                - Change DFSORT user profile
  1  SORT                          - Perform Sort Application
  2  COPY                          - Perform Copy Application
  3  MERGE                         - Perform Merge Application
  X  EXIT                          - Terminate DFSORT












 USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT.
```

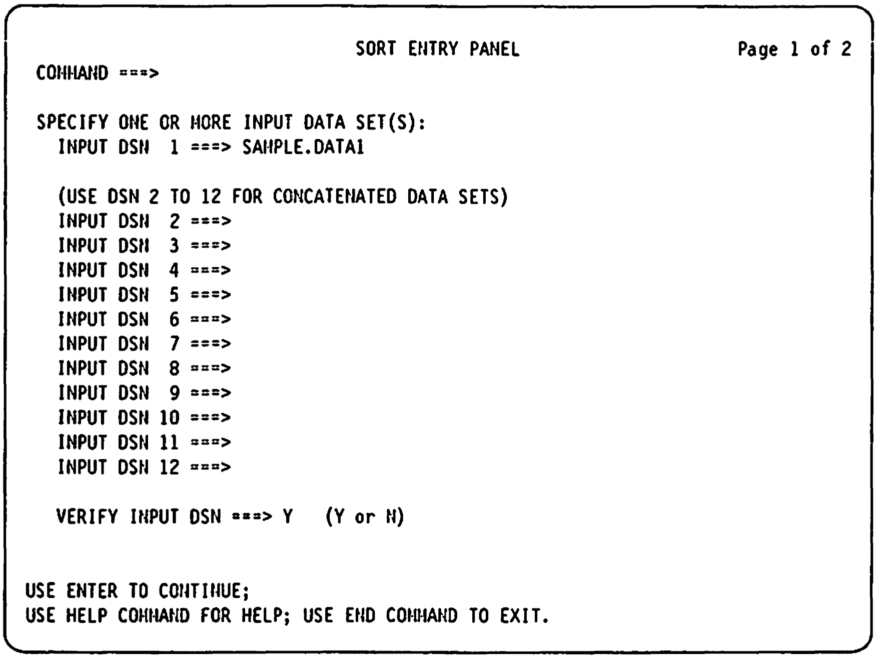
Figure 11. DFSORT Primary Option Menu

You can also change your DFSORT user profile by selecting **0** (CHANGE DFSORT USER PROFILE). The entries in the profile affect the way DFSORT Panels creates DFSORT jobs, and you might want to update it from time to time if you make system changes, or if you want to run a job with unusual requirements. After you have set up your profile, DFSORT Panels automatically uses that information in processing your job. See the section, "Changing Your DFSORT User Profile" on page 170 for more information.

Selecting Input Data Sets: SORT/COPY/MERGE Entry Panels, Page 1

You list the data sets you want to process on the first page of the SORT, COPY, or MERGE Entry Panel. You must specify one data set in the first field, but you can sort or copy up to 12 different data sets at once by specifying other data set names in the remaining fields. DFSORT concatenates the data sets and sorts or copies them into a single output data set. You can merge up to 16 data sets at once by listing the data sets containing the records you want to combine.

Type the name of your input data set into the INPUT DSN 1 field, as shown in Figure 12. Use the other INPUT DSN fields to list the names of additional data sets that you want to process. Remember that the names you use must conform to TSO naming conventions. When sorting and copying, you can leave blank lines between the names. When merging data sets, you must list the names consecutively without any blank lines between them.



```
                            SORT ENTRY PANEL                  Page 1 of 2
COMMAND ===>

SPECIFY ONE OR MORE INPUT DATA SET(S):
  INPUT DSN  1 ===> SAMPLE.DATA1

  (USE DSN 2 TO 12 FOR CONCATENATED DATA SETS)
  INPUT DSN  2 ===>
  INPUT DSN  3 ===>
  INPUT DSN  4 ===>
  INPUT DSN  5 ===>
  INPUT DSN  6 ===>
  INPUT DSN  7 ===>
  INPUT DSN  8 ===>
  INPUT DSN  9 ===>
  INPUT DSN 10 ===>
  INPUT DSN 11 ===>
  INPUT DSN 12 ===>

  VERIFY INPUT DSN ===> Y   (Y or N)


USE ENTER TO CONTINUE;
USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT.
```

Figure 12. SORT Entry Panel (page 1)

When you provide all input records to sort or copy jobs through an E15 exit routine, specify an asterisk (*) in the INPUT DSN 1 field. The panels check remaining fields for errors, but will not use them in your job. If you choose this option, you must also select E15 on the MODS Statement Data Entry panel. (The E15 exit is not applicable to MERGE jobs.) See Chapter 5, "Using Your Own Exit Routines" on page 181 for instructions on how you can prepare user exits.

Use the VERIFY INPUT DSN field to check that your input data set names are valid. If you type a **Y** into the field, you will receive an error message on the panel when you press **ENTER** if any of the input data set names you specified are invalid. Type **N** to suppress validation messages. Incorrect syntax in your input data set names always results in an error message no matter what the VERIFY INPUT DSN setting is.

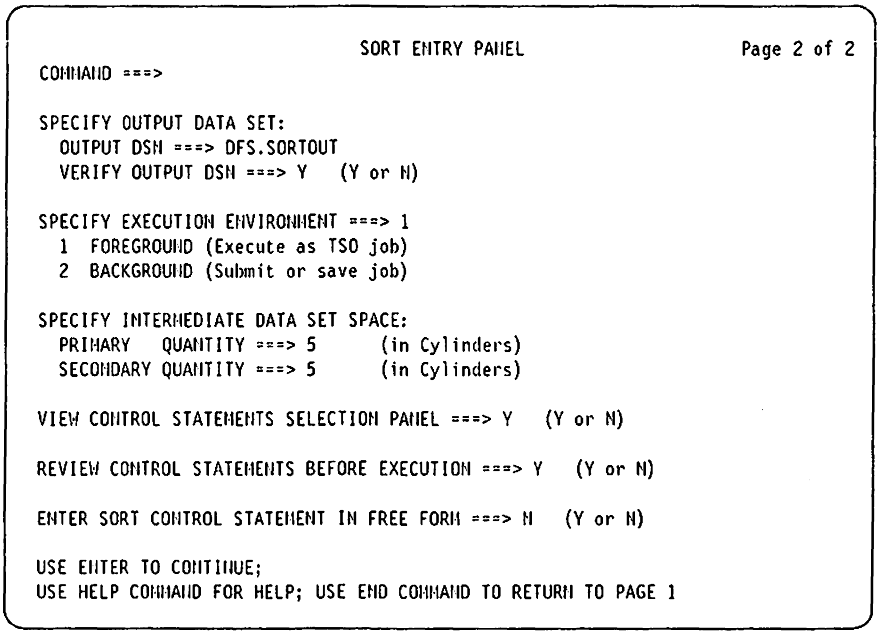
If you know that your input data set names are invalid (as when you construct and save a job prior to allocating the input data sets to which it refers), then the VERIFY INPUT DSN field must be set to **N** in order to continue. You cannot submit the job from the panels this way, but you can save it in a data set and later insert the missing JCL parameters such as volume and unit codes. This way you can construct and save jobs that you can then edit for use on systems in which your data sets are not cataloged.

Setting Your Task Environment: SORT/COPY/MERGE Entry Panels, Page 2

On the second page of the SORT, COPY, or MERGE Entry Panel (see Figure 13), you make decisions about how you want to work with your data sets.

Specifying the Output Data Set

You must first name the output data set in the OUTPUT DSN field. Whether you sort, copy, or merge one or several data sets, the output from a successful job is stored in the data set you name in this field. Remember that you can have as few as 1 or as many as 16 input data sets depending upon the task that you want to perform. DFSORT combines your input into one output data set. This lets you work with all the data sets at once and keep the entire output together as a single data set.



```
                              SORT ENTRY PANEL                Page 2 of 2
COMMAND ===>

SPECIFY OUTPUT DATA SET:
  OUTPUT DSN ===> DFS.SORTOUT
  VERIFY OUTPUT DSN ===> Y    (Y or N)

SPECIFY EXECUTION ENVIRONMENT ===> 1
  1  FOREGROUND (Execute as TSO job)
  2  BACKGROUND (Submit or save job)

SPECIFY INTERMEDIATE DATA SET SPACE:
  PRIMARY   QUANTITY ===> 5       (in Cylinders)
  SECONDARY QUANTITY ===> 5       (in Cylinders)

VIEW CONTROL STATEMENTS SELECTION PANEL ===> Y   (Y or N)

REVIEW CONTROL STATEMENTS BEFORE EXECUTION ===> Y   (Y or N)

ENTER SORT CONTROL STATEMENT IN FREE FORM ===> N   (Y or N)

USE ENTER TO CONTINUE;
USE HELP COMMAND FOR HELP; USE END COMMAND TO RETURN TO PAGE 1
```

Figure 13. SORT Entry Panel (page 2)

If all output is to be handled by an E35 exit routine, specify an asterisk (*) for the output data set name. If you choose this option, you must also select E35 on the MODS Statement Data Entry panel. See Chapter 5, "Using Your Own Exit Routines" on page 181 for instructions on how user exits can be prepared.

As with your input data set names, you can check whether the panels can locate your output data set before you go on. Specify **Y** in the VERIFY OUTPUT

DSN field to allow validation messages, or **N** to suppress them. Syntax errors in your output data set name always result in an error message, even when the VERIFY OUTPUT DSN field is set to **N**.

If you know that your output data set name is invalid (as when you construct and save a job prior to allocating the output data set to which it refers), then the VERIFY OUTPUT DSN field must be set to **N** in order to continue. You cannot submit the job from the panels this way, but you can save it in a data set and later insert the missing JCL parameters such as volume and unit codes. This lets you construct and save jobs which you can then edit for use on systems in which your data sets are not cataloged.

Choosing Between Foreground and Background Execution

You must also decide between running your job in the foreground or background by typing a **1** or a **2** in the SPECIFY EXECUTION ENVIRONMENT field.

Choose the foreground environment to execute your sort, copy, or merge job under TSO as soon as you complete the last panel. Foreground invokes a CLIST command procedure that places all selected DFSORT control statements into a temporary SYSIN data set, allocates the necessary input, output, and intermediate data sets, and then issues a call to the DFSORT program. Remember that TSO jobs occupy your terminal until they complete.

Choose the background environment to save the job stream in a data set that you name, or to submit the job immediately without losing control of your terminal to the program. Background execution generates a complete job stream containing the JCL and DFSORT control statements for processing your job.

Allocating Space for Intermediate Data Sets

When you sort records, you can choose how much space to allocate for the intermediate data set DFSORT uses to process your job. Type your choice in the fields marked PRIMARY QUANTITY and SECONDARY QUANTITY. The defaults are five cylinders each for both primary and secondary allocation. See Appendix A, "Calculating Storage Requirements" on page 311 for details on determining the space you require.

Note: You can allow dynamic allocation of intermediate sort data sets by specifying the DYNALLOC option on the Option Statement Data Entry (SORT) panel. See page 101 for details on using the DYNALLOC parameter.

Selecting and Editing DFSORT Control Statements

You can choose to view or edit DFSORT control statements by specifying **Y** in the VIEW CONTROL STATEMENTS SELECTION PANEL field. If you specify **N** in this field, the control statements and options set up in your previous job will be used, and the Control Statements Selection panel will not be displayed.

Reviewing DFSORT Control Statements

You can still see the DFSORT control statements prior to submitting the job in either foreground or background mode by specifying **Y** in the REVIEW CONTROL STATEMENTS BEFORE EXECUTION field. If you do, the control statements you chose for your job will be displayed for you later on the Review Control Statements panel before you save or submit your job. To change them at that time, you must return to the VIEW CONTROL STATEMENTS SELECTION PANEL field, set the value to **Y**, and edit the appropriate panels through the Control Statements Selection panel.

Entering SORT or MERGE Statements in Free Form

If you are experienced in coding DFSORT statements, you might want to skip the panels that help you to create SORT and MERGE statements and code them yourself. Specify **Y** in the ENTER SORT or MERGE CONTROL STATEMENT IN FREE FORM field. If you do, the next panel that you see is the SORT or MERGE Statement Free Form Entry panel, in which you enter DFSORT program control statements in free form. Use this option when you feel confident that you can save time by coding the control fields statements on your own.

You can also enter program control statements in free form on the General Statement Free Form Entry panel. To work with that panel, specify **Y** in the VIEW CONTROL STATEMENTS SELECTION PANEL field. This panel also lets you enter statements in free form for COPY applications.

Free Form panels also let you insert comment statements into your jobs. See "Specifying the Order with the SORT or MERGE Statement Free Form Entry Panel" on page 155 and "General Free Form Statement" on page 166 for details on using comment statements.

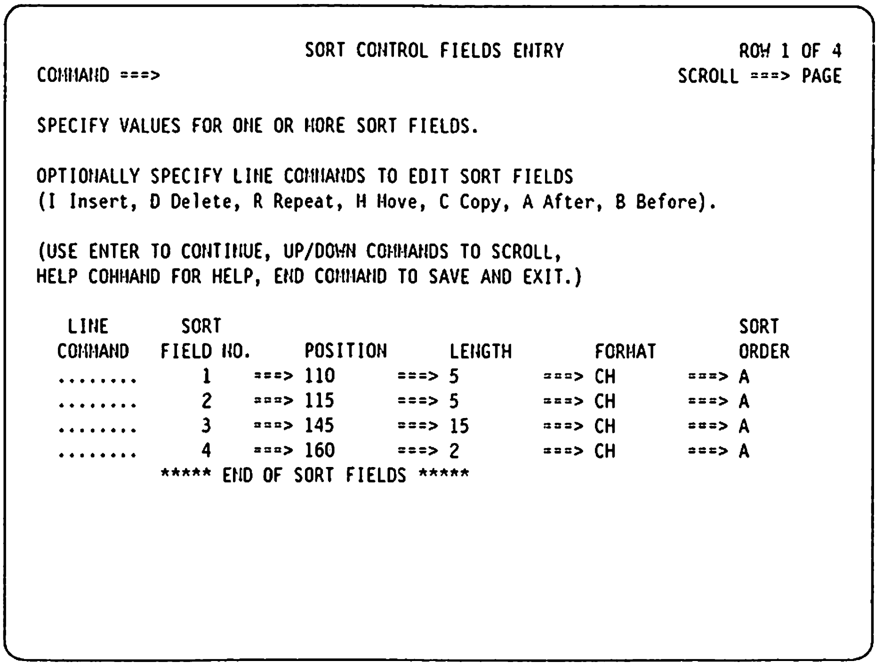
Specifying the Order of Your Sorted or Merged Records

You can specify a sort or merge order for your output records by describing your control fields and collating sequence on the SORT or MERGE Control Fields Entry panel, or you can enter a SORT or MERGE statement on your own on the SORT or MERGE Statement Free Form Entry panel.

Specifying the Order with the SORT or MERGE Control Fields Entry Panel

You see the SORT or MERGE Control Fields Entry panel only if you entered **N** in the ENTER SORT or MERGE STATEMENT IN FREE FORM field on the second page of the SORT, COPY, or MERGE Entry Panel.

The entries you make on the panel (see Figure 14 on page 154) determine which control fields and collating sequences DFSORT uses to organize your records. There are initially four lines of entry fields on the panel. You must specify at least one control field to continue constructing a sort or merge job stream, and you can work with up to 100 by adding fields with line commands. Valid line commands are shown in Table 9 on page 155. See Figure 14 on page 154 for an illustration of the entries you might make using four control fields.



```
                          SORT CONTROL FIELDS ENTRY                  ROW 1 OF 4
COMMAND ===>                                                  SCROLL ===> PAGE

SPECIFY VALUES FOR ONE OR MORE SORT FIELDS.

OPTIONALLY SPECIFY LINE COMMANDS TO EDIT SORT FIELDS
(I Insert, D Delete, R Repeat, M Move, C Copy, A After, B Before).

(USE ENTER TO CONTINUE, UP/DOWN COMMANDS TO SCROLL,
HELP COMMAND FOR HELP, END COMMAND TO SAVE AND EXIT.)

  LINE     SORT                                                  SORT
 COMMAND  FIELD NO.     POSITION      LENGTH        FORMAT       ORDER
 ........     1    ===> 110      ===> 5        ===> CH      ===> A
 ........     2    ===> 115      ===> 5        ===> CH      ===> A
 ........     3    ===> 145      ===> 15       ===> CH      ===> A
 ........     4    ===> 160      ===> 2        ===> CH      ===> A
          ***** END OF SORT FIELDS *****
```

Figure 14. SORT Control Fields Entry Panel

The sequence in which you list the control fields becomes the order of priority DFSORT uses to arrange your records. If two or more records have identical values for the first control field, they are sequenced according to the values in the second, and so on. Records with identical values for all the control fields retain their original input order or are arranged randomly, depending upon the defaults set during installation. You can direct DFSORT to retain the original input order with the EQUALS option on the OPTION Statement Data Entry panel.

For each control field that you want to use, you must specify:

- The position
- The length
- The data format
- Whether you want ascending or descending order.

DFSORT identifies your control fields by their POSITION and LENGTH. Type the position of the first byte of your first control field (the first position in the record is byte 1) into the POSITION field, and its length in bytes into the LENGTH field. Remember that the Record Descriptor Word (RDW) occupies bytes 1-4 in variable-length records. You can combine adjacent control fields into one, if they are in the same format, by specifying the position of the first control field and the combined length of both control fields. DFSORT processes the combined control fields as a single long one.

The FORMAT field indicates to DFSORT which format to use to read the data in that particular field. The default is **CH** for character data. See Table 8 on page 135 for a complete listing of acceptable data formats and their symbols.

Specify your choice of ascending or descending order in the SORT ORDER or MERGE ORDER field with **A** or **D**. You can use ascending and descending order simultaneously for different control fields within the same job. You can also specify **E** to indicate that your control fields will be modified by an E61 exit routine before DFSORT processing. See "E61 Exit: Modifying Control Fields" on page 200 for information on the E61 exit.

After you finish with your first control field, move on to any others you want to use. If you are working with multiple control fields, you can save time by using line commands to add, delete or rearrange control fields. You can insert blank control fields, delete fields that you do not want to use, or repeat fields once or many times. You can also copy or move control fields from place to place within the list. Line commands let you quickly modify lists of control fields to perform different tasks without having to retype the entire list. Valid line commands are shown in Table 9.

Table 9. Line Commands You Can Use in DFSORT Panels

Use:	To:
I	Insert a single unused field after a line.
Ix	Insert **x** unused fields after a line.
I-	Insert a single unused field before a line.
I-x	Insert **x** unused fields before a line.
D	Delete a single field.
Dx	Delete **x** fields, including this one.
DD	Mark the first and last fields of a block of fields to be deleted.
R	Repeat a single field.
Rx	Repeat a single field **x** times.
RR	Mark the first and last fields of a block of fields to be repeated.
RRx	Mark the first and last fields of a block of fields to be repeated **x** times.
C	Mark a single field to be copied.
Cx	Mark the first of **x** fields to be copied.
CC	Mark the first and last fields of a block of fields to be copied.
M	Mark a single field to be moved.
Mx	Mark the first of **x** fields to be moved.
MM	Mark the first and last fields of a block of fields to be moved.
A	Designates the field **after which** fields are to be moved or copied.
B	Designates the field **before which** fields are to be moved or copied.

Specifying the Order with the SORT or MERGE Statement Free Form Entry Panel

You see the SORT or MERGE Statement Free Form Entry panel only if you specified **Y** in the ENTER SORT or MERGE CONTROL STATEMENT IN FREE FORM field on the second page of the SORT or MERGE Entry Panel.

The SORT or MERGE Statement Free Form Entry panel lets you write your own control statements to use in the job. You enter your control statements with the same coding conventions you use when writing a DFSORT application without the panels. Use the Free Form panel to enter DFSORT control statements when you do not want to use the Control Fields Entry panels to help you build them.

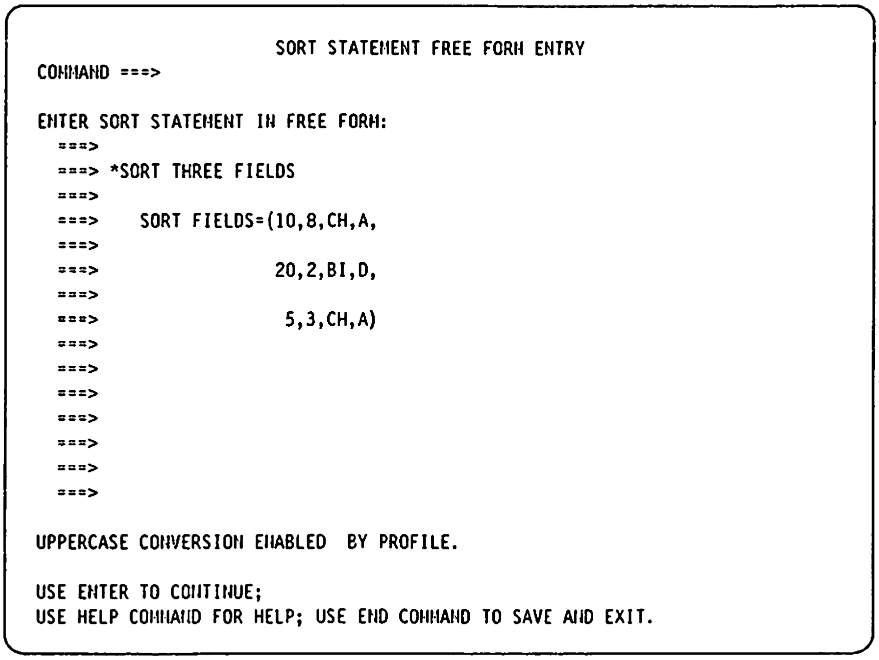
The panel contains 15 lines into which you can type DFSORT control statements. Consider the following as you organize statements and insert comments:

- To continue a statement on the next line, break the line at a blank space following a comma or a semicolon.
- Blank lines are ignored, but any entries you make on the Free Form panel are passed to DFSORT exactly as you type them. Free form entries are *not* verified automatically, so you must use correct syntax.
- DFSORT ignores any blank lines you leave between control statements, but any lines that begin with an asterisk (*) in column 1 are inserted into your job as comment statements.

See Figure 15 for an example of how you could use the SORT Free Form Entry panel.

Notes:

- You can use the Free Form Entry panels to insert non-DFSORT control statements to be processed through DFSORT's Extended Function Support interface. See Appendix B, "Using Extended Function Support (EFS)" on page 315 for details on using EFS.
- The General Statement Free Form Entry panel lets you insert up to 30 additional lines of free form control statements into your job. See "General Free Form Statement" on page 166 for details.



```
                        SORT STATEMENT FREE FORM ENTRY
COMMAND ===>

ENTER SORT STATEMENT IN FREE FORM:
  ===>
  ===> *SORT THREE FIELDS
  ===>
  ===>    SORT FIELDS=(10,8,CH,A,
  ===>
  ===>                20,2,BI,D,
  ===>
  ===>                 5,3,CH,A)
  ===>
  ===>
  ===>
  ===>
  ===>
  ===>
  ===>

UPPERCASE CONVERSION ENABLED  BY PROFILE.

USE ENTER TO CONTINUE;
USE HELP COMMAND FOR HELP; USE END COMMAND TO SAVE AND EXIT.
```

Figure 15. SORT Statement Free Form Entry Panel

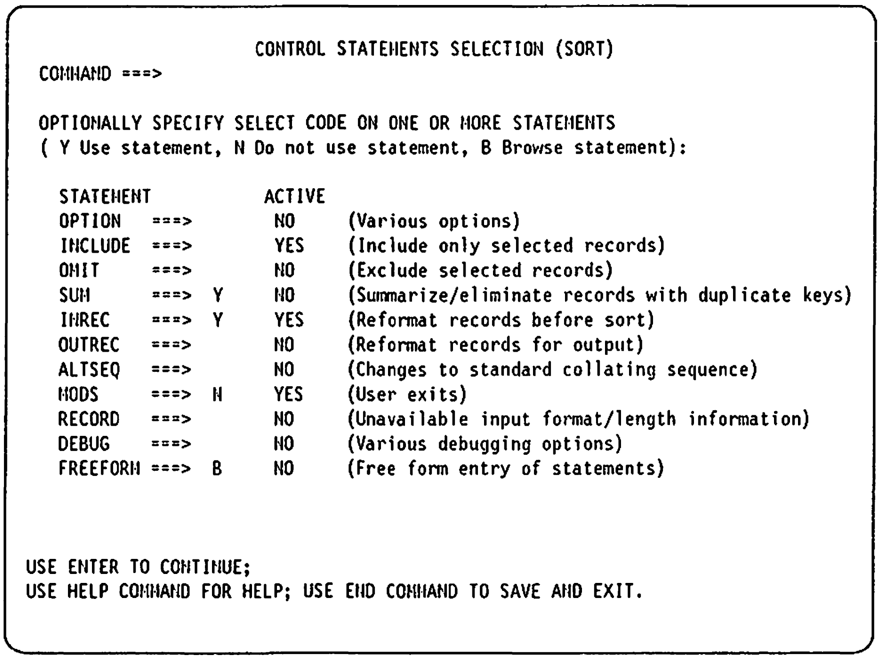
Using DFSORT Control Statements: Control Statements Selection Panel

You see the Control Statements Selection panel only if you entered **Y** on the VIEW CONTROL STATEMENTS SELECTION PANEL field on the second page of the SORT, COPY, or MERGE Entry Panel. Use the Control Statements Selection panel to select additional DFSORT control statements such as OPTION, INCLUDE, INREC, and SUM.

Activating Control Statements

Available control statements are listed on the left side of the panel, with their descriptions to the right. Only those control statements with "YES" in the column headed "ACTIVE" are activated for your job. Statements with "NO" in the column are not inserted into your job, but can be activated by typing **Y** into the blank space to the right of the arrow. You deactivate control statements the same way by typing **N**. By turning control statements on or off, you create different applications using different combinations of statements. Deactivated statements are not erased between sessions, and can be inserted into any application by reactivating them.

Figure 16 illustrates the Control Statements Selection (SORT) panel filled out to use the INCLUDE control statement without modification, to activate the SUM and INREC control statements for modification and use, to deactivate the MODS control statement, and to browse the control statements that would be created by the General Statement Free Form Entry panel were you to activate it.



```
                      CONTROL STATEMENTS SELECTION (SORT)
COMMAND ===>

OPTIONALLY SPECIFY SELECT CODE ON ONE OR MORE STATEMENTS
( Y Use statement, N Do not use statement, B Browse statement):

  STATEMENT          ACTIVE
  OPTION   ===>       NO     (Various options)
  INCLUDE  ===>       YES    (Include only selected records)
  OMIT     ===>       NO     (Exclude selected records)
  SUM      ===>  Y    NO     (Summarize/eliminate records with duplicate keys)
  INREC    ===>  Y    YES    (Reformat records before sort)
  OUTREC   ===>       NO     (Reformat records for output)
  ALTSEQ   ===>       NO     (Changes to standard collating sequence)
  MODS     ===>  N    YES    (User exits)
  RECORD   ===>       NO     (Unavailable input format/length information)
  DEBUG    ===>       NO     (Various debugging options)
  FREEFORM ===>  B    NO     (Free form entry of statements)



USE ENTER TO CONTINUE;
USE HELP COMMAND FOR HELP; USE END COMMAND TO SAVE AND EXIT.
```

Figure 16. Control Statements Selection Panel (SORT)

Each control statement has one or more panels you use to direct the function of the statement in your application. When you press **ENTER** after specifying **Y** select codes on the Control Statements Selection panel, you see the panels for each statement you specified, one after the other. You can modify them or retain them unchanged. After you inspect all specified panels you return to the Control Statements Selection panel and can continue constructing your job.

Note: Control statement panels must have entries in the fields to remain activated. For example, if you delete all the entries on a panel and then exit with the **END** command (which saves the blank panel), DFSORT Panels deactivates the statement automatically.

You can also type **B** (BROWSE STATEMENT) instead of **Y** or **N** to display any statement as DFSORT Panels would create it based on the current settings of its control statement panel. Use this option for a quick look at the statement in final format to decide if you want to use or modify it.

Editing DFSORT Control Statements

With DFSORT Panels, you can use control statements to perform the same operations on your data sets that you perform when you create and submit DFSORT jobs in nonpanel mode. Each control statement has one or more data entry panels where you specify necessary parameters, relational conditions, or other values. You decide how you want the control statement to function, and the panels create the statement for you in correct format.

Press **ENTER** to verify your entries. Use **END** to save changes and go to the next control statement panel that you specified (or return to the Control Statements Selection panel if no further statements were selected).

If you decide not to use a particular control statement in your job after you are within the data entry panel, you can exit the panel with either the **END** or the **CANCEL** command. Exiting with **END** saves the changes you have made to the panel. Exiting with **CANCEL** erases any changes you have made. The **END** or the **CANCEL** commands take you either to the next data entry panel (if you selected any other control statements on the Control Statement Selection panel) or back to the Control Statements Selection panel. When you return to the Control Statements Selection panel, enter the **N** select code in the field next to any control statements that you decided not to use, and press **ENTER** to deactivate them.

You can use **REINIT** from the command line of a data entry panel to reinitialize all the pages with default entries or blanks in all the fields. Use **REINIT** to start over with all new entries.

The control statements in DFSORT Panels work the same way that they do in JCL-invoked DFSORT applications. See Chapter 3, "Using DFSORT Program Control Statements" on page 53 for information on how control statements operate and which parameters are available to you in each of them. The following discussions explain only those features unique to using control statements in DFSORT Panels.

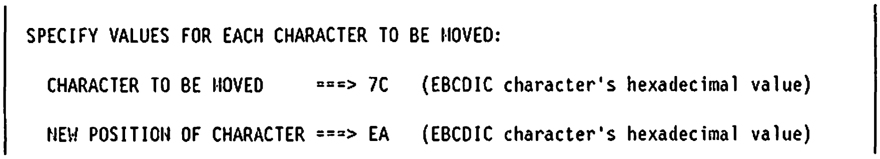
ALTSEQ Statement: The ALTSEQ statement lets you change the EBCDIC collating sequence for any control fields you specify with format AQ. If CHALT is in effect, your alternate collating sequence is also used for any control fields you specify with format CH. Your ALTSEQ statement overrides the default alternate collating sequence set during DFSORT installation at your site.

The two pages of the panel give you two different ways to define your alternate collating sequences. You can use either technique (or both) to indicate which EBCDIC characters to collate differently. The EBCDIC sequence is still used for any characters you do not change.

Use the ALTSEQ Data Entry panel to set up your alternate sequence by entering one or more pairs of hexadecimal values which represent:

- The *original* collating position for the EBCDIC character
- The *new* collating position for the EBCDIC character.

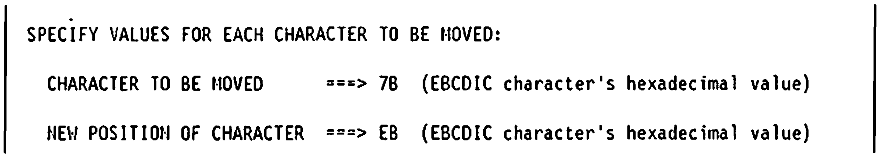
For example, if you want @ (hexadecimal 7C) to collate after Z (hexadecimal E9), enter the following values and press **ENTER**:



```
SPECIFY VALUES FOR EACH CHARACTER TO BE MOVED:

  CHARACTER TO BE MOVED     ===> 7C   (EBCDIC character's hexadecimal value)

  NEW POSITION OF CHARACTER ===> EA   (EBCDIC character's hexadecimal value)
```

The values are accepted and blanked out so that you can enter more values. @ now collates after Z.

If you next want # (hexadecimal 7B) to collate after @ (now at hexadecimal EA), type the following values and press **ENTER**:



```
SPECIFY VALUES FOR EACH CHARACTER TO BE MOVED:

  CHARACTER TO BE MOVED      ===> 7B  (EBCDIC character's hexadecimal value)

  NEW POSITION OF CHARACTER  ===> EB  (EBCDIC character's hexadecimal value)
```

The values are again accepted and blanked out so that you can enter more. @ now collates after Z, and # collates after @.

The ALTSEQ Code Table on the next page of the panel can show you all of the changes you made for your alternate sequence. You can also change the collating sequence directly from the table. The headings 0 through F for the rows and columns of the table correspond to the first and second hexadecimal digits, respectively, of the *old* EBCDIC characters. To change a collating position for any EBCDIC character, locate the blank corresponding to its first and second hexadecimal digits, and enter the new collating position in the blank.

The ALTSEQ Code Table displays the current alternate collating settings whenever you enter it.

For more information about the ALTSEQ statement, see page 60.

DEBUG Statement: Use the Debug Statement Data Entry panel to select various debugging options for your application. Normally, you do not need to use these options, but they might be useful when you try to isolate or bypass a problem in the DFSORT program. Use these options only if necessary, as their use can degrade performance.

The Debug Statement Data Entry panels for SORT, COPY, and MERGE are similar, but only those options which are relevant to each application appear on the corresponding panels. To select a DEBUG option for inclusion in your job, enter an **S** in the corresponding field.

See page 62 for detailed information about what these options do and when you might want to use them:

- BSAM (page 63)
- BUFFERS (page 63)
- EQUCOUNT (page 66)
- NOASSIST (page 66).

INCLUDE Statement: Use the INCLUDE Statement Data Entry panel to limit the records in your output data set to those for which a logical expression you specify is true. You can specify up to two relational conditions combined with a logical AND or OR operator to compare fields to fields, or fields to constants. (You can also limit the records in your input data sets by omitting, instead of including, those records for which a logical expression is true. See the following section describing the OMIT statement.)

For a field-to-field comparison, describe the POSITION, LENGTH and FORMAT of the first and second control fields in the entry fields marked with a **1**, and a **2**, respectively. Remember that the Record Descriptor Word (RDW) occupies bytes 1-4 in variable-length records. Valid format codes are listed on the panel, and are explained on page 70.

Enter a comparison operator in the COMPARISON field. Valid comparison operators are:

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.

If you want to compare the first control field with a constant, instead of another control field, enter the constant in the CONSTANT field. The correct formats for constant strings are identical to those you use in nonpanel mode:

Decimal number format: [±]*n*

Character string format: **C**'*xx...x*'

Hexadecimal string format: **X**'*xx...x*'

See page 69 for more information about constants and comparisons.

Note that if you specify a character constant string, you must ensure that you enter uppercase and lowercase characters appropriately between the quote marks. DFSORT Panels passes text within the character constant string to the program exactly as you enter it on the panel; it cannot determine whether the case is correct.

On the second page of the panel, a second relational condition can be described in the same fashion as the first. Direct DFSORT to combine the two conditions into a single logical expression by specifying the **AND** or **OR** operator on the second page of the panel. If your expression requires more than two relational conditions, use the General Statement Free Form Entry panel to specify your INCLUDE statement.

Padding is a technique that inserts fillers in the data, usually zeros or blanks. *Truncation* is the deletion or omission of a leading or of a trailing portion of a string. See "Padding and Truncation" on page 75 for specific information.

For more information about the INCLUDE statement, see page 68.

OMIT Statement: The OMIT statement, like the INCLUDE statement, is used to screen the records to be included in your output data set. With OMIT, you specify a logical expression that, if true, excludes a record from your output data set. The two pages of the OMIT Statement Data Entry panel work in the same way as the INCLUDE Statement Data Entry panel; see the discussion of that panel for details.

Use either the INCLUDE or the OMIT statement to filter your records, depending upon whichever is more convenient for you. You cannot specify both within the same job.

See page 95 for more information about using OMIT.

INREC and OUTREC Statements: The panels you use to create INREC and OUTREC control statements are similar, and you specify information in the fields in similar ways.

Use the INREC Input/Separation Fields Entry panel to reformat records prior to further processing during DFSORT execution. You can delete or reorder fields, and insert blanks or zeros. DFSORT processes the INREC statement *before* SORT, COPY, MERGE, SUM, OUTREC, and the E35 user exit, but *after* INCLUDE, OMIT, and the E15 user exit. When using INREC, remember that SORT, COPY, MERGE, SUM, OUTREC and the E35 user exit always refer to the positions and lengths of *reformatted* records. INCLUDE, OMIT, and the E15 user exit refer to the *original* records.

Use the OUTREC Input/Separation Fields Entry panel to reformat records *after* all other processing but before the records are written. You construct the OUTREC statement from the panel in the same way that you construct the INREC statement. As with INREC, you can delete or reorder fields, and insert blanks or zeros. You can use INREC and OUTREC statements in the same job to reformat records both before and after other processing.

Both the INREC and OUTREC Input/Separation Fields Entry panels initially contain four unused fields, but you can work with up to 100 by adding fields with line commands (see Table 9 on page 155). Type the position of the first byte of your first input field (the first position in the record is byte 1) into the POSITION field, and its length in bytes into the LENGTH field. For variable-length records, the first input field must consist of or include the Record Descriptor Word (RDW) in bytes 1-4.

If you want to alter the alignment of the input fields within your reformatted records, you can do so with the ALIGNMENT field on the panel. Specify a value in the field indicating your choice of displacement boundaries upon which that input field should be placed. (Because displacements start at byte 0 and positions start at byte 1, the displacement is always one byte less than the position.) Specifying an alignment value causes DFSORT to skip bytes before an input field, if necessary, so as to align the input field on the boundary you indicate. Binary zeros are placed in the bytes that are skipped.

The alignment values you can use are:

H Halfword aligned. The displacement is a multiple of 2 (0, 2, 4, etc.).

F Fullword aligned. The displacement is a multiple of 4 (0, 4, 8, etc.).

D Doubleword aligned. The displacement is a multiple of 8 (0, 8, 16, etc.).

If you specify an alignment value for an input field, you must also specify the position and length.

You can use the entries in the SEPARATION field to insert blanks or binary zeros into your records. These can be placed before, between, or after input fields or other separation fields. Separation fields permit you to improve the readability of data lists.

Each separation field consists of the length of the separation field followed by the separation type. Valid types are **X** for blank, and **Z** for binary zero. Length can vary from 1 to 256 bytes. For example, **8X** means to insert eight blanks, while **240Z** means to insert 240 bytes of binary zeros.

For variable-length records:

- Separation fields cannot come before the first input field (the Record Descriptor Word) in bytes 1-4.
- Separation fields cannot come after the last input field if the length is omitted (that is, if the last field is for the variable part of the record).

A SEPARATION field can be specified without also specifying values in the POSITION, LENGTH, or ALIGNMENT fields.

You can save time by using line commands to add, delete or rearrange entries in the INREC or OUTREC panels. The line commands in the INREC and OUTREC panels work in the same way as those in the SORT and MERGE Control Fields Entry panels (see Table 9 on page 155).

Because processing is quicker with shorter records, you can save time if you use INREC to delete fields and OUTREC to insert blanks or zeros. Rearranging the order of fields does not change total record length, so neither statement gives you slower processing after reordering fields. Both statements have these functions so that you can choose whichever is more convenient for you.

If processing time is not an issue, you can simplify working with your control fields by editing your records with OUTREC. Using OUTREC lets you specify positions and lengths in other control fields as they appear in your input data sets, while INREC requires that you specify positions and lengths in some of the other control statements as they appear in the reformatted records. However, INREC can improve performance when you use it to make your records significantly shorter.

For more detailed information about the INREC and OUTREC statements, see pages 78 and 121.

MODS Statement: Use the MODS Statement Data Entry panel to direct DFSORT calls to user exit routines which you supply. The MODS panel lets you specify:

- Which exit routines to call
- Where the exit routines are located
- Approximately how much main storage each routine uses
- Whether certain exit routines are written in COBOL.

The panel lists the exits you can use. Not all exits apply to all applications, so the list differs depending upon whether you want to sort, copy, or merge data sets. See Chapter 5, "Using Your Own Exit Routines" on page 181 for complete details about each of these exits, as well as restrictions that apply to the MODS statement.

You describe your exit routine in the MEMBER NAME, MAIN STORAGE, and COBOL ROUTINE fields. Type the 1- to 8-character member name of that routine in the appropriate MEMBER NAME field. For example, if you link-edit an E31 routine into 'SYSTEM1.EXITS(SPOPEN)' and an E35 routine into 'SYSTEM2.EXITS(ADDZIP)', the E31 and E35 member names would be SPOPEN and ADDZIP, respectively.

You can use any valid operating system name for a routine. You can also keep several alternative routines with different names in the same library (for example, E35A, and E35B), and select the one that you want to call for a given application by its name.

If you enter a MEMBER NAME, you must also enter the approximate amount of bytes of main storage that the routine requires in the corresponding MAIN STORAGE field. The value that you specify must be the approximate sum of:

- The storage required for the load module of the routine
- The storage that the routine obtains using system services such as GETMAIN and OPEN
- The storage required to load the COBOL library routine (if the routine is written in COBOL).

If you specify an E15 or E35 exit routine, indicate whether or not your routine is written in COBOL by typing **Y** or **N**, respectively, in the COBOL ROUTINE field.

Enter the names of the library (partitioned) data sets containing your exit routines in the LIBRARY DSN fields. You can use from one to eight data set names for sorting, and from one to four data set names for merging and copying, but you must have a data set name in the LIBRARY DSN 1 field to continue. If the same data set contains two or more members, you need to specify the library name only once. Note that the order in which you specify data set names in the LIBRARY DSN fields determines the search order used to locate them. DFSORT ignores any blank LIBRARY DSN fields.

Use the VERIFY LIBRARY DSN field to check that your library data set names are valid. If you type a **Y** into the field, you will receive an error message on the panel when you press **ENTER** if any of the library data set names you specified are invalid. Type **N** to suppress validation messages. Incorrect syntax in your library data set names always results in an error message no matter what the VERIFY LIBRARY DSN setting is.

If you know that your library data set names are invalid (as when you construct and save a job prior to allocating the library data sets to which it refers), the VERIFY LIBRARY DSN field must be set to **N** to continue. You cannot submit the job from the panels this way, but you can save it in a data set and later insert the missing JCL parameters such as volume and unit codes. This way you can construct and save jobs that you can then edit for use on systems in which your data sets are not cataloged.

For more information about the MODS statement, see page 91.

OPTION Statement: Use the OPTION Statement Data Entry panel to select various DFSORT program options for your application. Only the options relevant to a particular sort, copy, or merge application appear on the corresponding OPTION panel, so some of the following options might not apply to a given task. In general, these options let you override DFSORT defaults set during installation by your system programmers, and to request optional information for your application. What you type into the entry fields depends upon the particular option. You might specify **Y** or **N**, a number, or simply **S** to activate a selection. The necessary values are listed opposite the option entry field.

See the pages listed below for detailed information about what these options do, and when you might want to use them:

ARESALL	page 98
CHALT	page 99
CHECK	page 99
COBEXIT	page 101
DYNALLOC	page 101
EQUALS	page 104
FILSZ	page 105
MAINSIZE	page 108
NOBLKSET	page 110
NOOUTREL	page 110
NOOUTSEC	page 111
NOSTIMER	page 111
NOWRKREL	page 111
NOWRKSEC	page 112
RESALL	page 112
SIZE	page 105
SKIPREC	page 113
STOPAFT	page 115
VERIFY	page 116
VLSHRT	page 116.

RECORD Statement: To process your records, DFSORT must work with their format and length. Under certain conditions, this information is unavailable to the program from the system control blocks, and must be supplied from the

Record Statement Data Entry panel. The panels for SORT, COPY, and MERGE applications are basically similar and are not discussed separately here. Because MERGE applications do not use the E15 user exit routine, references to that exit do not apply to the Record Statement Data Entry (MERGE) panel.

You must specify *record type* when:

- You work with VSAM input and VSAM output data sets.
- You use an E15 exit to provide all of your input records.
- The system has no record type information about your input data sets.

Indicate your record type in the OPTIONALLY SPECIFY RECORD TYPE field:

F Fixed-length records

V Variable-length records

D ISCII/ASCII variable-length records.

You must specify *maximum input record length* when:

- You use an E15 routine to provide all your input records.
- The system has no record length information about your input data sets.

Indicate the length in the MAXIMUM INPUT RECORD LENGTH (L1) field. For variable-length records, the L1 value you enter must be at least as large as the length of your longest input record.

If you use an E15 exit routine that changes your input record length (for example, to eliminate fields within your input records), you must indicate the *maximum record length after invoking the E15.* Enter the number in the MAXIMUM LENGTH AFTER E15 (L2) field. For variable-length records, the L2 value that you enter must be at least as large as the length of the longest record you pass to DFSORT from your exit routine.

You must specify *output record length* when it differs from your input record length, *and* the system has no record length information about your output data set. Enter your output record length in the MAXIMUM OUTPUT RECORD LENGTH (L3) field. For variable-length records, the L3 value that you enter must be at least as large as the length of your longest output record.

For more information about the RECORD statement, see page 127.

SUM Statement: Use the SUM Summary Fields Entry panel to handle processing of input records with equal control fields; that is, those records with duplicate keys. With SUM, you choose to keep only one record from a group of input records with duplicate keys, and can also choose whether or not to sum one or more numeric fields from those records. (Select EQUALS on the OPTION Statement Data Entry panel to direct DFSORT to retain the *first* record with each unique control field. See page 104 for details on EQUALS).

To retain only one input record from the set and sum numeric fields, select **1** (SUMMARIZE RECORDS WITH DUPLICATE KEYS) in the SPECIFY ONE OF THE FOLLOWING field. If you choose this option, you must also indicate which of the numeric fields (the *summary fields*) you want to sum.

To retain only one input record from the set without summing any numeric fields, specify 2 (ELIMINATE RECORDS WITH DUPLICATE KEYS) in the same entry field. If you choose this option, you do not need to enter further information on the panel. Any existing summary fields on the panel are saved but otherwise ignored.

The other entry fields on the panel let you describe which numeric summary fields to sum during DFSORT processing. The panel initially displays four unused summary fields but you can work with up to 100 by adding fields with line commands. See Table 9 on page 155 for an explanation of the line commands you can use. Type the position of the first byte of each summary field (the first position in the record is byte 1) into the POSITION field, and the length in bytes into the LENGTH field. Indicate the data format in the FORMAT field. Remember that the Record Descriptor Word (RDW) occupies bytes 1-4 in variable-length records. Valid format codes are explained on page 141. Summary fields can be located anywhere within the first 4092 bytes of your input records, as long as they do not overlap control fields, other summary fields, or the Record Descriptor Word (RDW).

For more information about the SUM statement, see page 141.

General Free Form Statement: Use the General Statement Free Form Entry panel to insert complete DFSORT program control statements into your job. The General Free Form panel works like the SORT and MERGE Statement Free Form panels. You can enter up to 30 lines containing DFSORT control statements. Specify a comment statement by typing an asterisk (*) in column 1. Blank lines are ignored, but any entries you make on the Free Form panel are passed to DFSORT exactly as you type them. Free form entries are *not* verified automatically, so you must use correct syntax.

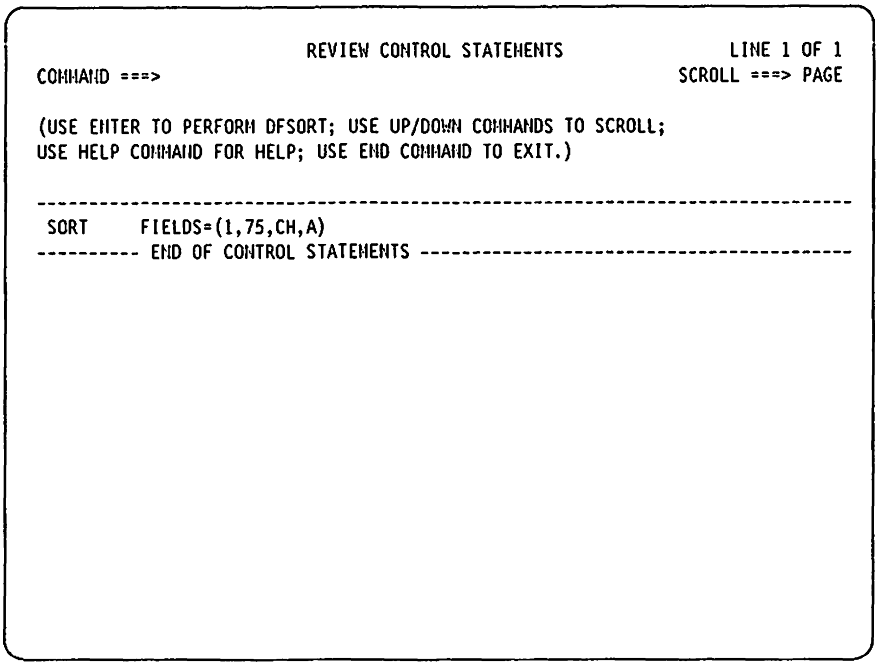
If you enter a control statement (for example, INCLUDE) in the General Statement Free Form Entry panel and also use the corresponding data entry panel for that statement, both statements are passed to DFSORT. However, because DFSORT Panels places the free form statement *before* the data entry statement, DFSORT ignores the data entry statement as a duplicate.

Note: You can use the General Statement Free Form Entry panel to insert non-DFSORT control statements to be processed through DFSORT's Extended Function Support interface. See Appendix B, "Using Extended Function Support (EFS)" on page 315 for details on using EFS.

Inspecting Your Work: Review Control Statements Panel

After you enter all required information describing your DFSORT sort, copy, or merge application, you can inspect the completed program control statements before saving or submitting the job. You see only the Review Control Statements panel if you specified **Y** in the REVIEW CONTROL STATEMENTS BEFORE EXECUTION field on the second page of the SORT, COPY, or MERGE Entry Panel. The panel displays all active control statements as they would be passed to DFSORT.

Use the Review Control Statements panel (see Figure 17 on page 167) to check whether you activated the correct control statements, and that they are appropriate for your application. If you used free form entry to write some or all of the control statements for your job, you can check them here for errors and proper syntax. Any program control statements created using DFSORT control statements entry panels are already in correct format and syntax. Free form entries are *not* verified automatically.



```
                       REVIEW CONTROL STATEMENTS               LINE 1 OF 1
COMMAND ===>                                                SCROLL ===> PAGE

(USE ENTER TO PERFORM DFSORT; USE UP/DOWN COMMANDS TO SCROLL;
USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT.)

-------------------------------------------------------------------------------
 SORT     FIELDS=(1,75,CH,A)
---------- END OF CONTROL STATEMENTS -----------------------------------------
```

Figure 17. Review Control Statements Panel

If the statements are incorrect, or are not appropriate to your task, use **END** one or more times to return to earlier panels that you can use to alter them. You activate or deactivate control statements by using the Control Statements Selection panel; you modify control statements by using the control statement data entry panels.

If you are satisfied with the way your statements appear, press **ENTER** to continue. If you specified foreground execution of the job, DFSORT Panels processes your job under TSO. If you specified background execution, DFSORT Panels takes you on to the panels from which you can submit the job, or save the application in a data set for future use.

Using Panels to Save or Submit Your DFSORT Job

Using Foreground Execution

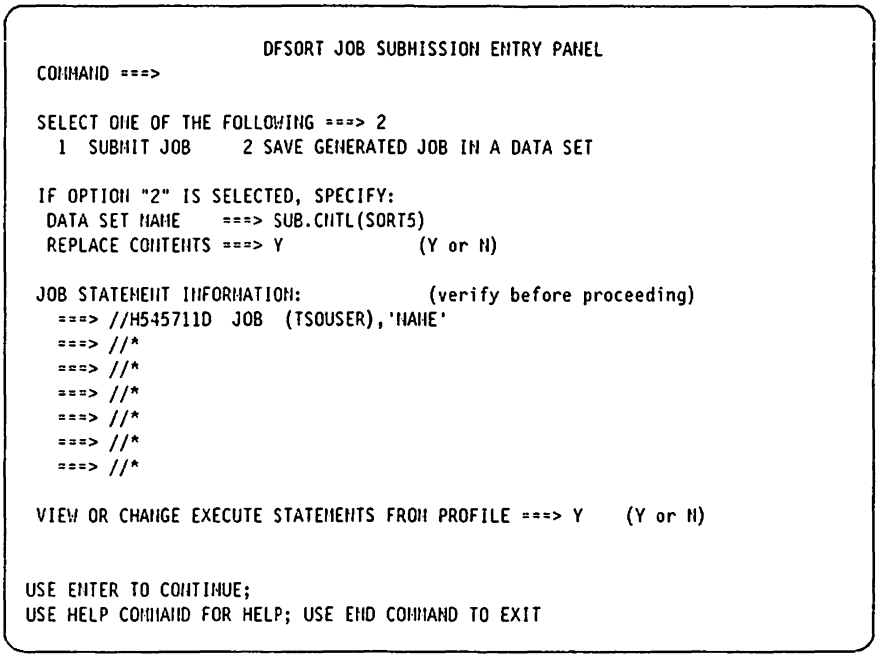
If you specify the foreground environment, DFSORT Panels automatically submits your completed application as a TSO job. TSO jobs are processed as soon as you complete all required panels, so the last panel that you see depends upon whether you are sorting, copying, or merging data sets, and whether you chose to work with or review any program control statements. While the job is running, your terminal displays TSO and DFSORT messages as they are issued. These messages contain important information about your job, including any errors detected by TSO or DFSORT, so examine them carefully when they appear. These messages are cleared from the screen when you press **ENTER**, and can be regenerated only by rerunning the job. After you clear the messages from the screen, DFSORT Panels returns to the first page of the SORT, COPY, or MERGE Entry panel that you used to begin building your job.

Using Background Execution

If you specify the background environment, you must use the DFSORT Job Submission Entry Panel to select whether to submit the job immediately, or save it in a data set for future use. As with foreground mode, the job is submitted or saved as soon as you complete all required panels.

Choosing to Save or Submit the Job: DFSORT Job Submission Entry Panel

To submit your background job to the execution queue, type **1** (SUBMIT JOB) in the SELECT ONE OF THE FOLLOWING field near the top of the panel. To retain the application for future use, specify 2 (SAVE GENERATED JOB IN A DATA SET), as shown in Figure 18. If you choose this option, you must also specify values in the DATA SET NAME and REPLACE CONTENTS fields to continue. The data set must be pre-allocated and cataloged with either fixed or fixed-block format and a logical record length of 80. A member of a partitioned data set (PDS) can be used provided that the PDS has the characteristics listed above.



```
                       DFSORT JOB SUBMISSION ENTRY PANEL
COMMAND ===>

SELECT ONE OF THE FOLLOWING ===> 2
  1  SUBMIT JOB     2 SAVE GENERATED JOB IN A DATA SET

IF OPTION "2" IS SELECTED, SPECIFY:
 DATA SET NAME    ===> SUB.CNTL(SORT5)
 REPLACE CONTENTS ===> Y            (Y or N)

JOB STATEMENT INFORMATION:           (verify before proceeding)
  ===> //H545711D  JOB  (TSOUSER),'NAME'
  ===> //*
  ===> //*
  ===> //*
  ===> //*
  ===> //*
  ===> //*

VIEW OR CHANGE EXECUTE STATEMENTS FROM PROFILE ===> Y    (Y or N)


USE ENTER TO CONTINUE;
USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT
```

Figure 18. DFSORT Job Submission Entry Panel

You can use the REPLACE CONTENTS field either to write over previous contents of the data set or member, or to append your current job to the end of an existing data set or member. Specify **Y** to write over data, or **N** to concatenate your completed DFSORT job to the previous contents.

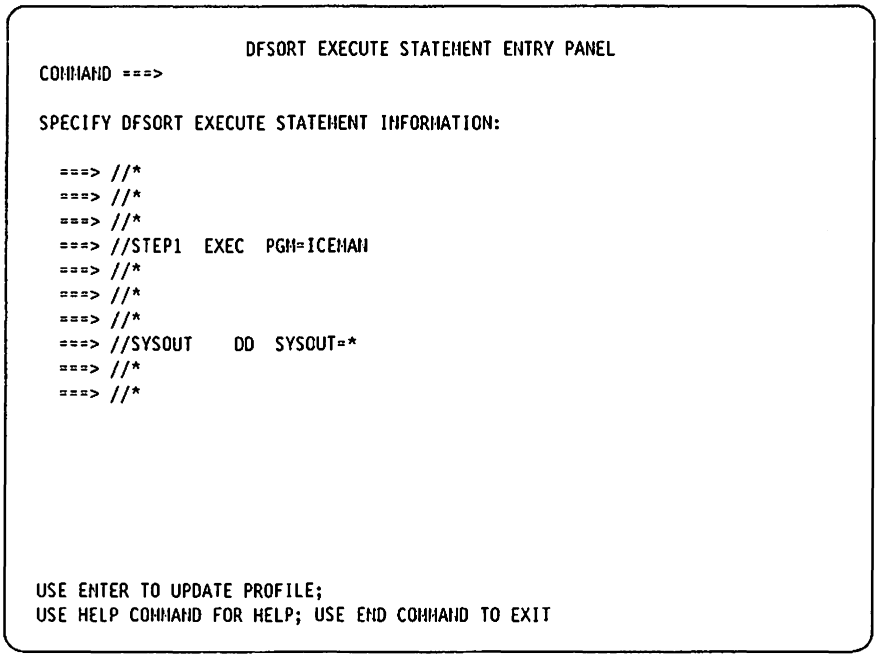
The DFSORT Job Submission Entry Panel also displays current JOB statements in your DFSORT profile. This JCL typically includes a JOB, JOBLIB, or JOBCAT statement, and JES2 or JES3 control statements. You can inspect or edit up to seven lines of JCL on this panel.

To edit the JOB statement information for your current application (and also to update your default values) before you save or submit it, type over the existing text with the statements you want to use. The values are temporarily updated in your DFSORT user profile when you save your profile changes. New values are saved *permanently* in your DFSORT user profile *only* when you completely exit DFSORT Panels. If you modify the profile information but encounter an abend before exiting DFSORT Panels, then your modifications might be lost. To work with the EXEC statement information to be attached to your job, type **Y** in the VIEW OR CHANGE EXECUTE STATEMENTS FROM PROFILE field.

Note: DFSORT Panels generates SORTIN, SORTOUT, SORTWK01, and SYSIN job control statements automatically, but reads the JOB and EXEC statements from your DFSORT user profile. See Chapter 2, "Executing DFSORT with Job Control Language" on page 15 for information on coding job control language for DFSORT Panels.

Editing the JCL EXEC Statement: DFSORT Execute Statement Entry Panel

You see this panel only if you specified **Y** in the VIEW OR CHANGE EXECUTE STATEMENTS FROM PROFILE field on the previous panel. It displays the current job control language EXEC statements, stored in your DFSORT profile, that are used to process your DFSORT job (see Figure 19). As with the previous panel, you can verify or change statements here before you save or submit the job. Remember that the values are temporarily updated in your DFSORT user profile when you save your profile changes. New values are saved *permanently* in your DFSORT user profile *only* when you completely exit DFSORT Panels. If you modify the profile information but encounter an abend before exiting DFSORT Panels, then your modifications might be lost.



```
                    DFSORT EXECUTE STATEMENT ENTRY PANEL
COMMAND ===>

SPECIFY DFSORT EXECUTE STATEMENT INFORMATION:

  ===> //*
  ===> //*
  ===> //*
  ===> //STEP1  EXEC  PGM=ICEMAN
  ===> //*
  ===> //*
  ===> //*
  ===> //SYSOUT    DD  SYSOUT=*
  ===> //*
  ===> //*







USE ENTER TO UPDATE PROFILE;
USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT
```

Figure 19. DFSORT Execute Statement Entry Panel

The JCL on this panel typically includes an EXEC statement that specifies PGM=SORT, PGM=ICEMAN, or a PROC name, and a SYSOUT DD statement for DFSORT messages. You can work with up to 10 lines of JCL on the panel.

Note: DFSORT Panels generates SORTIN, SORTOUT, SORTWK01, and SYSIN job control statements automatically, but reads the JOB and EXEC statements from your DFSORT user profile. As before, see Chapter 2, "Executing DFSORT with Job Control Language" on page 15 for information on coding job control language for DFSORT Panels.

In both the DFSORT Job Submission Entry Panel and the DFSORT Execute Statement Entry Panel, you can reset any line of JCL to the defaults that appeared when you entered the panel. Simply type over the line or lines with blanks, and press **ENTER**. The default JCL reappears in the blanked lines, and you can retain it or modify it again. If you decide to modify the JCL, ensure that you follow the procedures and requirements set during installation of DFSORT at your site.

When you finish inspecting or changing the EXEC statements on the panel, press **ENTER** to save or submit your DFSORT job.

In Figure 20, you see an example of how a job stream created with DFSORT Panels might appear after you save it in a data set. The same job stream is submitted if you specified background execution. (Jobs executed in the foreground environment use CLIST processing.)

```
//H545711D  JOB  (TSOUSER),'NAME'
//*
//*
//*
//*
//*
//*
//*
//*
//*
//STEP1  EXEC  PGM=ICEMAN
//*
//*
//SYSOUT   DD  SYSOUT=*
//*
//*
//*
//SORTIN    DD    DISP=SHR,
//              DSN=H545711.SAMPLE.DATA1
//SORTOUT   DD    DISP=OLD,
//              DSN=H545711.SAMPLE.SORTOUT
//SORTWK01  DD    UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSIN     DD    *
 SORT     FIELDS=(1,75,CH,A)
/*
```

Figure 20. An Example of a Completed Job for Background Execution or Saving

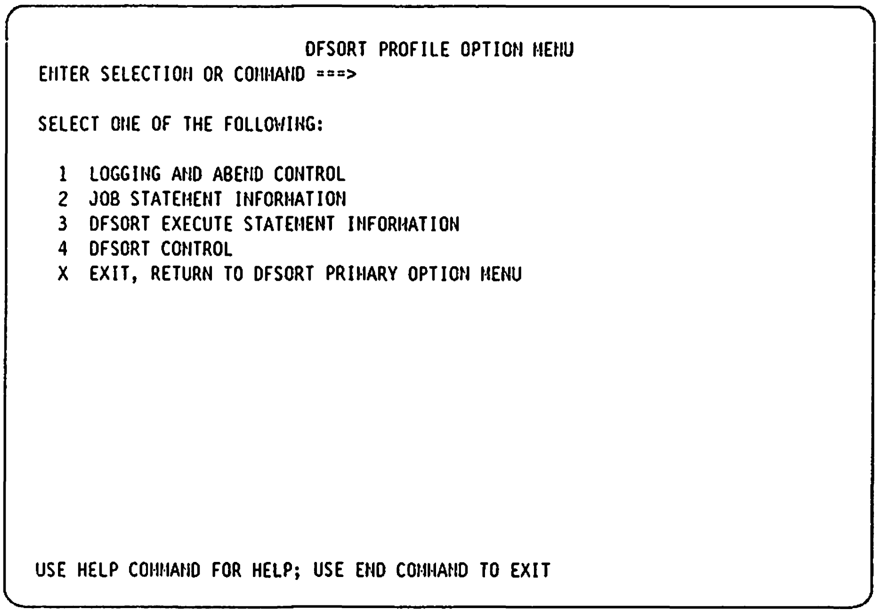
Changing Your DFSORT User Profile

The information in your DFSORT user profile controls the way DFSORT Panels operates during your terminal session. The profile automatically controls:

- The logging of trace data
- How DFSORT Panels recovers from abends
- What job control language (JCL) is used to process background jobs
- Whether to issue diagnostic messages
- Whether to convert your free form entries to uppercase.

After the correct values are set during your initial use of the panels, you normally do not need to use or see the profile to create DFSORT applications.

The Profile application can be accessed from the DFSORT Primary Option Menu, along with the sort, copy, and merge applications. Specify the appropriate selection to get to the DFSORT Profile Option Menu, shown in Figure 21.



```
                              DFSORT PROFILE OPTION MENU
ENTER SELECTION OR COMMAND ===>

SELECT ONE OF THE FOLLOWING:

  1  LOGGING AND ABEND CONTROL
  2  JOB STATEMENT INFORMATION
  3  DFSORT EXECUTE STATEMENT INFORMATION
  4  DFSORT CONTROL
  X  EXIT, RETURN TO DFSORT PRIMARY OPTION MENU









USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT
```

Figure 21. DFSORT Profile Option Menu

You can also access the profile directly by entering the **PROFILE** command on the command line of any sort, merge, or copy functional panel.

The choices in the profile menu correspond to the panels on which you can view or change operating defaults. To see any of the profile panels, specify the selection corresponding to that panel and press **ENTER**. Make any necessary changes to the entries on the panel, and press **ENTER** to save them as your new defaults. Then use **END** to return to the DFSORT Profile Option Menu.

To return to the DFSORT Profile Option Menu without changing the settings in a profile panel, use the **END** command. Using **END** erases any changes made to a panel, so long as they were not previously saved with **ENTER**.

Note: Remember that the values are temporarily updated in your DFSORT user profile when you save your profile changes. New values are saved *permanently* in your DFSORT user profile *only* when you completely exit DFSORT Panels. If you modify the profile information but encounter an abend before exiting DFSORT Panels, then your modifications might be lost.

If your system programmers installed DFSORT Panels as an option under ISMF, then some modifications you make in your DFSORT User Profile also update the settings in your ISMF Profile. The entries on the Logging/Abend Control Entry Panel and the JOB Statement Entry Panel in DFSORT Panels occupy the same storage as the corresponding entries in the ISMF Profile, so any changes you make to these values in one profile also update the values in the other.

Note: If you select DFSORT from ISPF (outside of ISMF), then the ISMF Profile is unaffected by changes you make in DFSORT Panels.

Four panels control the DFSORT user profile. They are discussed in sequence below, along with the defaults that you can modify with them.

Handling Trace Data and Abends: Logging/Abend Control Entry Panel

Use this panel to specify values that control whether to log inter-module trace data in the ISPF log data set, and whether DFSORT Panels should be allowed to recover from an abend, with a record of the results of a DFSORT Panels symptom dump being placed in the ISPF log data sets.

Specify **Y** in the LOG INTER-MODULE TRACE field to record the trace data in the ISPF log data set. A DFSORT Panels trace record in the log has a FUNCTION ID of "TRACE," and includes the following data: MODULE ID(xx). An **N** in the field results in no trace data being recorded in the ISPF log data set.

Note: The LOG INTER-MODULE TRACE is intended only for diagnosing problems. You will get better performance from DFSORT Panels if you leave the field set to **N**.

Specify **Y** in the RECOVER FROM ABENDS field to direct DFSORT Panels to attempt recovery from abends, and to record the results of a DFSORT Panels symptom dump in the ISPF log data set. Specify **N** if you want DFSORT to cancel an operation without a symptom dump if an abend occurs.

Specifying DFSORT Panels Job Control Language

Two panels in your user profile control the default JCL that the panels generate when you process jobs in the background environment. Unless you save updated statements when the job is submitted or saved in a data set for future use, the statements in the profile are used for every background job. See Chapter 2, "Executing DFSORT with Job Control Language" on page 15 for details on JCL for DFSORT. If you decide to modify the JCL used to process your job, ensure that you follow the procedures and requirements set during installation of DFSORT at your site.

To reset a line on the panel to the default, type over the entry with blanks and press **ENTER**. DFSORT Panels resets the field to the current default value.

When you execute DFSORT to submit or save your application, ISPF generates the necessary job control language line by line from the entries on the first and the second panels. This permits you quite a bit of flexibility in coding your job control language. You can enter both JOB and EXEC statements on either panel, so long as the JOB statements appear *before* the EXEC statements. Because ISPF reads the JOB Statement Entry Panel *before* reading the DFSORT Execute Statement Entry Panel, you must ensure the correct order of your statements if you choose to combine them between panels.

Note: DFSORT Panels generates SORTIN, SORTOUT, SORTWK01, and SYSIN job control statements automatically, but reads the JOB and EXEC statements from your DFSORT user profile. As before, see Chapter 2, "Executing DFSORT with Job Control Language" on page 15 for information on coding job control language for DFSORT Panels.

Specifying the JOB Statement: JOB Statement Entry Panel

Use this panel to specify up to seven lines of job control language for processing background jobs. This JCL typically includes a JOB, JOBLIB, or JOBCAT statement, and JES2 or JES3 control statements.

Specifying the EXEC Statement: DFSORT Execute Statement Entry Panel

Ten more lines are available to you here for EXEC statements and any other JCL that you might want to use in your jobs. This JCL typically includes an EXEC statement that specifies PGM=SORT, PGM=ICEMAN, or a PROC name, and a SYSOUT DD statement for the DFSORT message set.

Handling Messages and Case Conversion: DFSORT Control Entry Panel

The two fields on this panel control whether DFSORT issues diagnostic messages, and whether your entries on Free Form Entry panels should automatically be converted to uppercase.

Requesting Diagnostic Messages

DFSORT normally issues informational and error messages during execution. Additional diagnostic messages can be requested to document EXCP counts, buffer allocation, and processing techniques.

You might want to see these diagnostic messages to tune your application or determine why a particular processing technique could not be used. To record DFSORT diagnostic messages, specify **Y** in the ISSUE DFSORT DIAGNOSTIC MESSAGES field. DFSORT automatically generates a SORTDIAG DD statement for messages from background execution, or the corresponding ALLOCATE command to achieve the same result from foreground execution under TSO. Specifying **N** in the field directs DFSORT not to issue diagnostic messages.

Requesting Conversion from Lower to Uppercase

Lowercase input in DFSORT statements is valid only in comments or in character constants within the INCLUDE or OMIT control statements. You can ensure that all entries on the Free Form panels are converted to uppercase automatically. Enter **Y** in the CONVERT FREE FORM ENTRY TO UPPERCASE field. This will probably be your most common setting. If you do need to enter lowercase text in a Free Form panel, select **N** in the field to prevent automatic conversion. You must then use uppercase text in the control statements wherever it is required.

If you select **N** to retain lowercase text on a Free Form panel, but later change your selection to **Y**, note that your input is converted to uppercase if you save entries on the Free Form panel again. Do not change the select code from **N** to **Y** until you no longer require lowercase input.

Each Free Form panel tells you whether the lowercase to uppercase conversion is ENABLED or DISABLED.

Using SORTREC for a Quick and Simple Sort Under ISMF

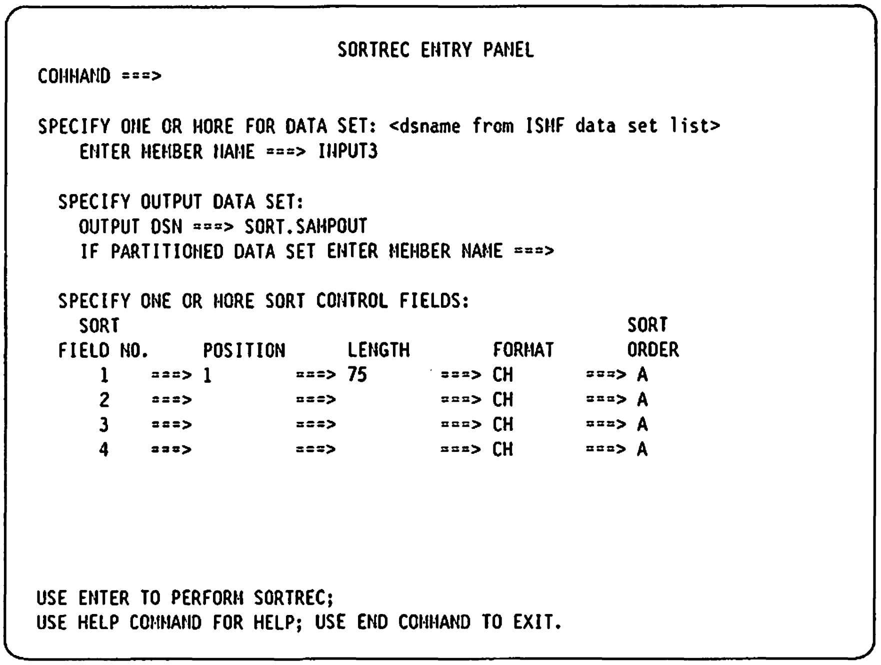
If your site has DFSORT Panels installed as an ISMF option, you can perform quick, basic sorts using SORTREC, an ISMF data set application line operator. SORTREC lets you sort by up to four control fields at a time, using a single entry panel.

Select the ISMF data set application option from the ISMF Primary Option Menu, and generate your data set list using the Data Set Selection Entry Panel. When the Data Set List panel is displayed, enter **SORTREC** in the line operator field next to the name of the data set whose records you want to sort.

When the SORTREC Entry Panel is displayed, the data set name from your ISMF data set list appears automatically in the SPECIFY ONE OR MORE FOR DATA SET field. There are three versions of the SORTREC Entry Panel. The format varies slightly depending upon whether your input data set is physical sequential, partitioned, or VSAM (the illustration shows the most complex version).

If you are sorting a partitioned input data set, specify the member name in the ENTER MEMBER NAME field of that SORTREC panel. If you are sorting a VSAM input data set, specify whether the input record length is fixed or variable by entering **F** or **V** in the ENTER RECORD TYPE (F OR V) field. Specify the name of your output data set in the OUTPUT DSN field, and the member name, if any, in the field labeled IF PARTITIONED DATA SET ENTER MEMBER NAME.

Complete the remaining fields on the SORTREC Entry Panel (see Figure 22 on page 175) in the same manner as on the SORT Control Fields Entry panel. Type the position of the first byte of your first control field (the first position in the record is byte 1) into the POSITION field, and its length in bytes into the LENGTH field. Remember that the Record Descriptor Word (RDW) occupies bytes 1-4 in variable-length records. Indicate the format of data in the FORMAT field (valid format codes are listed in Table 8 on page 135). Specify **A** or **D** in the SORT ORDER field to select ascending or descending sort order, respectively.



```
                              SORTREC ENTRY PANEL
COMMAND ===>

SPECIFY ONE OR MORE FOR DATA SET: <dsname from ISMF data set list>
    ENTER MEMBER NAME ===> INPUT3

  SPECIFY OUTPUT DATA SET:
    OUTPUT DSN ===> SORT.SAMPOUT
    IF PARTITIONED DATA SET ENTER MEMBER NAME ===>

  SPECIFY ONE OR MORE SORT CONTROL FIELDS:
    SORT                                                    SORT
  FIELD NO.     POSITION      LENGTH        FORMAT        ORDER
      1     ===> 1        ===> 75       ===> CH       ===> A
      2     ===>          ===>          ===> CH       ===> A
      3     ===>          ===>          ===> CH       ===> A
      4     ===>          ===>          ===> CH       ===> A




USE ENTER TO PERFORM SORTREC;
USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT.
```

Figure 22. SORTREC Entry Panel for a Partitioned Input Data Set

Submit the job for processing by pressing **ENTER**. Only foreground execution under TSO is available with the SORTREC line operator, so the job occupies your terminal until it completes. The SORTREC Entry Panel is redisplayed after each job completes. You can modify any field except the data set name from the ISMF data set list and submit another job for processing.

Because SORTREC operates under ISMF, entering the **PROFILE** command from the SORTREC Entry Panel invokes the ISMF Profile Application, not the DFSORT Profile Application. Using the **END** command without pressing **ENTER** returns you to the ISMF Data Set List Panel without saving any changes.

For additional information on using the ISMF line operators, including SORTREC, see the *MVS/Extended Architecture Interactive Storage Management Facility User's Guide.*

Getting HELP

DFSORT Panels supports an extensive interactive HELP facility that gives you online information about both DFSORT Panels and the DFSORT program. HELP is keyed to the individual DFSORT panels, so you can also use it as a tutorial to locate specific information and examples while you construct your job. HELP can also explain error messages you see on the panels when you make typing mistakes or when DFSORT cannot perform an action you want.

Getting HELP with DFSORT Panels

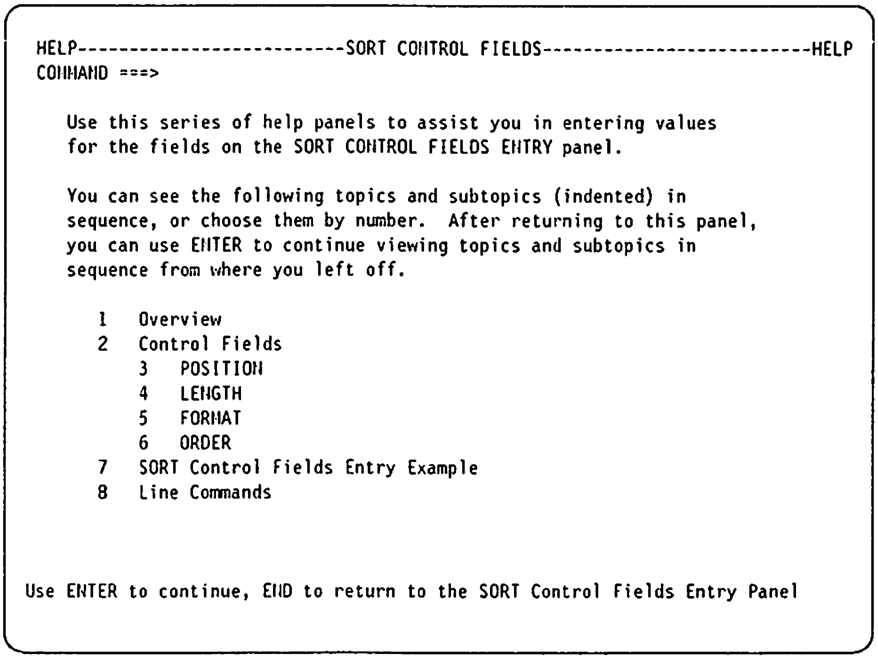
To access HELP, enter **HELP** on the command line in any panel, or press the appropriate **PF** key.

The first HELP panel you see usually displays a selection menu that lists specific topics that you can select for details. To see the HELP panels for additional topics, enter the selection code corresponding to your choice in the command line and press **ENTER**.

You can page through all the HELP panels that correspond to a particular DFSORT functional panel by pressing **ENTER** on the first panel, and then on all subsequent HELP panels in turn. You can return at any time to the original DFSORT functional panel from which you invoked HELP by using the **END** command.

Using the **UP** command always returns you to the HELP selection menu that includes your current HELP panel as a selection option. Using **UP** gets you to the menu that displays other related topics when you are done with your current one.

As an example, say you want online HELP while entering data in the SORT Control Fields Entry panel (illustrated in Figure 14 on page 154). If you enter **HELP** from the SORT Control Fields Entry panel, you see the HELP panel shown in Figure 23.



```
HELP-------------------------SORT CONTROL FIELDS-------------------------HELP
COMMAND ===>

   Use this series of help panels to assist you in entering values
   for the fields on the SORT CONTROL FIELDS ENTRY panel.

   You can see the following topics and subtopics (indented) in
   sequence, or choose them by number.  After returning to this panel,
   you can use ENTER to continue viewing topics and subtopics in
   sequence from where you left off.

      1   Overview
      2   Control Fields
             3   POSITION
             4   LENGTH
             5   FORMAT
             6   ORDER
      7   SORT Control Fields Entry Example
      8   Line Commands



Use ENTER to continue, END to return to the SORT Control Fields Entry Panel
```

Figure 23. SORT Control Fields Entry HELP Panel

The HELP panel lists the topics you can select for specific information.

Getting HELP for Messages

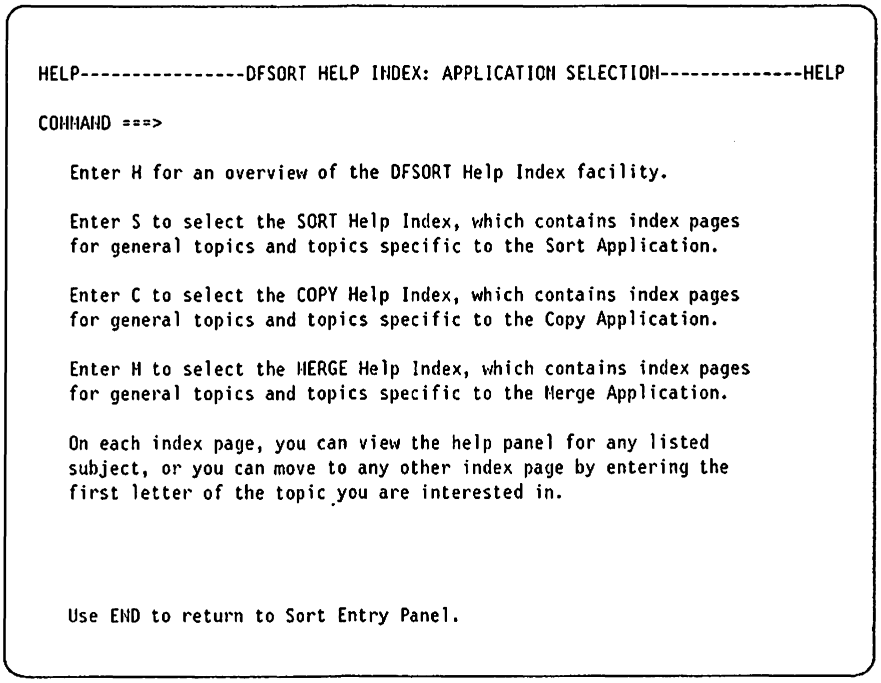
You can also use HELP to get information about DFSORT Panels informational and error messages. During your terminal session, DFSORT Panels issues short messages that appear in the upper right corner of the screen. For example, a message might explain that a command you entered was inappropriate, or that the job you submitted has completed successfully.

To see a more detailed version of the message, use the **HELP** command or press the appropriate **PF** key. A longer version of the error message then appears under the command line.

If you need more detail about why the message occurred, and what you might need to do to correct the problem, use the **HELP** command or **PF** key again to display the HELP panel for that message. Pressing **ENTER** at this point displays the HELP panel for the functional DFSORT panel from which you originally entered **HELP**. Use **END** to return to the DFSORT functional panel when you are done.

Using the HELP Index

To display the DFSORT help index, enter **INDEX** or **I** on the command line of any DFSORT help panel.

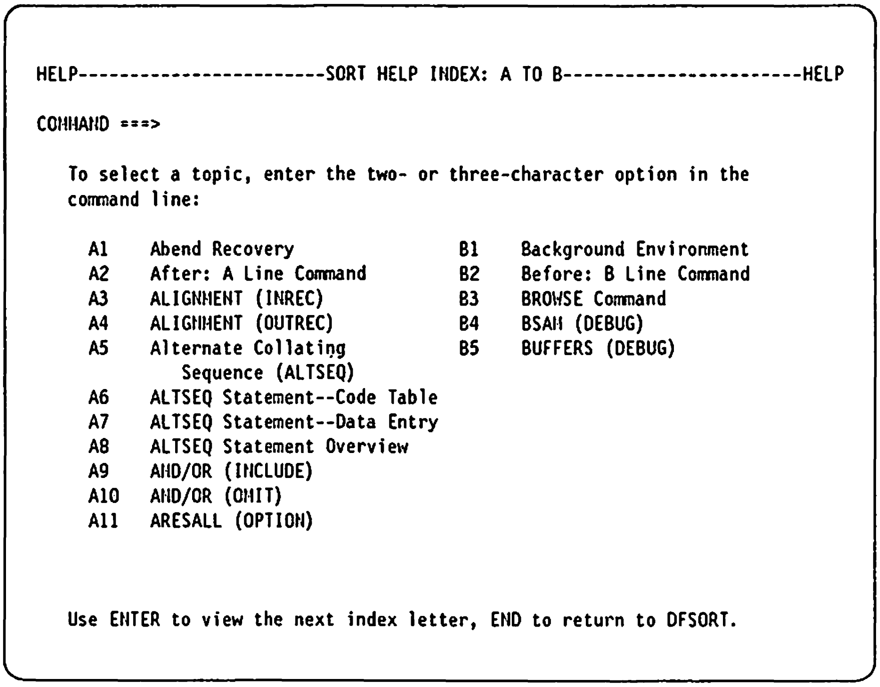


```
HELP----------------DFSORT HELP INDEX: APPLICATION SELECTION--------------HELP

COMMAND ===>

   Enter H for an overview of the DFSORT Help Index facility.

   Enter S to select the SORT Help Index, which contains index pages
   for general topics and topics specific to the Sort Application.

   Enter C to select the COPY Help Index, which contains index pages
   for general topics and topics specific to the Copy Application.

   Enter M to select the MERGE Help Index, which contains index pages
   for general topics and topics specific to the Merge Application.

   On each index page, you can view the help panel for any listed
   subject, or you can move to any other index page by entering the
   first letter of the topic you are interested in.



   Use END to return to Sort Entry Panel.
```

Figure 24. DFSORT Help Index Application Selection Panel

The first index panel (see Figure 24) is a special Application Selection panel. Enter **S**, **C**, or **M** to use the index for SORT, COPY, or MERGE, respectively. Each index includes common information on general topics, as well as specific information for the application selected. You can also enter **H** to see an overview of the DFSORT Panels help index.

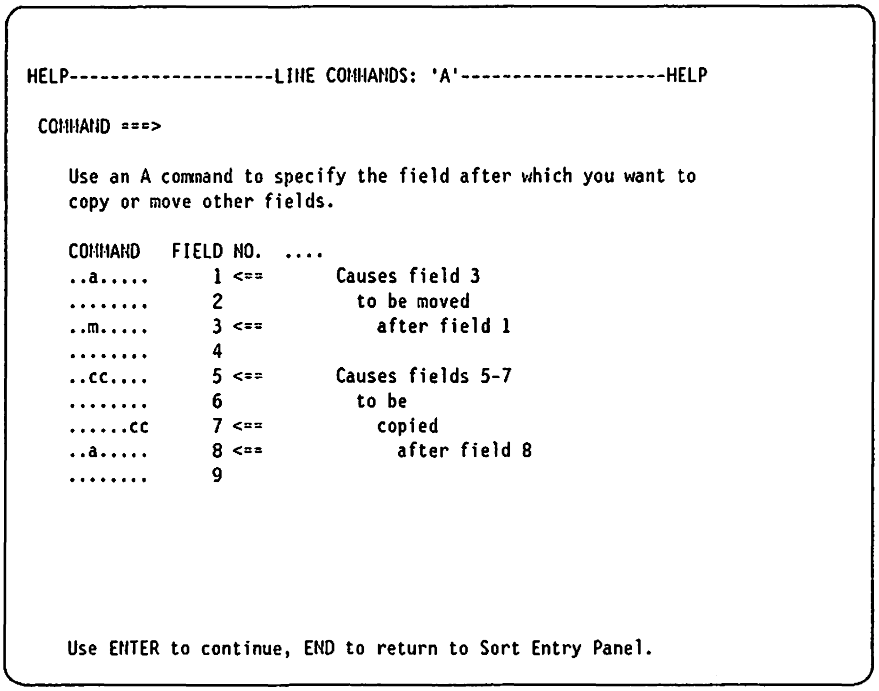
If you specify **S**, for instance, you see the beginning of an alphabetic listing of all help topics in the SORT index (see Figure 25 on page 178).



```
HELP-----------------------SORT HELP INDEX: A TO B----------------------HELP

COMMAND ===>

   To select a topic, enter the two- or three-character option in the
   command line:

     A1    Abend Recovery              B1    Background Environment
     A2    After: A Line Command       B2    Before: B Line Command
     A3    ALIGNMENT (INREC)           B3    BROWSE Command
     A4    ALIGNMENT (OUTREC)          B4    BSAM (DEBUG)
     A5    Alternate Collating         B5    BUFFERS (DEBUG)
              Sequence (ALTSEQ)
     A6    ALTSEQ Statement--Code Table
     A7    ALTSEQ Statement--Data Entry
     A8    ALTSEQ Statement Overview
     A9    AND/OR (INCLUDE)
     A10   AND/OR (OMIT)
     A11   ARESALL (OPTION)


   Use ENTER to view the next index letter, END to return to DFSORT.
```

Figure 25. Help Index for SORT

To see the help panel for any topic on an index panel, enter its two- or three-letter select code on the command line. If you chose **A2**, for example, on the SORT HELP INDEX: A TO B panel, you would see the following panel.



```
HELP--------------------LINE COMMANDS: 'A'--------------------HELP

 COMMAND ===>

    Use an A command to specify the field after which you want to
    copy or move other fields.

    COMMAND   FIELD NO.  ....
    ..a.....       1 <==      Causes field 3
    ........       2            to be moved
    ..m.....       3 <==          after field 1
    ........       4
    ..cc....       5 <==      Causes fields 5-7
    ........       6            to be
    ......cc       7 <==          copied
    ..a.....       8 <==            after field 8
    ........       9




    Use ENTER to continue, END to return to Sort Entry Panel.
```

Figure 26. Help Panel for Line Commands

The **END** command returns you to the functional panel from which you first requested help. **ENTER** returns you to the previous index panel (in this example, SORT HELP INDEX: A TO B).

To see the help panel for a topic not on the index panel currently displayed, enter the first letter of that topic in the command line and press **ENTER**. DFSORT then displays the index panel for that topic, and you can select the appropriate two- or three-letter select code. For example, to see the help panel for the REINIT command, enter **R**, to get to the appropriate index panel, and then **R9** to get the help screen for REINIT. You can also page through the index panels alphabetically by pressing **ENTER**.

Chapter 5. Using Your Own Exit Routines

DFSORT can pass control to your own routines at points in the executable code called *user exits*. Your user exit routines can perform a variety of functions, including deleting, inserting, altering, and summarizing records.

If you do need to perform these tasks, however, you should be aware that DFSORT already provides extensive facilities for working with your data in the various DFSORT program control statements. See the discussions of the INCLUDE, OMIT, INREC, OUTREC, and SUM program control statements in Chapter 3, "Using DFSORT Program Control Statements" on page 53. You might decide that using a program control statement to work with your records is more appropriate to your needs.

Although this chapter discusses only routines written in assembler or COBOL, you can write your exit routines in any language that can:

- Pass and accept the address into general register 1 of a:
  - Record
  - Fullword of zeros
  - Parameter list
- Pass a return code in register 15.

You can activate user exit routines during execution most easily with the MODS program control statement (see "MODS Control Statement" on page 91). Alternatively, under certain circumstances you can also activate a user exit routine by passing the address of your exit routine in the invocation parameter list. See Chapter 6, "Invoking DFSORT from a Program" on page 231 for details.

Parameters that affect the way user exit routines are handled include:

- The MODS program control statement, explained in "MODS Control Statement" on page 91.
- The COBEXIT option of the ICEMAC installation macro, explained in "Installation Defaults" on page 11.
- The E15=COB and E35=COB PARM options of the EXEC statement, explained in "Specifying EXEC Statement PARM Options" on page 21.
- The COBEXIT option of the OPTION program control statement, explained in "OPTION Control Statement" on page 97.
- The ICEMAC installation option EXITCK, explained in "E15/E35 Return Codes and EXITCK" on page 226.

  Note: To avoid ambiguity in this chapter, it is assumed that the IBM default, EXITCK=STRONG, was selected at your site.

Certain user exit routines can be written in COBOL, using a special interface. If you write your exit routines in PL/I, you must use the PL/I subroutine facilities.

You might need to reserve space to be used by your exits. See "Using Main Storage Efficiently" on page 256 for more information about storage.

DFSORT Program Phases

A DFSORT program phase is a large DFSORT component designed to perform a specific task, such as writing the output file. Various user exits are contained in the input and output phases, and are activated at a particular time during DFSORT execution. Figure 27 on page 184 is a representation of DFSORT input/output logic.

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. All sorting techniques use this phase.

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.
- It copies each input record to the output data set for a copy application.

After this phase runs, DFSORT returns control to the operating system (or invoking program).

Functions of Routines at User Exits

You can use exit routines to accomplish a variety of tasks:

- Open and initialize data sets
- Modify control fields
- Insert, delete, or alter records
- Summarize records
- Handle special I/O conditions
- Close data sets
- Terminate DFSORT.

Figure 27 on page 184, Figure 28 on page 186, and Figure 29 on page 187 summarize the functions of user exit routines and the exits and phases with which they can be associated.

DFSORT Input/Exit/Output Logic Examples

Figure 27 on page 184 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.)

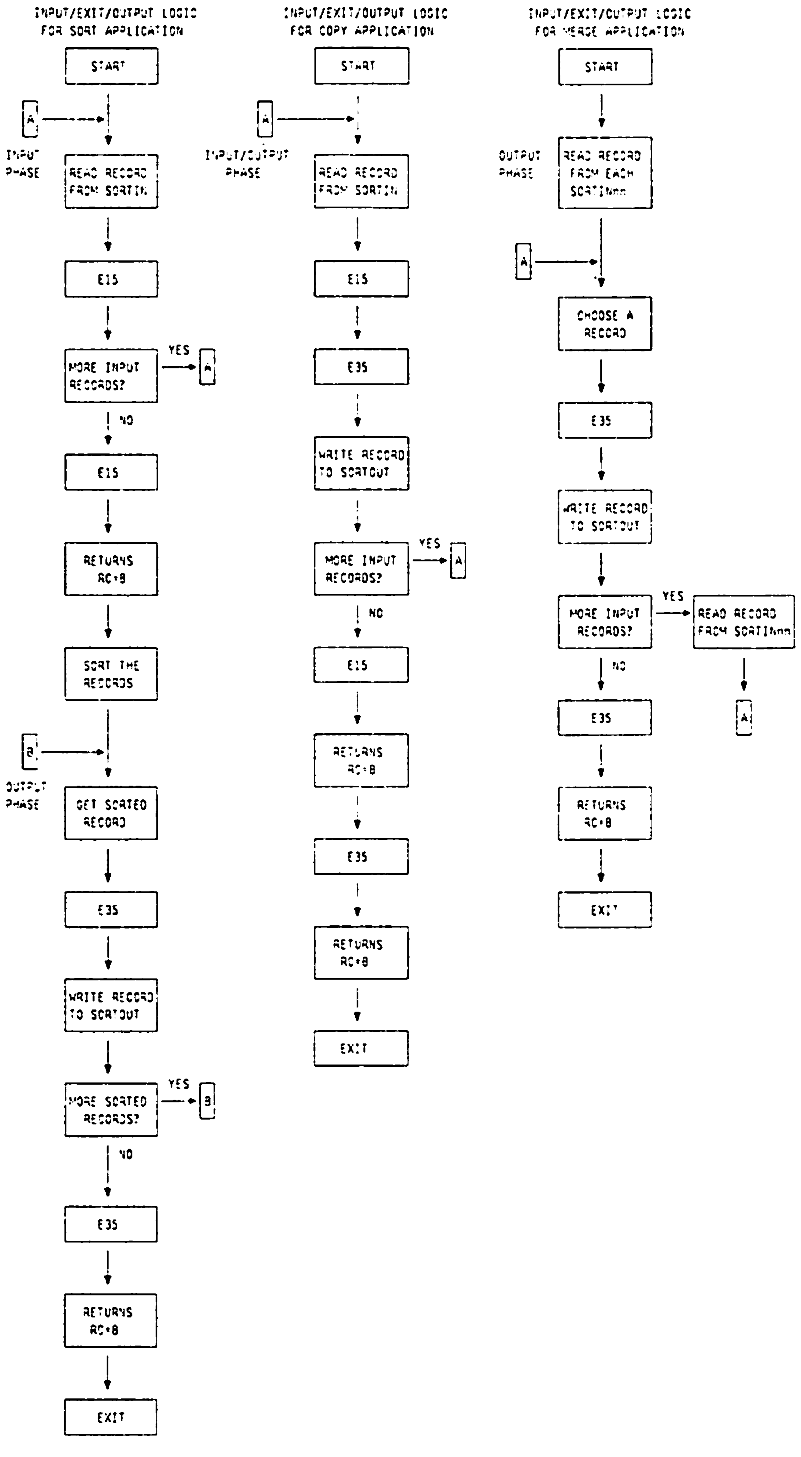


Figure 27. Examples of DFSORT Input/Exit/Output Logic

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 when there are still input records to be processed, the records are processed by DFSORT *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 by DFSORT. 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.

Opening and Initializing Data Sets

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 E35 exits.

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

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 DFSORT's ALTSEQ program control statement instead. You must associate this routine with the E61 exit.

Your routine modifies a copy of the control fields, which is used for comparison. It does not change the original control fields. Thus your original records are not altered. When an E61 exit is used, the subsequent comparisons always arrange the modified control fields in ascending order.

Inserting, Deleting, and Altering Records

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

Note: DFSORT also provides INCLUDE and OMIT statements that automatically include or delete records based on your field criteria. For more information on these control statements, refer to Chapter 3, "Using DFSORT Program Control Statements" on page 53.

Summing Records

You can sum records in the output data set by using the E35 exit. However, you can also use DFSORT's SUM program control statement to accomplish this without an exit routine. See "SUM Control Statement" on page 141.

Handling Special I/O

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 site''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 uncorrec-

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

Notes to Figure 28:

[1] The SUM control statement can be used instead of your own routine to sum records.

[2] Applies only to a tape work data set sort.

Figure 28. Functions of Routines at Program Exits (Sort)

table 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 exit, as appropriate. Your routine must return to VSAM via register 14. For details, see *VSAM Programmer's Guide* or *VSAM User's Guide*.

Routines for Read Errors: You must associate these routines with the E18 and 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 can also attempt to correct the error.

Functions	Copy	Merge
Open	E31, E15	E31
Insert	E15, E35	E32, E35
Delete/alter	E15, E35	E35
Terminate DFSORT	E15, E35	E32, E35
Sum 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

Note to Figure 29:

[1] The SUM control statement can be used instead of your own routine to sum records.

Figure 29. Functions of Routines at Program Exits (Copy and Merge)

Routines for Write Errors: You must associate these routines with the E19 and 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. They can 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.

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 313.

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 E37 exits. 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.

Terminating DFSORT

You can write an exit routine to terminate DFSORT before all records have been processed. You must associate these routines with the E15, E16, E32, and E35 exits.

MVS/XA and MVS/ESA Support of User Exits

To allow user exits called by Blockset or Peerage/Vale (executing in an MVS/XA or MVS/ESA system) to reside above or below 16-megabyte virtual, and to 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 can 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.

How Assembler and COBOL User Exit Routines Affect DFSORT Performance

Before writing a user exit, consider the following factors:

- Your routines occupy main storage that would otherwise be available to DFSORT. Because its main storage is restricted, DFSORT might need to execute extra passes to sort the data. This, of course, increases sorting time.

- User exit routines increase the overall run-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 must 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 that:

- 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 91.
- Your invoking program can alternatively pass the address of an E15, E18, E35, and 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 length you specify for an exit must include storage for the exit itself as well as any storage used by the exit outside of the load modules, such as I/O buffers or COBOL library subroutines. If you specify a ddname for an exit in the MODS statement, it must match the DD statement that defines the library containing that exit. For example:

```
//MYLIB DD   DSNAME=MYRTN, etc.
        .
        .
        .
  MODS   E15=(MODNAME,500,MYLIB,N)
```

User Exit Linkage Conventions

To enter a user exit, DFSORT loads the address of the DFSORT return point in register 14 and the address of the user exit routine in register 15. A branch to the address in register 15 is then executed.

The general registers used by DFSORT for linkage and communication of parameters observe operating system conventions (see Figure 30 on page 191).

You can return control to DFSORT by executing a branch to the DFSORT return point address in register 14 or by using a RETURN macro instruction. The RETURN instruction can also be used 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 general 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 cannot 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 module's RMODE; they are entered in the addressing mode of the exit.

Note:

- The Blockset technique is not used for dynamic link-editing.
- Dynamic link-editing cannot be used with copy.

When the link-edit option T is specified for a user exit routine, that routine *must* contain an entry point whose name is that of the associated program exit. This is to accommodate special DFSORT dynamic link-edit requirements. For example, when the link-edit option T is specified on the MODS statement for

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 can 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. This register can be used as a base register for your exit, and your exit can also use it to pass return codes to DFSORT.

Figure 30. Register Conventions

E35, the following assembler instructions must be included in the user exit routine associated with the E35 exit:

```
        ENTRY E35
E35     .
        .
```

or

```
E35   CSECT
        .
  .     .
```

In all other circumstances, the user exit is *not* required to have an entry point whose name is that of the associated program exit.

Linkage Examples

When calling your exit, DFSORT 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's save area address in general register 13. DFSORT than makes a branch to your routine.

Your routine for the E15 exit might incorporate the following assembler 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 can 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*.

Assembler Exit Routines (Input Phase Exits)

You can use these program exits located in the DFSORT input phase:

E11
E15
E16
E17
E18
E19
E61

These exits are discussed in sequence. To determine whether a particular exit can be used for your application, refer to Figure 28 and Figure 29 on page 187. For example, E15 cannot be used for a merge application.

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 189), 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 209 and "COBOL E15 Exit: Passing or Changing Records for Sort" on page 212.

The ICEMAC installation option EXITCK affects the way DFSORT interprets certain return codes from user exit E15. To avoid ambiguity, this section assumes that the IBM default, EXITCK=STRONG, was selected at your site. For complete information about E15 return codes in various situations with EXITCK=STRONG and EXITCK=WEAK, see "E15/E35 Return Codes and EXITCK" on page 226.

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 27 on page 184 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 can declare the records as fixed-length, and pad them to the maximum length.)

Note:

- If you use the E15 exit to pass all your records to DFSORT, the SORTIN DD statement can be omitted, in which case you must include a RECORD statement in the program control statements.
- 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.
- 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 general 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."
- 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.
- Remember to build an RDW for variable-length VSAM records (see Data Management Services).

Information DFSORT Passes to Your Routine at E15

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, the address constant is passed to your E15 exit the first time it is entered. This address constant can be changed by your E15 exit any time it is entered; the address constant is passed along on subsequent entries to your E15 exit and also on the first entry to your E35 exit. For example, you can 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

E15 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 return code of 0 (zero).

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 return code of 0 (zero). 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 127 and *Supervisor Services and Macro Instructions* for further information about the length indicator and the RDW.)

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 general 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 general register 1 when you return with return code 8. *Unless you are inserting records after the end of the 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 general register 1 and return to 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.

See "E15/E35 Return Codes and EXITCK" on page 226 for complete details of the meanings of return codes in various situations.

E16 Exit: Handling Intermediate Storage Miscalculation

For a tape sort, you would use a routine at this exit to decide what to do if the 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 in that case DFSORT uses the WRKSEC option to determine whether secondary allocation is allowed. See "SORTWKnn DD Statement" on page 45. See also "Exceeding Intermediate Storage Capacity" on page 313.

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 can be exceeded before "NMAX EXCEEDED" is indicated.

E16 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 SKIPREC 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. If tapes are used, enough space might be available for DFSORT to complete processing, If enough space is not available, DFSORT generates a message and terminates. Refer to "Exceeding Intermediate Storage Capacity" on page 313.

8—Terminate DFSORT
If you want DFSORT to terminate, return with a return code of 8. DFSORT then returns to its calling program or to the system 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 189), you can include these functions in your E15 routines rather than in a separate E17 routine.

E18 Exit: Handling Input Data Sets

Using E18 with QSAM/BSAM

Your routines at this exit can pass DFSORT a parameter list containing the specifications for three data control block (DCB) fields (SYNAD, EXLST, and EROPT). 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 at E18: 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 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, must 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

This field indicates what action DFSORT must take when it encounters an uncorrectable read error. 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 contains the address of your end-of-file routine. If you specify EODAD, you must include the end-of-file routine in your own routine.

A full description of these DCB fields is contained in *Data Management Macro Instructions.*

Using E18 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.

E18 Restrictions: If passwords are to be entered through an exit and 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 at E18: When you return to DFSORT, you must place in general 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 address entry can be omitted; if they both are included, they can be in any order.

E18 Password List: A password list included in your routine must have the following format:

Two bytes on a halfword boundary:

Number 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 must not be altered at any time during the program. MAINSIZE|SIZE=MAX must not be used if this function is used.

E18 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 can 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 must not be altered when returning to VSAM, because 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'1'
         DC     CL8'SORTIN'      SORTIN DDNAME
         DC     CL8'INPASS'      SORTIN PASSWORD
USYNAD   ...                     VSAM SYNCH ERROR RTN
ULERAD   ...                     VSAM LOGIC ERROR RTN
SER      ...                     QSAM ERROR RTN
LST      DC     X'85',AL3(RTN)   EXLST ADDRESS LIST[1]
RTN      ...                     EXLST ROUTINE
QSAMEOD  ...                QSAM END OF FILE ROUTINE
```

[1] X'85' = X'80' plus X'05', where:

X'80' means this entry is the LAST ENTRY of the list.

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 is used only for a tape work data set sort.

Using E19 with QSAM/BSAM

Your routines at this exit can pass 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 at E19: 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 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]

Note: E61 will not be used with EFS fields that have a D1 format.

Some Uses of E61

Your routine can normalize floating-point control fields or change any other type of control field in any way that you desire. You need to be familiar with the standard data formats used by the operating system before modifying control fields.

If you want to modify the collating sequence of EBCDIC data, for example, to permit the alphabetic collation of national characters, you can do so without the need for an E61 exit routine using the ALTSEQ control statement (as described in Chapter 2).

[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.

Information DFSORT Passes to Your Routine at E61

DFSORT places the address of a parameter list in general register 1. The list begins on a fullword boundary and is three fullwords long. 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	

The control field length allows you to write a more generalized modification routine.

The control field number is relative to all fields in the SORT or MERGE statement. For example, if you specify:

```
SORT FIELDS=(4,2,CH,A,8,10,CH,E,25,2,BI,E)
```

field numbers 2 and 3 will be passed to E61.

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 that contain *any bits* of the control field are passed. If the control field is longer than 256 bytes, DFSORT splits it into fields of 256 bytes each and passes them one at a time to your routine.

Your routine cannot physically change the length of the control field. If you must increase the length for collating purposes, you must previously specify that length in the *m* parameter of the SORT or MERGE statement. If you must shorten the control field, you must pad it to the specified length before returning it to DFSORT. Your routine must return the field to DFSORT with the same number of bytes that it contained when your 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, because the field is only an extracted image.

Note that if E61 is used to resolve ISCII/ASCII collating for special alphabetic characters, substituted characters must be in EBCDIC, but the sequencing depends upon the byte value of the ISCII/ASCII translation for the substituted character.

Assembler Exit Routines (Output Phase Exits)

You can use these program exits located in the DFSORT output phase:

E31
E32
E35
E37
E38
E39

The functions of these exits are discussed in sequence.

E31 Exit: Opening Data Sets/Initializing Routines

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 189), 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 be used only in a merge operation invoked from another program, and cannot be specified on the MODS statement. When an E32 exit is activated, either (1) it must supply all input to the merge, and the parameter list passed to the program, or (2) a MERGE statement in SORTCNTL must indicate the number of input files.

If input is variable-length records, you must be sure the beginning of each record contains a 4-byte record descriptor word (RDW) before it is handed to the merge. The format of an RDW is described in *Data Management Services Guide*. (Alternatively, you can declare the records as fixed-length and pad them to the maximum length.)

See Figure 27 on page 184 for logic flow details.

Information DFSORT Passes to Your Routine at E32

Your E32 exit routine is entered each time the merge program requires a new input record. DFSORT passes a two-word parameter list to your routine. The address of the list is in general 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. Thus, the increment 0 (zero) represents the first input file, 4 the second file, 8 the third, and so on.

Your routine must provide a separate input buffer 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.

E32 Return Codes

Your routine must pass one of the following return codes to 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 209, and "COBOL E35 Exit: Changing Records" on page 218.

The ICEMAC installation option EXITCK affects the way DFSORT interprets certain return codes from user exit E35. To avoid ambiguity, this section assumes that the IBM default, EXITCK=STRONG, was selected at your site. For complete details of the meaning of E35 return codes in various situations with EXITCK=STRONG and EXITCK=WEAK, see "E15/E35 Return Codes and EXITCK" on page 226.

The E35 routine is entered each time DFSORT prepares to place a record in the output area.

See Figure 27 on page 184 for logic flow details.

Some uses are:

- Add, delete, or change records in the output data set.
- Terminate DFSORT.

Note:

- If you use the E35 routine to dispose of all your output records, the SORTOUT DD statement can be omitted.
- If you invoke DFSORT from a 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.
- 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."

- If your E35 routine is inserting variable-length records, you must be sure the beginning of each record contains a 4-byte record descriptor word (RDW) before the routine passes the record to DFSORT. The format of an RDW is described in *Data Management Services* or *System Programming Library: Data Management*. (Alternatively, you can declare the records as fixed-length, and pad them to the maximum length.)
- Remember that if input records are variable-length from a VSAM data set, they will have been prefixed by a 4-byte RDW.
- After records have been put into the output area, their lengths cannot be increased.
- For a merge job, records deleted by an E35 exit routine are not sequence-checked. If you use an E35 exit routine without a SORTOUT data set, sequence checking is not performed. In this case, you must ensure that the records are sequenced correctly.

Information DFSORT passes to Your Routine at E35

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 reached, 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 an RDW 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

E35 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 general register 1 and return to DFSORT with a return code of 0 (zero).

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 general register 1, and return to DFSORT with a return code of 0 (zero). 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 general 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 general 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
To add a record to the SORTOUT data set ahead of the record leaving DFSORT, place the address of the new record in general register 1 and return to DFSORT with a return code of 12. DFSORT returns to your routine with the same address as passed on the previous call to the exit for the record leaving DFSORT. DFSORT places the address of the inserted record into the output area. You can then 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.

See "E15/E35 Return Codes and EXITCK" on page 226 for complete details of the meanings of return codes in various situations.

Summing Records at E35

You can use the SUM control statement to sum records. However, you can sum 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: Closing Data Sets

Your routine at this exit is executed once at the end of i2.E37 exit 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 189), you can include these functions in your E35 routine rather than in a separate E37 routine.

E38: 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.

Using E38 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'2'
         DC       CL8'SORTIN01'    SORTIN01 DDNAME
         DC       CL8'INPASS1'     SORTIN01 PASSWORD
         DC       CL8'SORTIN02'    SORTIN02 DDNAME
         DC       CL8'INPASS2'     SORTIN02 PASSWORD
USYNAD   ...                       VSAM SYNCH ERROR RTN
ULERAD   ...                       VSAM LOGIC ERROR RTN
```

E39 Exit: Handling Output Data Sets

Using E39 with QSAM/BSAM

The technique is the same as for E19 for QSAM/BSAM. See "E19 Exit: Handling Output to Work Data Sets" on page 199 for details.

Using E39 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

This section provides some sample program exits written in assembler.

E15: Deleting Expired Records

This routine checks each record's expiration date, and deletes out-of-date records.

```
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    MORETRET           NO-END OF DATA SET
         L      13,SAVEAREA+4      YES-INPUT DATA SET EMPTY
         RETURN (14,12),RC=16      'TERMINATE SORT' CODE
MORETRET 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 Exit: When NMAX Exceeded, Sort Current Records

This routine tells DFSORT that, when DFSORT issues the message "NMAX EXCEEDED", it must sort only the records already read in.

```
E16    CSECT
       LA     15,0      SET RETURN CODE
       BR     14
       END
```

E35 Exit: Deleting Records

This routine checks byte 5 of each record. If the byte contains the letter N, the exit deletes the record. You can 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

You can perform the same tasks with E15 and E35 exit routines written in COBOL that you can perform with E15 and E35 exit routines written in Assembler. However, COBOL routines differ from assembler routines in the way they pass information between themselves and DFSORT. Your COBOL routine:

- Must pass information through fields described in the LINKAGE SECTION of the DATA DIVISION. Assembler uses general register 1 and pointers in a parameter list.
- Must use RETURN-CODE, a COBOL special register. Assembler uses register 15 for the return code.
- Must use return code 20 to alter or replace a record. Assembler uses return code 0.
- Can use the exit area for E15/E35 communication. Assembler uses the user address constant.

COBOL Exit Requirements

The following rules apply to COBOL exits. Failure to observe these COBOL exit rules can 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 must 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 must 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 can, therefore, be lost because of interleaving of output.

An alternative is to direct the COBOL output to another data set by 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.
- When coding the MODS control statement to describe a COBOL exit, use C for the fourth parameter. This instructs DFSORT to build the correct parameter list.
- 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 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 degrades performance.
- If COBEXIT=COB2 is in effect for this run, you must use the VS COBOL II library, even if your COBOL exit is compiled by the OS/VS COBOL compiler.
- If you run exits compiled with either the OS/VS COBOL compiler or the VS COBOL II compiler and you specify the RES option, the COBOL library routines must be available at run time. The COBOL library *might* be required for an exit compiled with the OS/VS COBOL, NORES option. See your OS/VS COBOL manual for information on options that require the COBOL library.
- Exits compiled with OS/VS COBOL can 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 must not issue STAEs unless the DFSORT NOESTAE option is in effect. (OS/VS COBOL compiler options that cause STAE to be issued are: STATE, FLOW, SYMDMP, COUNT, and TRACE.)

COBOL Requirements for Copy Processing

For copy processing, all sort requirements apply except for the following restrictions:

- When you invoke DFSORT through JCL and COBEXIT=COB2 is in effect, you can use *either* a separately compiled COBOL E15 exit *or* a separately compiled COBOL E35 exit, but not both.
- When you invoke DFSORT from a VS COBOL II program, the following limitations apply when FASTSRT is in effect for:
  - Input only: You can use a separately compiled E15 exit, but not a separately compiled E35 exit.
  - Output only: You can use a separately compiled E35 exit, but not a separately compiled E15 exit.
  - Input and output: You can use *either* a separately compiled E15 *or* a separately compiled E35, 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.

COBOL Storage Requirements

If you are running COBOL exits compiled with the RES compile-time 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, thus leaving no room to load the COBOL library subroutines required during execution of your exit.

On an MVS/XA or MVS/ESA 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 must 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 the note on page 211 for additional circumstances under which you might need to release additional storage for VS COBOL II.) Additional storage for buffers is specified by the *m* parameter on the MODS statement. A VS COBOL II exit requires less storage than a similar OS/VS COBOL exit 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: You might need to release an additional 70K bytes of storage when you are:

- Calling both E15 and E35 exits
- Running with nonresident VS COBOL II library subroutines
- Executing a sort with DFSORT residing above 16-megabyte virtual.

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.

COBOL Exit Routines (Input Phase Exit)

COBOL E15 Exit: Passing or Changing Records for Sort

The ICEMAC installation option EXITCK affects the way DFSORT interprets certain return codes from user exit E15. To avoid ambiguity, this section assumes that the IBM default, EXITCK=STRONG, was selected at your site. For complete details of the meaning of E15 return codes in various situations with EXITCK=STRONG and EXITCK=WEAK, see "E15/E35 Return Codes and EXITCK" on page 226.

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 27 on page 184 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).

Note:

- If both E15 and E35 exits are used, they must be in the same version of COBOL.
- If you use the E15 exit, you can omit the SORTIN DD statement as long as you include a RECORD statement in the program control statements.
- 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."
- 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.
- You cannot use dynamic link-editing with a COBOL E15 exit.

E15 Interface with COBOL

Each time the E15 exit is called, DFSORT supplies the following fields:

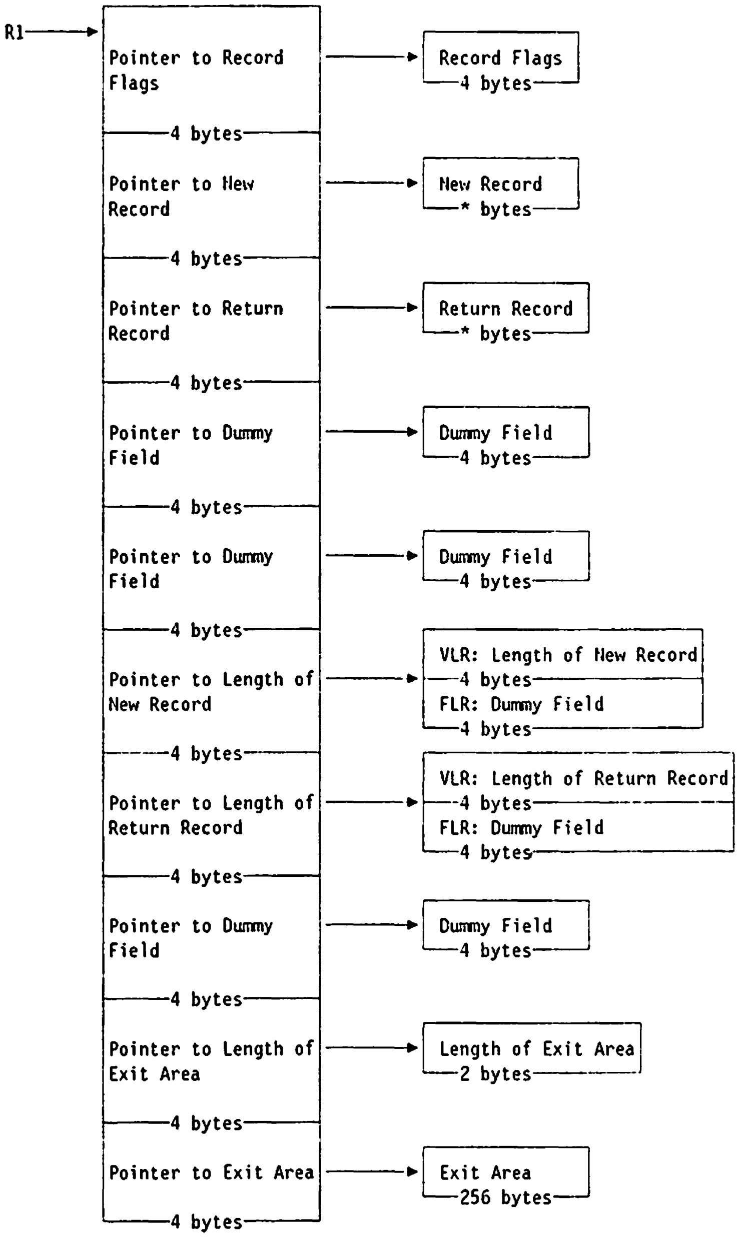
- Record flags
- New record
- Length of the new record (for variable-length records)
- Length of exit area
- Exit area.

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 others can 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 "E35 Linkage Section Fields For Fixed-Length and Variable-Length Records" on page 222.

Figure 31 on page 214 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 31. E15 DFSORT Interface with COBOL

E15 Linkage Examples: Figure 32 is an example of the LINKAGE SECTION code for a fixed-length record (FLR) data set with a logical record length (LRECL) of 100, showing the layout of the fields passed to your COBOL routine.

```
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 32. LINKAGE SECTION Code Example for E15 (Fixed-Length Records)

Figure 33 is an example of the LINKAGE SECTION code for a variable-length record (VLR) data set with a maximum LRECL of 200, showing the layout of the fields passed to your COBOL routine.

Note:

- If the data used for input was not created by a COBOL run, you need to know the LRECL defined for your data set. For a VLR, 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 record descriptor word (RDW) as part of the record. (Each VLR begins with an RDW field of 4 bytes. The RDW is not included in the record passed to your COBOL exit.)
- You need to code only 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 33. LINKAGE SECTION Code Example for E15 (Variable-Length Record)

E15 Linkage Section Fields for Fixed-Length and Variable-Length Records

The fields in the LINKAGE SECTION are used by DFSORT and your routine as stated below. For clarity, the field names from Figure 33 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 VLR does not contain an RDW, but DFSORT places the length of this VLR 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 VLR 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 must not include the 4 bytes for the RDW.
- Each time DFSORT calls your COBOL E15 or COBOL E35 exit, it passes the exit a 256-byte exit area field (EXITAREA). 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 are passed back both to your COBOL E15 exit and your COBOL E35 exit.

  Note:

  - Do not set the exit area length field to more than 256 bytes.
  - You need to code only up to the last field that your routine actually uses (for example, up to RETURN-REC if you do not use the exit area).

E15 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 re-entered during the DFSORT application. *Unless you are inserting records after the end of the 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:

- Move the insert record to the return record field.
- For VLR, 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 re-enters 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 the end of the 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:

- 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:

- 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.

See "E15/E35 Return Codes and EXITCK" on page 226 for complete details of the meanings of return codes in various situations.

E15 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 32 on page 215, 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 33 on page 215, you would code:

  ```
  PROCEDURE DIVISION USING RECORD-FLAGS, NEW-REC,
            RETURN-REC, UNUSED1, UNUSED2,
            NEW-REC-LEN, RETURN-REC-LEN,
            UNUSED3, EXITAREA-LEN, EXITAREA.
  ```

COBOL Exit Routines (Output Phase Exit)

COBOL E35 Exit: Changing Records

Note: The ICEMAC installation option EXITCK affects how DFSORT interprets certain return codes from user exits E15 and E35. To avoid ambiguity, this section assumes that the IBM default, EXITCK=STRONG, was selected at your site. For complete details of the meaning of E15 and E35 return codes in various situations with EXITCK=STRONG and EXITCK=WEAK, see "E15/E35 Return Codes and EXITCK" on page 226.

The E35 routine is entered each time DFSORT prepares to place a record in the output area.

See Figure 27 on page 184 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.

Note:

- If both E15 and E35 exits are used, they must be in the same version of COBOL.
- If you use the E35 exit, you can omit the SORTOUT DD statement, but you must include a RECORD statement in the program control statements.
- 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.
- You cannot use dynamic link-editing with a COBOL E35 exit.

E35 Interface with COBOL

Each time your E35 exit is called, DFSORT supplies the following fields:

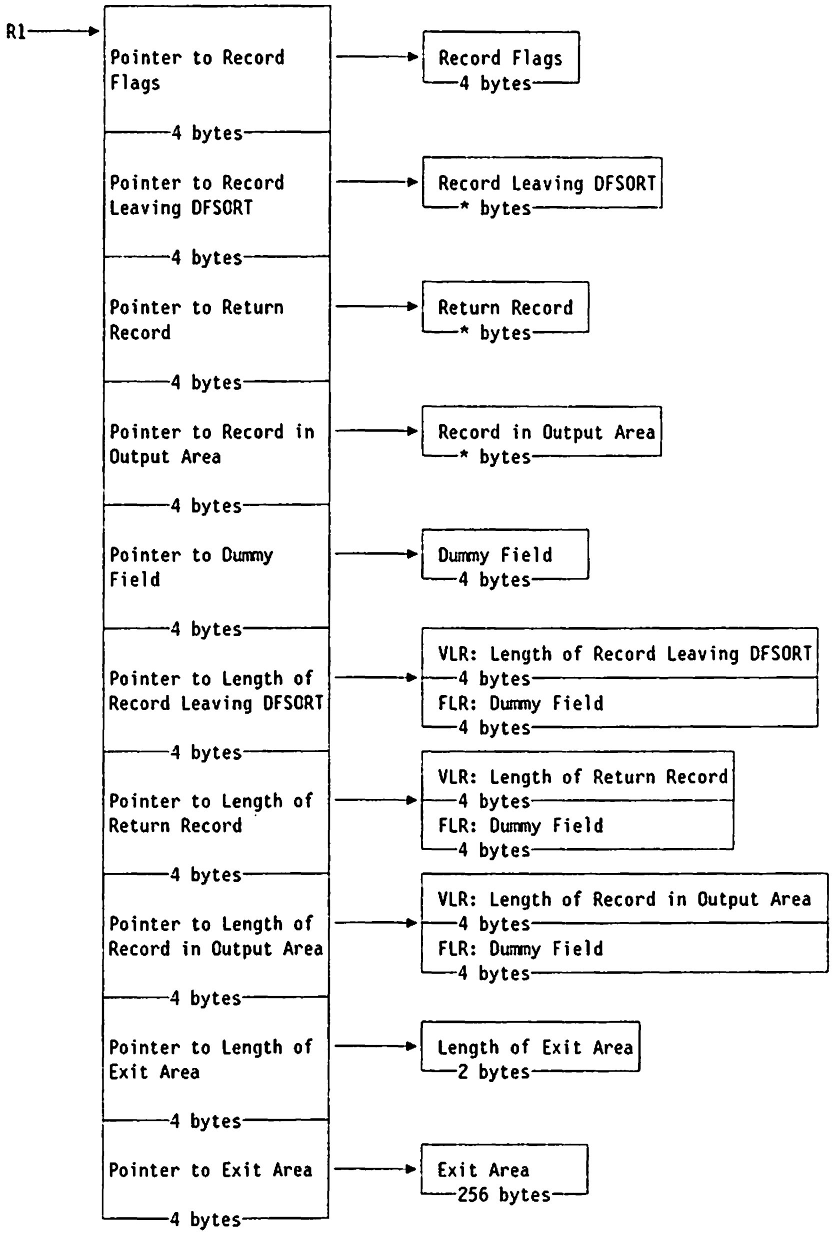
- Record flags
- Record leaving DFSORT
- Length of record leaving DFSORT (for variable-length records)
- Length of exit area
- Exit area.

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 others can 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 variable-length records)
- Length of exit area
- Exit area.

For more information on how these fields are used in a COBOL E35 exit, see "E35 Linkage Section Fields For Fixed-Length and Variable-Length Records" on page 222.

Figure 34 on page 220 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 34. E35 Interface with COBOL

E35 Linkage Section Examples: Figure 35 is an example of the LINKAGE SECTION code for a fixed-length record (FLR) data set with a logical record length (LRECL) of 100, showing the layout of the fields passed to your COBOL routine.

Note: You need to code only 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 35. LINKAGE SECTION Code Example for E35 (Fixed-Length Records)

Figure 36 is an example of the LINKAGE SECTION code for a variable-length record (VLR) data set with a maximum LRECL of 200, showing the layout of the fields passed to your COBOL routine.

Note:

- VLR records have a 4-byte record descriptor word (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.
- You need to code only 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 36. LINKAGE SECTION Code Example for E35 (Variable-Length Records)

E35 Linkage Section Fields For Fixed-Length and Variable-Length Records

The fields in the LINKAGE SECTION are used by DFSORT and your routine as stated below. For clarity, the field names from Figure 36 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 VLR does not contain an RDW; DFSORT places the length of this VLR 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 VLR 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 VLR does not contain an RDW. DFSORT places the length, not including the 4 bytes for RDW, of this VLR in the record in output area length field (OUTPUT-REC-LEN).
- DFSORT passes your COBOL E35 routine a 256-byte exit area field (EXITAREA) that can contain information passed by your COBOL E15 routine. If no information is passed in the EXITAREA field by your COBOL E15 routine the first time the field is passed to your COBOL E35 routine, EXITAREA 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 are 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.

E35 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 re-entered during the DFSORT application. *Unless you are inserting records after the end of the 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:

- 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 re-enters 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 the end of the 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:

- 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:

- 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.

See "E15/E35 Return Codes and EXITCK" on page 226 for complete details of the meanings of return codes in various situations.

E35 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 35 on page 221, 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 36 on page 221, 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

This section provides some sample program exits written in COBOL.

Altering Records: COBOL E15 Exit

Figure 37 is an example of a COBOL E15 routine for 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 37. COBOL E15 Routine Example (FLR)

Inserting Records: COBOL E35 Exit

Figure 38 on page 226 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 38. COBOL E35 Routine Example (VLR)

E15/E35 Return Codes and EXITCK

DFSORT's interpretation of E15 and E35 return codes depends upon whether the ICEMAC option EXITCK=STRONG or EXITCK=WEAK was selected during installation. The following tables show the exact meaning of each E15 and E35 return code with EXITCK=STRONG and EXITCK=WEAK in all possible situations.

Note:

- You can determine whether EXITCK=STRONG or EXITCK=WEAK is in effect from message ICE132I.
- Use of EXITCK=WEAK can make it difficult to detect errors in the logic of your E15 and E35 user exit routines.
- EXITCK=WEAK is treated like EXITCK=STRONG if tape work data sets are specified for a sort application or if the Blockset technique is not selected for a merge application.

Table 10. E15 Without a SORTIN Data Set

E15 Return Code	**Meaning with EXITCK=STRONG**	**Meaning with EXITCK=WEAK**
0	Invalid	Do not return
4	Invalid	Do not return
8	Do not return	Do not return
12	Insert record	Insert record
16	Terminate DFSORT	Terminate DFSORT
20 (COBOL only)	Invalid	Do not return
All others	Invalid	Invalid

Table 11. E15 With a SORTIN Data Set Before End-of-File

E15 Return Code	**Meaning with EXITCK=STRONG and EXITCK=WEAK**
0	No action/record altered
4	Delete record
8	Do not return
12	Insert record
16	Terminate DFSORT
20 (COBOL only)	Alter/replace record
All others	Invalid

Table 12. E15 With a SORTIN Data Set After End-of-File

E15 Return Code	Meaning with EXITCK=STRONG	Meaning with EXITCK=WEAK
0	Invalid	Do not return
4	Invalid	Do not return
8	Do not return	Do not return
12	Insert record	Insert record
16	Terminate DFSORT	Terminate DFSORT
20 (COBOL only)	Invalid	Do not return
All others	Invalid	Invalid

Table 13. E35 With a SORTOUT Data Set Before End-of-Input

E35 Return Code	Meaning with EXITCK=STRONG and EXITCK=WEAK
0	No action/record altered
4	Delete record
8	Do not return
12	Insert record
16	Terminate DFSORT
20 (COBOL only)	Alter/replace record
All others	Invalid

Table 14. E35 Without a SORTOUT Data Set Before End-of-Input

E35 Return Code	Meaning with EXITCK=STRONG	Meaning with EXITCK=WEAK
0	Invalid	Delete record
4	Delete record	Delete record
8	Do not return	Do not return
12	Invalid	Delete record
16	Terminate DFSORT	Terminate DFSORT
20 (COBOL only)	Invalid	Delete record
All others	Invalid	Invalid

Table 15. E35 With a SORTOUT Data Set After End-of-Input

E35 Return Code	Meaning with EXITCK=STRONG	Meaning with EXITCK=WEAK
0	Invalid	Do not return
4	Invalid	Do not return
8	Do not return	Do not return
12	Insert record	Insert record
16	Terminate DFSORT	Terminate DFSORT
20 (COBOL only)	Invalid	Do not return
All others	Invalid	Invalid

Table 16. E35 without a SORTOUT Data Set After End-of-Input

E35 Return Code	Meaning with EXITCK=STRONG	Meaning with EXITCK=WEAK
0	Invalid	Do not return
4	Invalid	Do not return
8	Do not return	Do not return
12	Invalid	Do not return
16	Terminate DFSORT	Terminate DFSORT
20 (COBOL only)	Invalid	Do not return
All others	Invalid	Invalid

Chapter 6. Invoking DFSORT from a Program

DFSORT can be invoked dynamically from programs written in COBOL or PL/I. Specific information on dynamic invocation is covered in the COBOL and PL/I programming guides. JCL requirements are the same as those for assembler.

This section explains what you need to know about DFSORT in order to initiate it from within your assembler program with a system macro instruction, instead of with an EXEC job control statement in the input stream. Specific restrictions on invoking DFSORT from PL/I and COBOL are listed in "Restrictions for Dynamic Invocation" on page 245.

What Are 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 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*.

Using System Macro Instructions

To initiate DFSORT processing with a system macro instruction, you must:

- Write the required job control language (JCL) DD statements.
- Write DFSORT control statements as operands of assembler DC instructions. (Using DFSPARM or SORTCNTL data sets to specify program control statements can be more convenient. See Chapter 2, "Executing DFSORT with Job Control Language" on page 15 for details.)
- Write a parameter list containing information to be passed to DFSORT and a pointer containing the address of the parameter list. DFSORT accepts two types of parameter lists: a 24-bit parameter list, and an extended parameter list. Although you can choose either parameter list, 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 are usually required when invoking DFSORT from another program. The statements and their necessary parameters are described in detail in Chapter 2, "Executing DFSORT with Job Control Language" on page 15.

Program Control Statements for the 24-Bit Parameter List

When using the 24-bit parameter list, you must supply the starting and ending address of a valid image of each control statement to be used during execution. You must provide the image as a character string in EBCDIC format using assembler DC instructions. The rules for preparing the program control statements are:

- 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 60.
- 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 can 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 3, "Using DFSORT Program Control Statements" on page 53, except:
  - Labels are not allowed; a leading blank is optional.
  - Because each control statement image must be defined contiguously by one or more assembler DC instructions, explicit and implicit continuation of statements is neither necessary nor allowed.
- Neither comment statements, blank statements, nor remark fields are permitted.
- If you use ATTACH to initiate the program, you cannot use the checkpoint/restart facility and must not specify CKPT in the SORT statement image.

  For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

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.

Format of the 24-Bit Parameter List

Figure 39 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 C, “Specification/Override of DFSORT Options” on page 353.

Address of pointer

(Hex)	(Dec)	X'80'	Pointer to the beginning of the parameter list

		Byte 1	Byte 2	Bytes 3 and 4	
-2	-2	Unused	Unused	Number of bytes in following list.	Notes
2	2	X'00'	Starting address of SORT or MERGE statement image.		1
6	6	X'00'	Ending address of SORT or MERGE statement image.		1
A	10	X'00'	Starting address of RECORD statement image.		1
E	14	X'00'	Ending address of RECORD statement image.		1
12	18	X'00'	Address of E15 or E32 routine (zeros if none).		1
16	22	X'00'	Address of E35 routine (zeros if none)		1
1A	26	X'02'	Starting address of MODS statement.		2
1E	30	X'00'	Ending address of MODS statement.		2
22	34	X'00'	Optional main storage value (hex).		3
26	38	X'01'	Optional reserved main storage value (hex).		3
2A	42	X'03'	Starting address of message DDNAME.		3
2E	46	X'04'	Number of input files to a merge-only.		3,4
32	50	X'05'	Starting address of DEBUG statement image.		3
36	54	X'00'	Ending address of DEBUG statement image.		3,5
3A	58	X'06'	Starting address of ALTSEQ statement image.		3
3E	62	X'00'	Ending address of ALTSEQ statement image.		3,6
42	66	X'F6'	Pointer to ALTSEQ translation table.		3
46	70	X'F7'	User exit address constant.		3
4A	74	X'FD'	Bytes after X'FD' are ignored.		3
4E	78	X'FE'	Address of a pointer to 104-byte ESTAE work area (or zeros).		3
52	82	X'FF'	Message option (MSGPRT).		3
56	86	Optional 4-character prefix for DDNAME.			3

Figure 39. The 24-Bit Parameter List

Notes to Figure 39 :

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 C, "Specification/Override of DFSORT Options" on page 353).
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.

The specifications for each of the parameter list entries follow:

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. The 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 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 the E15 or E32 routine that your program has placed in main storage, if any; otherwise, all zeros. Must be in the last three bytes. The first byte must contain X'00'.
22-25	The address of the E35 routine that your program has placed in main storage, 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. 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 installation option. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353. For an explanation of this value, see the discussion of the MAINSIZE parameter in "OPTION Control Statement" on page 97.

38-41 A reserved main storage value. Optional. The first byte must contain 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 explanation of the RESINV parameter in "OPTION Control Statement" on page 97.

You can use this option to temporarily override the RESINV installation option. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

42-45 Message ddname. Optional. The DDNAME for the output data set for program messages. You can use this option to temporarily override the MSGDDN installation option. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353. For details on use of the message data set, see *DFSORT Messages and Codes*.

The first byte must contain X'03'. The next 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 without the FILES option and input to the merge is supplied through the E32 exit. 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 C, "Specification/Override of DFSORT Options" on page 353.

50-53 The starting address of the DEBUG control statement image. Must be in the last three bytes. The first byte must contain 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 staring address of the ALTSEQ control statement image. Must be in the last three bytes. The first byte must contain X'06'. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

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. The first byte must contain X'F6'. If this parameter is present, the X'06' parameter is ignored. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

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 it finds a variable-length input record too short to contain all specified control fields. For full details of this option, see the discussion of the VLSHRT parameter in "OPTION Control Statement" on page 97. You can use this option to temporarily override the VLSHRT installation option. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

78-81 If the first byte contains X'FE', you can use the next three bytes to pass an address of a 104-byte field save area where ESTAE information is saved; these bytes must contain zeros if the ESTAE information is not saved.

If a system or user exit abend occurs, the DFSORT ESTAE recovery routine will copy the first 104 bytes of the SDWA into this area before returning to any higher level ESTAE recovery routines.

For more information on the DFSORT ESTAE recovery routine, see Appendix F, "DFSORT Abend Processing" on page 383

82-85 The message option. The first byte must contain X'FF'. The following three bytes contain the characters NOF, (I), or (U). You can use this option to temporarily override the MSGPRT installation option.

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 printed (NOLIST is forced).

All messages are written to the message data set. For details on use of the message data set, see *DFSORT Messages and Codes*. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

For compatibility reasons, the forms (NO, (AP, (AC, (CC, (CP, and (PC are also accepted.

The following table lists the equivalent specifications for these alternate forms.

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 uses statements ABC#IN, ABC#CNTL, ABC#WKnn, ABC#OUT, and ABC#INnn instead of SORTIN, SORTCNTL, SORTWKnn, SORTOUT, and SORTINnn.

Note: This parameter is equivalent to the SORTDD=cccc execution option.

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 in EBCDIC format, using assembler DC instructions and containing valid images of the control statements to be used during execution.

The rules for preparing the program control statements are:

- 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. Because each control statement image must be defined contiguously by one or more assembler DC instructions, explicit and implicit continuation of statements is not necessary or allowed.
- The MODS statement is required when exits other than E15, E18, E32, E35, and E39 are to be used or when the E15, E18, E35, or E39 routine addresses are not passed by the parameter list.
- All of the control statements described in Chapter 3, "Using DFSORT Program Control Statements" on page 53 can be specified; none is required, but SORT, MERGE, or OPTION COPY must be specified in the parameter list, SORTCNTL, or DFSPARM.

- If you use ATTACH to initiate the program, you cannot use the checkpoint/restart facility. Do not specify CKPT on the SORT or OPTION statement.

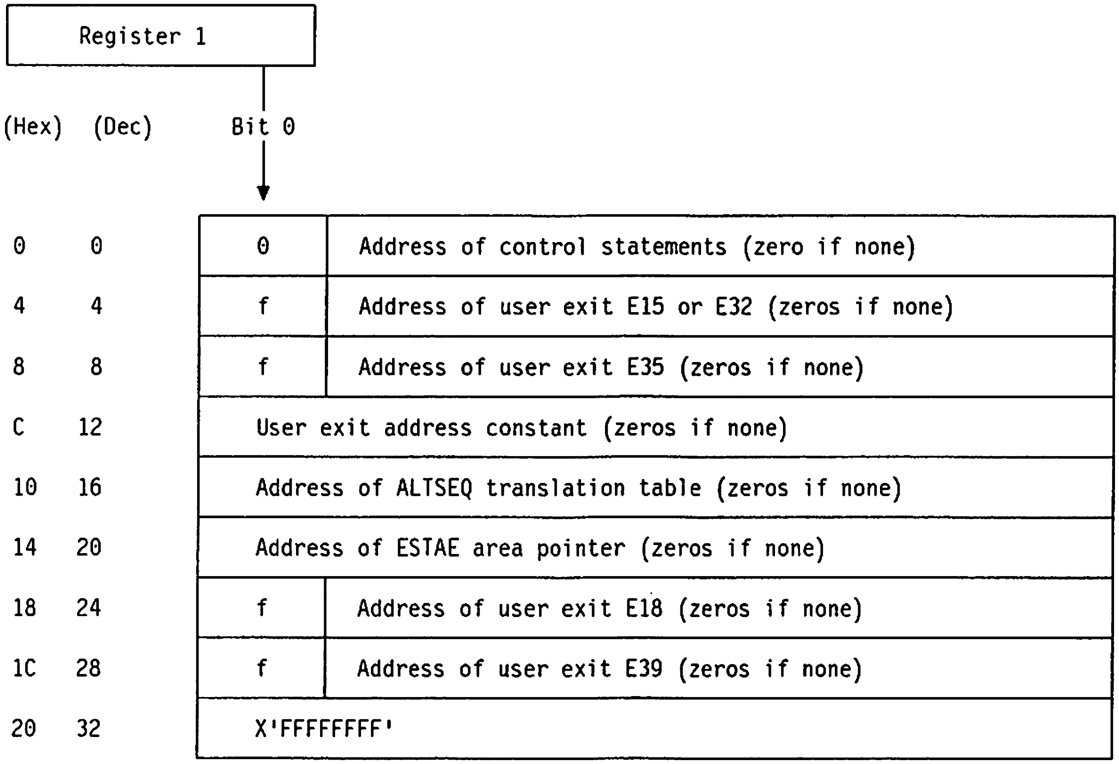
For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

Format of the Extended Parameter List

Figure 40 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. It 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 is treated as if it were specified as zeros. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.



(Hex)	(Dec)		
0	0	0	Address of control statements (zero if none)
4	4	f	Address of user exit E15 or E32 (zeros if none)
8	8	f	Address of user exit E35 (zeros if none)
C	12	User exit address constant (zeros if none)	
10	16	Address of ALTSEQ translation table (zeros if none)	
14	20	Address of ESTAE area pointer (zeros if none)	
18	24	f	Address of user exit E18 (zeros if none)
1C	28	f	Address of user exit E39 (zeros if none)
20	32	X'FFFFFFFF'	

Figure 40. The Extended Parameter List

Detailed specifications for each of the entries in the parameter list follow:

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 you specify this parameter as zeros, you must supply all the required control statements in DFSPARM or SORTCNTL.

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.

f (bit 0) has the following meaning when executing in an MVS/XA or MVS/ESA 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.

f (bit 0) has the following meaning when executing in an MVS/XA or MVS/ESA 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. This field will be passed to the E15 and/or E35 routines.

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. For full override and applicability details, see Appendix C, "Specification/Override of DFSORT Options" on page 353.

20-23 Optional. The address of a 4-byte field containing the address of a 112-byte work area where ESTAE information is saved, or all zeros if the ESTAE information is not saved.

If a system or user-exit abend occurs, the DFSORT recovery routine will copy the first 112 bytes of the SDWA into this area before returning to your ESTAE recovery routine.

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.

Note: This parameter is ignored for a merge application and for a tape work data set sort application.

f (bit 0) has the following meaning when executing in an MVS/XA or MVS/ESA 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.

Note: This parameter is ignored for a conventional merge application and for a tape work data set sort application. This parameter must be followed by X'FFFFFFFF' if you specify it.

f (bit 0) has the following meaning when executing in an MVS/XA or MVS/ESA 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).

Note: The list can be ended after any parameter. The last parameter in the list *must* be followed by X'FFFFFFFF'.

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 sort, merge, and copy jobs.)
- 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 might 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 DFSORT. 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 C, "Specification/Override of DFSORT Options" on page 353.

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 re-enterable (provided that any exit routines you use are also re-enterable).

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 Option (24-Bit Parameter List)

Figure 41 shows the format of the 24-bit parameter list you would use to specify the main storage option for a sort application.

(Hex)	(Dec)	Byte 1	Byte 2	Bytes 3 and 4
-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'	Zeros (no E15 routine provided)	
16	22	X'00'	Zeros (no E35 routine provided)	
1A	26	X'00'	Main storage value (in hexadecimal)	

Figure 41. Specifying the Main Storage Option (24-Bit Parameter List)

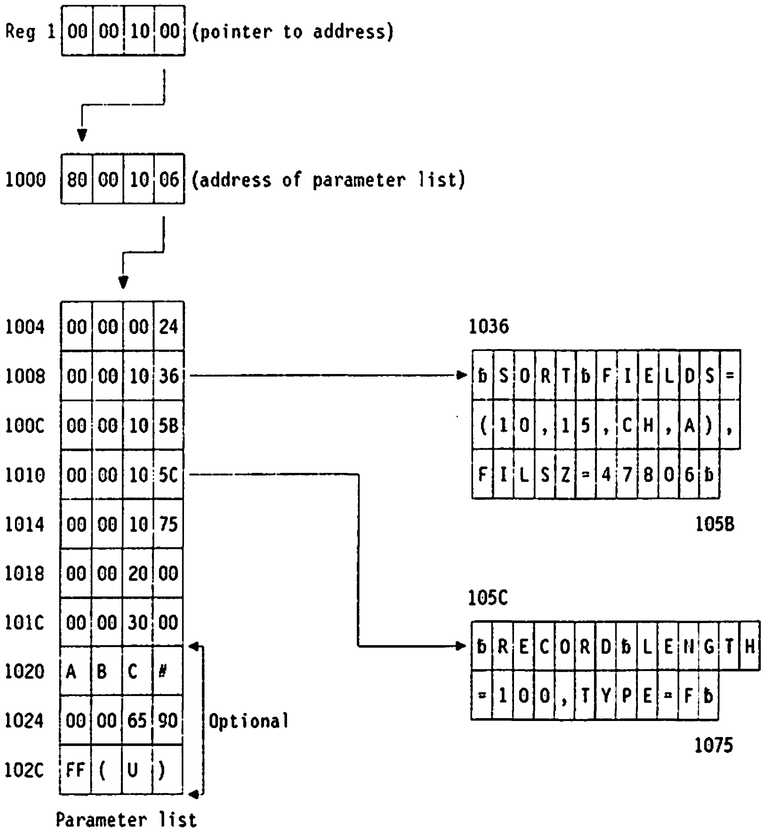
Example 2. Supplying Input through Exit E32 and Giving Control to the ESTAE Routine (24-Bit Parameter List)

Figure 42 on page 243 shows the format of the 24-bit parameter list that you would use for a merge application when you want to supply input through exit E32 and give control to the ESTAE routine if the program fails.

(Hex) (Dec)	Byte 1	Byte 2	Bytes 3 and 4
-2 -2	Unused		X'001C'
2 2	X'00'	Starting address of MERGE statement	
6 6	X'00'	Ending address of HERGE 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 42. Specifying E32 and ESTAE Routine (24-Bit Parameter List)

Example 3. How a 24-Bit Parameter List Might Appear in Main Storage



```
Reg 1  00 00 10 00  (pointer to address)

1000   80 00 10 06  (address of parameter list)

1004   00 00 00 24                1036
1008   00 00 10 36  ------------> b S O R T b F I E L D S =
100C   00 00 10 5B                ( 1 0 , 1 5 , C H , A ) ,
1010   00 00 10 5C                F I L S Z = 4 7 8 0 6 b
1014   00 00 10 75                                   105B
1018   00 00 20 00
101C   00 00 30 00                105C
1020   A  B  C  #                 b R E C O R D b L E N G T H
1024   00 00 65 90   Optional     = 1 0 0 , T Y P E = F b
102C   FF (  U  )                                    1075
       Parameter list
```

Figure 43. The 24-Bit Parameter List in Main Storage

Figure 43 shows how a 24-bit parameter list might appear 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 44 shows, in assembler language, how to code the parameters and statement images needed for the 24-bit parameter list in Figure 43 on page 243, 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 44. Coding a 24-Bit Parameter List

Example 5. Coding an Extended Parameter List

The example in Figure 45 shows, in assembler language, how to use an extended parameter list to code parameters and statement images, and pass control to DFSORT.

```
         .
         .
         .
         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 45. Coding an Extended Parameter List

Restrictions for Dynamic Invocation

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 or DFSPARM 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 "COBOL Requirements for Copy Processing" on page 210 for exit requirements.

Chapter 7. Improving Efficiency

DFSORT is designed to optimize performance automatically. It sets optimization variables (such as buffer sizes) and selects the most efficient of several sorting and merging techniques.

You can improve DFSORT performance in several ways:

- Design your jobs to maximize performance:
  - Use JCL to invoke DFSORT processing.
  - Plan ahead when designing new applications.
  - Specify efficient sort/merge techniques.
  - Specify input/output data set characteristics accurately.
  - Use options that enhance performance whenever possible.
  - Use options that decrease performance only when necessary.
- Use main storage efficiently.
- Allocate temporary work space efficiently.
- Use Hipersorting on MVS/ESA systems.
- Use EXCPVR on MVS/XA and MVS/ESA systems.
- Use ICEGENER instead of IEBGENER.

Designing Your Jobs to Maximize Performance

Even though DFSORT automatically optimizes performance when your job is executed, you can still improve efficiency by using specifications and options that permit DFSORT to make the best possible use of available resources.

Use JCL to Invoke DFSORT Processing

You can enhance performance by invoking DFSORT with JCL instead of invoking it from a COBOL or a PL/I program. Generally, COBOL or PL/I is used for convenience. However, the trade-off can be degraded performance. You can improve efficiency by taking advantage of the way DFSORT installation defaults and run-time options can be fine-tuned for optimum performance, especially to make use of control statements that "work together," such as INCLUDE/OMIT, INREC/OUTREC, and SUM. You can eliminate records from input files, reformat records to eliminate unwanted fields, and combine records arithmetically, without requiring routines from other programs.

Plan Ahead

You should consider several factors when designing new applications. Some of these factors are discussed below.

Whenever possible:

- Use either EBCDIC character or binary control fields.
- Place binary control fields so they start and end on byte boundaries.
- Avoid using the alternative collating sequence character translation.
- Specify fixed-point, packed decimal. and zoned decimal control fields (if you know they will always be positive) so that they can be sorted or merged as if they were binary control fields.
- Use packed decimal format rather than zoned decimal.

- If several contiguous character or binary control fields in the correct order of significance are to be sorted or merged in the same order (ascending or descending), specify them as one control field.
- Avoid overlapping control fields.

Efficient Blocking

Performance of DFSORT can be significantly improved if you block your input and output records. A block size of 6000 or more bytes is generally considered to be a good value for DASD data sets. For large files, files on tape, or files that you sort often, you might want to choose a larger block size. In general, the larger your input and output block sizes, the better DFSORT's performance.

Specify 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, DFSORT issues error message ICE160A and stops processing.

Sorting Techniques

One condition that affects which sorting technique DFSORT selects is the type of device used for intermediate storage. If you use a tape device, the Conventional technique is used, which is less efficient. For more information on using tape devices for intermediate storage, see "Tape Work Storage Devices" on page 261.

The Blockset and Peerage/Vale techniques can be used only with disk devices. These techniques are discussed below.

Blockset Disk Sorting Techniques: DFSORT's most efficient techniques, FLR-Blockset (for fixed-length records) and VLR-Blockset (for variable-length records), will be used for most sorting applications.

Notes:

- The Blockset technique might require more intermediate work space than Peerage/Vale. For more information, see "Allocating Temporary Work Space Efficiently" on page 259.
- 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.

Peerage/Vale Disk Sorting Techniques: When the conditions for use of the Blockset sorting technique are not met, DFSORT uses Peerage/Vale.

Merging Techniques

For merging applications, DFSORT uses the Blockset and Conventional techniques.

Blockset Merging Techniques: DFSORT's most efficient techniques, FLR-Blockset (for fixed-length records) and VLR-Blockset (for variable-length records), will be used for most merging applications.

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 Merge Technique: When 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 merge technique, which is less efficient.

Specify Input/Output Data Set Characteristics Accurately

DFSORT uses the information given it (about the operation it is to perform) to optimize for highest efficiency. When you supply incorrect information or 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.

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 work storage. Therefore, if the exact number of records to be sorted is known, use that number as the value for the FILSZ run-time option. If an exact FILSZ is not specified, DFSORT will estimate the number of records.

If you are doing a tape work data set sort, the most important information you can give DFSORT is an accurate data set size in the FILSZ execution option.

Variable-Length Records

When the input data set consists of variable-length records and dynamic allocation of intermediate data sets is used, specify the average record length as accurately as possible in the RECORD statement.

Direct Access

System performance is improved if storage is specified in cylinders rather than tracks. Storage on sort work data sets will be reallocated in cylinders. The number of tracks per cylinder for direct-access devices is shown in Table 17.

Table 17. Number of Tracks per Cylinder for Direct-Access Devices

Device	Tracks per Cylinder
2314/2319	20
3330/3333	19
3340/3344	12
3350	30
3375	12
3380	15

If WRKSEC is in effect and the work data set is not allocated to virtual I/O, DFSORT allocates secondary extents as required, even if not requested in the JCL.

Allocating twice the space used by the input data set(s) is usually adequate for the work data sets. Certain conditions can cause additional space requirements. These include:

- Long control words (more than 150 bytes)
- Using different device types or work data sets
- Using an alternative collating sequence.

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.

Tape

Three different techniques are available to the program: Balanced, Polyphase, and Oscillating. For information on how to calculate their requirements, see Table 18.

Table 18. External Work Storage Requirements of the Various Tape Techniques

Tape Technique	Maximum Input	Work Storage Areas Required	Max. No. of Work Area	Comments
Balanced tape BALN	15 volumes	Min=2(V+1) tape units	32 volumes	Used if more than three work storage tapes are provided and file size is not given.
Polyphase tape POLY	1 volume	Min=3 tape units	17 volumes	Used if three work storage tapes are provided.
Oscillating tape OSCL	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.

Key to Table 18:

V = Number of input volumes.

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 it is spanned, you will need more work storage. DFSORT selects the most appropriate tape technique using these criteria.

Use Options that Enhance Performance

To obtain optimum performance, you can fine-tune the options specified during installation and at run time. Several options that can enhance performance are described below.

COBEXIT

To take advantage of the COBOL II interface with DFSORT, and enhance performance, specify COBEXIT=COB2 on the OPTION control statement or define it as the installation default when you run exits compiled with VS COBOL II.

EXCPVR

To improve Blockset sorting performance on MVS/XA and MVS/ESA by using EXCPVR, specify EXCPVR=ALL. Remember that using EXCPVR reduces CPU time at the expense of paging other jobs in and out. You can compromise between reducing CPU time and increasing the paging activity for other jobs by specifying EXCPVR=NOWRK. If the higher paging rates are a prime concern, specify EXCPVR=NONE.

Note: The EXCPVR option is not supported for MVS/370. See "EXCPVR" on page 261 for details.

FASTSRT

By specifying the VS COBOL II FASTSRT compile-time option, you can significantly reduce DFSORT processor time, EXCPs, and elapsed time. With FASTSRT, DFSORT input/output operations are more efficient because DFSORT rather than COBOL does the input and output (see Figure 46 on page 253). For more details, see the VS COBOL II publications.

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 or OMIT, STOPAFT, SKIPREC, and SUM functions. You might be able to eliminate your input and output procedures by coding the appropriate DFSORT program control statements and placing them in either the DFSPARM (DFSORT), SORTCNTL (DFSORT), or IGZSRTCD (COBOL) data set, thereby allowing your SORT statement to qualify for FASTSRT.

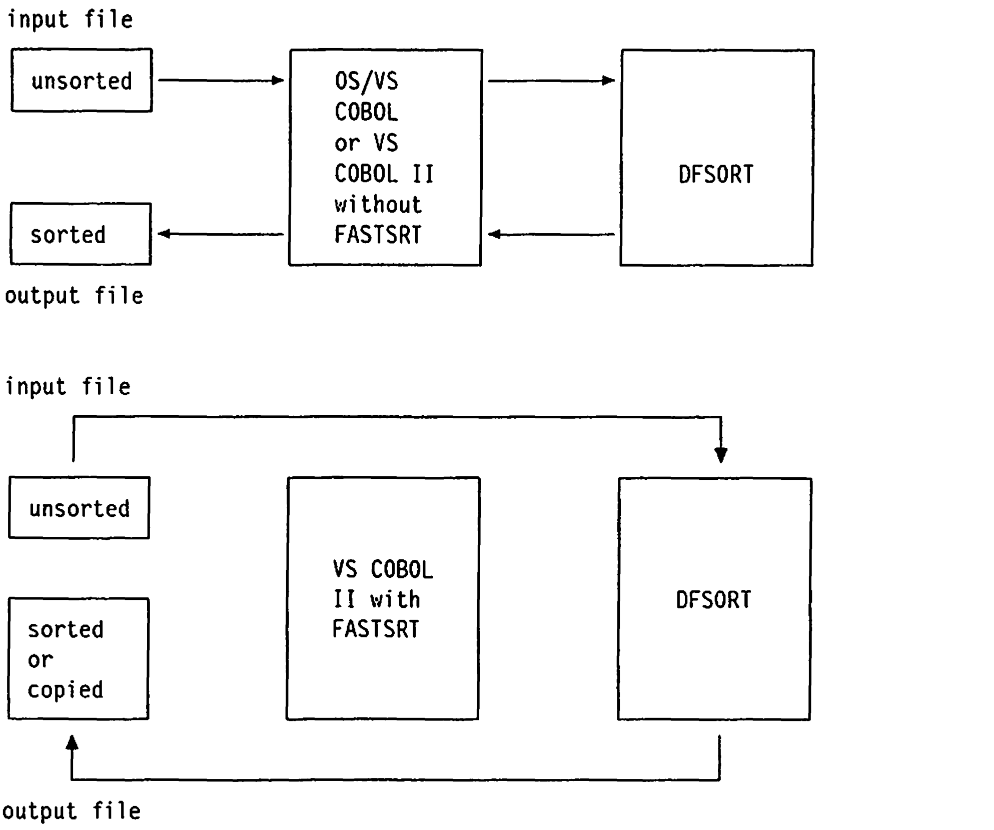


Figure 46. Faster Sorting with VS COBOL II

HIPRMAX

Blockset sorting performance can be improved by using hiperspace along with DASD for temporary storage on an MVS/ESA system.

The HIPRMAX parameter sets the maximum amount of hiperspace to be committed during a run. Specifying HIPRMAX=OPTIMAL permits DFSORT to optimize the maximum amount of hiperspace allocated for a run dynamically, based on system activity at the time the job begins. See "Hipersorting" on page 261 for more information.

INCLUDE or OMIT, STOPAFT, and SKIPREC

You can use either the INCLUDE or OMIT statement and the STOPAFT and SKIPREC options to reduce the size of the input file, which can 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 to be accepted for sorting or copying.
- The SKIPREC option allows you to skip records at the beginning of the input file and sort or copy only the remaining records.

INREC and OUTREC

You can use the INREC statement to reformat the input records *before* processing. This can increase efficiency if you reduce the size of the records, or reduce efficiency if you increase the size of the records.

You can use OUTREC 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 OUTREC are used with INCLUDE or OMIT, DFSORT first processes the INCLUDE and OMIT statements, thus reducing the number of records for INREC and OUTREC to format.

You must be aware of:

- The change in record size and layout of the resulting reformatted input/output records
- Which DFSORT functions must refer to the layout of the reformatted input records and which functions must refer to the layout of the original input records.

Three types of fields can be removed by INREC/OUTREC:

- Padding fields (blanks, constants, or binary zeros) can be removed before processing by using INREC, and then reinserted after processing by using OUTREC.
- Fields not needed in the output records can be removed before processing by using INREC.
- If a variable-length file is being processed and only the fixed part of the record is needed, the variable part can be eliminated before processing. This allows DFSORT to manage the records internally more efficiently.

SUM

You can improve performance by using SUM to add the contents of fields. The SUM statement adds the contents of its control fields. The result is placed in one record while the other record is deleted, thus reducing the number of records to be sorted or merged by DFSORT. SUM is processed after SORT or MERGE, INCLUDE or OMIT, and/or INREC.

As an installation option, you can specify the printing of positive zoned decimal fields that result from summing. That is, you can tell DFSORT whether to change the last digit of the zone from hex C to hex F. For the correct syntax of this specification, see the ZDPRINT option on the ICEMAC macro in *DFSORT Installation and Customization*.

Eliminating Duplicate Keys: You can eliminate duplicate keys by specifying

```
SUM FIELDS=NONE
```

when using the SUM control statement.

For a diagram of the processing sequence for record handling statements, exits, and options, see Figure 2 on page 6.

Avoid Options that Decrease Performance

Certain options can adversely affect performance, and should be used only when necessary. For example, the CKPT option, which activates checkpoint/restart, prevents use of the efficient Blockset techniques.

CKPT

The CKPT option might 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 selects the Blockset technique.

EQUALS

The EQUALS option causes an additional field of 4 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.

EQUCOUNT

The EQUCOUNT option takes additional time to count the number of records with equal keys.

NOCINV

The NOCINV option precludes the use of control interval access for more efficient VSAM processing.

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.

VERIFY

The VERIFY option degrades performance, because it involves extra processing.

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 application 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-editing of user exit routines degrades performance.

EFS Programs

When EFS programs are included in an application, the time required to run the application might increase.

Using Main Storage Efficiently

In general, the more virtual storage you make available to DFSORT (up to a certain limit), the better the performance. However, your virtual storage allocation must not exceed the amount of real storage generally available for one initiator; otherwise, excessive paging might occur.

DFSORT requires a minimum of 88K bytes, but to get better performance, use a much larger amount of storage. Recommended amounts are about 1 megabyte for MVS/370 or 4 megabytes for MVS/XA and MVS/ESA. Improved performance will be most noticeable with large input files.

Note: Under MVS/XA and MVS/ESA when using the Blockset technique for a sort application, DFSORT can place selected buffers above 16-megabyte virtual. This makes more storage available to DFSORT without having to increase the region size in the job control language. A region size of at least 440K bytes must be available to allow DFSORT to use storage effectively.

Tuning Main Storage

Either the REGION value or the MAINSIZE/SIZE value can determine how much storage is available to DFSORT. See *DFSORT Installation and Customization* for 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 the TMAXLIM and/or MAXLIM installation options have been set excessively high (or low). Guidelines for setting these values are given in *DFSORT Installation and Customization.*

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 give n a value 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 or MVS/ESA) are available to DFSORT.

Although DFSORT requires a minimum of 88K bytes (below 16-megabyte virtual for MVS/XA or MVS/ESA), the minimum amount of main storage required depends on the application.

For best performance, it is strongly recommended that you use significantly more than the minimum amount of main storage.

You will generally 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).
- V◌ry large blocks or logical records
- VSAM data sets (for more information, see your VSAM manuals)
- An Extended Function Support program
- An ICETEXIT routine
- A large ICEIEXIT routine.

Notes:

- In some cases, this release of DFSORT might use more storage than prior releases for comparable jobs. This might affect the operation of some jobs. For example, some jobs that run as in-storage sorts (with no SORTWKnn data sets) in previous releases might not run in-storage when using this release. The amount of storage allocated is normally controlled by TMAXLIM. A REGION size of at least 440K bytes must 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.
- For extremely large sorts (for example, 500 megabytes or more of data), use 16 megabytes or more of main storage for best performance.

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 site'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) can give you some relief from the time constraint. If you are concerned with processor time, then decreasing storage (using REGION or SIZE/MAINSIZE=n) can reduce the processor time associated with sorting small files.

In general, when you vary the amount of storage available to DFSORT, several things occur:

1. If you increase the amount of storage:
   - EXCPs are reduced.
   - For larger files, processor time generally decreases; that is, overhead in managing the extra storage is offset by DFSORT having to make fewer passes over the data.

  - For a very heavily loaded system, elapsed time might increase because DFSORT can be swapped out more often.
  - For very small sorts, processor time might remain stable or increase because of the overhead in managing the extra storage. For larger files, processor time will usually 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.
   - Processor time decreases for very small files, but increases for larger 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.

Releasing Main Storage

Under some circumstances, DFSORT uses all the available storage in your REGION. For MVS/XA and MVS/ESA, 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 is in effect and n is greater than the REGION parameter or the default REGION value for your sort job, or when SIZE/MAINSIZE=MAX and MAXLIM (or TMAXLIM for MVS/XA and MVS/ESA 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 be modified only 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 many of your DFSORT applications pass the exit address in the parameter list and SIZE/MAINSIZE=n is in effect, then 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.

Allocating Temporary Work Space 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, 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 the sort uses for a particular file size when secondary allocation is allowed. For large data sets, performance can be improved by the use of larger values for primary and secondary allocation. 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 during installation. However, even if you do not allow for secondary allocation and you allocate fewer cylinders than indicated in the table, the sorting technique might 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. You can get the best performance using direct-access intermediate storage devices when you:

- Select the storage device with the fastest data transfer rate.
- Assign one work data set per actuator.
- Use devices that are of the same type.
- Use two channel paths to devices.
- Make all data sets the same size, or as nearly the same size as possible.
- Assign SORTIN, SORTOUT, and SORTWKnn on different spindles and separate channels.
- Specify contiguous space for work data sets, and make sure there is enough primary space so that the automatic secondary allocation is not needed.

Elapsed time is decreased when DFSORT can both 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 A, "Calculating Storage Requirements" on page 311.

Tape Work Storage Devices

The use of tape work storage devices prevents the use of the more efficient Blockset technique. 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 reduces the block size used for intermediate storage; this can 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 might find deterioration) if you use more than four tape work units. Therefore, apply the general rule of using as many tapes as possible only when DFSORT has more than 100K bytes available.

For information on how to determine storage requirements when using different tape techniques, see Appendix A, "Calculating Storage Requirements" on page 311.

Note: The frequency with which the tape direction changes during DFSORT work file operations has 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 can vary widely.

Hipersorting

Hipersorting is a new DFSORT capability which uses hiperspace available with MVS/ESA on Enterprise Systems Architecture/370. A hiperspace is a high-performance data space which resides in expanded storage, and is backed by auxiliary storage when necessary. With Hipersorting, hiperspace is used in place of and along with DASD for temporary storage of records during a Blockset sort. Hipersorting reduces I/O processing which in turn reduces elapsed time, EXCPs, and channel usage.

The maximum amount of hiperspace for Hipersorting can be controlled with the HIPRMAX option passed to DFSORT at either installation or run time. HIPRMAX can direct DFSORT to dynamically determine the maximum amount of hiperspace that can be used without affecting system paging or other facilities that use expanded storage. HIPRMAX can also be used to suppress Hipersorting when CPU time optimization is your major concern, since Hipersorting may result in a small CPU time degradation.

EXCPVR

To improve Blockset sorting and copying on MVS/XA and MVS/ESA, DFSORT can use EXCPVR when reading and writing SORTIN, SORTOUT, and SORTWKnn data sets.

Three options for EXCPVR (ALL, NOWRK, and NONE) specify the data sets for which DFSORT can use EXCPVR (when Hipersorting is requested, EXCPVR will not be used for SORTWKnn data sets).

- Specifying EXCPVR=ALL reduces CPU time requirements the most, but can cause other jobs to be paged in and out excessively.
- Specifying EXCPVR=NOWRK prevents DFSORT from using EXCPVR for SORTWKnn data sets, and therefore compromises between reducing CPU time and increasing system paging activity.
- Specifying EXCPVR=NONE prevents DFSORT from using EXCPVR for any data sets. No reduction in CPU time results, but no increases in system paging activity occur.

ICEGENER Facility

You can achieve more efficient processing for jobs set up to use the IEBGENER system utility by using DFSORT's ICEGENER facility. Qualifying IEBGENER jobs are processed by the equivalent, but more efficient, DFSORT copy function. If, for any reason, the DFSORT copy function cannot be used (for example, if IEBGENER control statements are specified), control is automatically transferred to the IEBGENER system utility.

If your site has installed ICEGENER to be invoked by the name IEBGENER, you need not make any changes to your jobs to use ICEGENER. If your site has not chosen automatic use of ICEGENER, you can use ICEGENER by substituting the name ICEGENER for IEBGENER on the EXEC statement (when DFSORT is JCL-invoked) or LINK macro (when DFSORT is program-invoked) in any jobs you choose. Program-invoked jobs must be recompiled.

Following is an example of how an IEBGENER job can be changed to use ICEGENER by substituting the name ICEGENER for the name IEBGENER in the EXEC statement.

```
//GENER JOB...
// EXEC PGM=ICEGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=CONTROL.MASTER,DISP=OLD,UNIT=3380,VOL=SER=MASTER
//SYSUT2 DD DSN=CONTROL.BACKUP,DISP=OLD,UNIT=3380,VOL=SER=BACKUP
//SYSIN DD DUMMY
```

The IEBGENER DD statements SYSUT1 (input), SYSUT2 (output), and SYSPRINT (messages) are used by DFSORT for SORTIN, SORTOUT, and SYSOUT, respectively. These DD statement names will be translated by using an extended parameter list to invoke the copy function. If DFSORT cannot be used (for example, because IEBGENER control statements are specified), control will be transferred to IEBGENER.

Notes:

- Whether ICEGENER is JCL- or program-invoked, DFSORT will be program-invoked using an extended parameter list. Therefore, the ICEMAC INV options apply and SORTCNTL can be used to provide additional control statements for the copy application; for example, INCLUDE or OPTION.

- For most error conditions that prevent the use of DFSORT copy, control will be transferred to the IEBGENER system utility (using its new name, if appropriate). DFSORT messages will not be printed unless a SORTDIAG DD statement is supplied. Use of the SORTDIAG DD statement will allow you to determine why DFSORT copy could not be used.
- If DFSORT copy is used, its operation and messages will be equivalent to a directly called DFSORT copy application. If an unrecoverable error is encountered (for example, an I/O error), DFSORT's return code of 16 will be changed by ICEGENER to a return code of 12 to emulate the return code from a failing IEBGENER application.
- DFSORT copy can perform some functions not provided by IEBGENER, such as certain padding and truncation operations.

Chapter 8. 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 DFSPARM	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
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

No.	Description	Input	Output
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
26	Use the ICEGENER facility	Blocked fixed- or variable-length records	Blocked fixed- or variable-length records.

Sort Examples

Example 1. DISK SORT

INPUT	Fixed or variable-length records on 3380.
OUTPUT	Fixed or variable-length records on 3400.
INTERMEDIATE STORAGE	Two SYSDA areas of 10 cylinders each.

```
//EXAMP   JOB  A402,PROGRAMMER                               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
```

Line **Explanation**

01 The JOB statement introduces this job to the operating system.

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 in ascending order.

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	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
   RECORD  TYPE=F,LENGTH=(,100,80)                            16
   MODS    E15=(MODREC,784,MODLIB),E16=(E16,1024,MODLIB),     17
           E35=(ADDUP,912,MODLIB),E61=(CHGE,1000,MODLIB)      18
```

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 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.

17-18 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.
OUTPUT	Fixed-length blocked records on a 3380.
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 in 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.
OUTPUT	Variable-length records on 3400.
INTERMEDIATE STORAGE	Two 3380 areas of 15 cylinders each.
USER ROUTINES	E11 routine performs initialization for the E16 Nmax routine.
OPTIONS	Alternate collating sequence.

```
//EXAMP     JOB   B999,PROGRAMMER
//STEPN     EXEC  SORTD                                    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
   RECORD   TYPE=V,LENGTH=(120,,,80,120)                   13
   MODS     E11=(PREPMOD,504,USERLIB),E16=(MODMAX,554,     14
            USERLIB)                                       15
   ALTSEQ   CODE=(5BEA,7BEB,7CEC)                          16
```

Line	Explanation
01	Calls the SORTD cataloged procedure.
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 in ascending sequence.

13 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.

14-15 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.

16 ALTSEQ statement. Specifies that the three characters $, #, and @ are to collate in that order after Z.

Example 5. COPY USING DFSPARM

INPUT	Variable-length records
OUTPUT	Variable-length records
INTERMEDIATE STORAGE	None
USER ROUTINES	None
OPTION	COPY,STOPAFT,MSGPRT,ABEND

```
//EXAMP     JOB   B999,PROGRAMMER                           01
//S1 EXEC   PGM=SORT                                        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
//DFSPARM   DD    *                                         07
   OPTION   STOPAFT=500,COPY,MSGPRT=CRITICAL                08
   DEBUG    ABEND                                           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.

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 DFSPARM DD statement indicates that data follows in the input stream.

08 THE OPTION statement indicates that a copy is to be done, that only the first 500 records are to be copied, and that only error messages are to be printed.

09 The DEBUG statement indicates that if this application fails, an abend occurs with a user completion code that is equal to the appropriate message number.

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.

```
//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
   RECORD   TYPE=D,LENGTH=(,,,20,23)                              12
```

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	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.

```
//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
   RECORD   TYPE=D,LENGTH=(,,,20,80)                             09
```

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 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.

```
//EXAMP     JOB   A402,PROGRAMMER
//STEP1     EXEC  PGM=SORT                                  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
```

Line **Explanation**

01 This EXEC statement calls the program module by its alias, SORT.

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 in 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 in descending order.

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.

```
//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
   RECORD    TYPE=F,LENGTH=(120,100,80)                            15
   MODS      E15=(MODREC,784,MODLIB),                              16
      E16=(E16,1024,MODLIB),E35=(ADDUP,912,MODLIB)                 17
                                                                   18
```

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 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.

16-18 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).

```
//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
   RECORD  TYPE=V,LENGTH=(,,,80,120)                        10
   ALTSEQ  CODE=(5BEA,7BEB,7CEC)                            11
   MODS    E11=(PREPMOD,504,USERLIB),E16=(MODMAX,554,       12
           USERLIB)                                         13
```

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 in ascending sequence.

10 RECORD statement. Indicates that the input data set contains variable-length records with a minimum record length of 80 bytes, and an average length of 120 bytes. The RECORD statement is not required for this example, but without it, the program 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.

11 ALTSEQ statement. Specifies that the three characters $, #, and @ are to collate in that order after Z.

12-13 MODS statement. Describes two user routines. The first, PREPMOD, 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 in 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.

```
//EXAMP    JOB   A400,PROGRAMMER
//STEPT    EXEC  PGM=ICEMAN                                      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  DYNALLOC=(3350,2)                                     17
```

Example 12 differs from Example 7 in two ways: The input is a concatenation of three input data sets on unlike devices, and work storage is dynamically allocated on tape.

Line **Explanation**

01 The EXEC statement calls the program module by its name, ICEMAN.

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 in descending order.

17 OPTION statement. The DYNALLOC operand indicates that two work data sets are to be dynamically allocated on 3350.

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
* USE BSAM FOR I/O                                             12
  DEBUG BSAM                                                   13
/*                                                             14
//SORTDIAG DD DUMMY                                            15
```

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 Comment statement. This statement is printed, but otherwise ignored.

13 DEBUG statement. BSAM specifies that the program is to use BSAM to process SORTIN and SORTOUT.

14 Marks the end of the SYSIN data set.

15 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 BSAM
```

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	Blocked 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,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. 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 19 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.

```
//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
```

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 in ascending order. The second field begins on byte 28, is 5 bytes long, contains character data, and is to be sorted in descending order. The optional FORMAT operand is used because both fields contain data of the same format.

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**

01 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 in ascending order. The second field begins on byte 20, is 4 bytes long, contains fixed integer data, and is to be merged in 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                                      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
             REUSE)                                               18
```

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).

18 The REUSE statement specifies that the cluster can be opened repeatedly as a reusable cluster. See "VSAM Considerations" on page 10 for the case of a non-empty VSAM SORTOUT data set defined *without* the REUSE parameter.

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                                              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                                          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.

ICEGENER Example

Example 26. Using the ICEGENER Facility

```
//COPY3    EXEC  PGM=ICEGENER                                01
//SYSPRINT DD    SYSOUT=A                                    02
//SYSUT1   DD    DSN=CTL.MASTER,DISP=OLD                     03
//SYSUT2   DD    DSN=CTL.BACKUP,DISP=OLD                     04
//SYSIN    DD    DUMMY                                       05
```

This example shows how to use the ICEGENER facility for an IEBGENER job, if your installation has not selected automatic use of ICEGENER for IEBGENER jobs. The ICEGENER facility will select the more efficient DFSORT copy function for this IEBGENER job.

Line **Explanation**

01 The EXEC statement calls the ICEGENER facility. PGM=IEBGENER has been replaced with PGM=ICEGENER.

02-05 No other changes to the IEBGENER job are required.

Appendix A. Calculating Storage Requirements

This appendix describes the guidelines for allocating main storage to DFSORT.

It also describes storage devices used for intermediate storage and the factors determining the amount of intermediate storage required for a DFSORT run.

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 256. More information on efficient use of storage is explained in Chapter 7, "Improving Efficiency" on page 247.

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
- The amount of main storage assigned to the program
- The amount of hiperspace used by the job. (MVS/ESA only).

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

For best performance, an optimal amount of hiperspace in combination with disk intermediate storage is strongly recommended. See ."Hipersorting" on page 261 for more information on using the HIPRMAX option.

Direct Access Storage Considerations

You can specify a mixture of direct access devices for a given sort application. The types of device available for intermediate storage are:

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

System performance is improved if intermediate 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 Table 19.

Table 19. Number of Tracks per Cylinder for Direct Access Devices

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

If WRKSEC is in effect and the work data set is not allocated to virtual I/O, DFSORT allocates secondary extents as required, even if not requested in the JCL.

Allocating twice the space used by the input data set(s) is usually adequate for the work data sets. 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 Storage Considerations

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 Table 20.

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 Table 20 on page 313. DFSORT selects the most appropriate tape technique using these criteria.

Table 20. External Work Storage Requirements of the Various Tape Techniques

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

* 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 B. Using Extended Function Support (EFS)

Like the user exits described in Chapter 5, "Using Your Own Exit Routines," the DFSORT Extended Function Support (EFS) interface is a means by which you can pass execution control to an EFS program you write yourself. An EFS program is essential if you want to process double-byte character sets (such as Japanese characters) with DFSORT. To process Japanese data types with DFSORT, you can use the IBM Double Byte Character Set Ordering Support Program (DBCS Ordering), Licensed Program 5665-3601, Release 2.0.

Using an EFS program and EFS program exit routines, you can:

- Sort or merge user-defined data types (such as double-byte character sets) with user-defined collating sequences
- Include or omit records based on the user-defined data types
- Provide user-written messages to DFSORT for printing to the message data set
- Examine, alter, or ignore control statements or EXEC PARM options prior to processing by DFSORT.

The EFS program can also perform routine tasks such as opening and initializing data sets, terminating DFSORT, and closing data sets.

You can write your EFS program in any language that uses standard register and linkage conventions, and can:

- Pass a parameter list and a record (if you provide the EFS01 and EFS02 exit routines in the EFS program) in register 1
- Pass a return code in general register 15.

Note: DFSORT does not support EFS programs for Conventional merge or tape work data set sort applications.

VLSHRT is not allowed if EFS interface processing is in effect and an EFS01 exit routine is provided by the EFS program.

The DFSORT target library, ICEMAC, contains a mapping macro called ICEDEFS. ICEDEFS provides a separate Assembler DSECT for the EFS parameter list.

MVS/XA or MVS/ESA Support of the EFS Program

You can design the EFS program to reside and execute above or below 16-megabyte virtual. Residency and addressing mode can be any valid combination of 24-bit, 31-bit, or ANY. If your EFS program is designed to reside and execute below 16-megabyte virtual on an MVS/XA or MVS/ESA system, the EFS program must determine the proper return mode.

How EFS Works

The EFS interface consists of a variable-length parameter list used to communicate between DFSORT and your EFS program. DFSORT activates the EFS program you write at specific points during execution, and communicates information back and forth across the interface as your EFS program executes.

You can activate your EFS program during execution with the EFS=*name* option (*name* is the name of your EFS program):

- As set during DFSORT installation with the ICEMAC macro (see "Installation Defaults" on page 11).
- On the PARM parameter of your EXEC statement when you use job control language to invoke DFSORT (see "Specifying EXEC Statement PARM Options" on page 21).
- On the OPTION program control statement (see "OPTION Control Statement" on page 97).

See Appendix C, "Specification/Override of DFSORT Options" on page 353 for override information. Figure 47 illustrates the role of the EFS interface in linking DFSORT's processing capabi!ities to the EFS program you write.

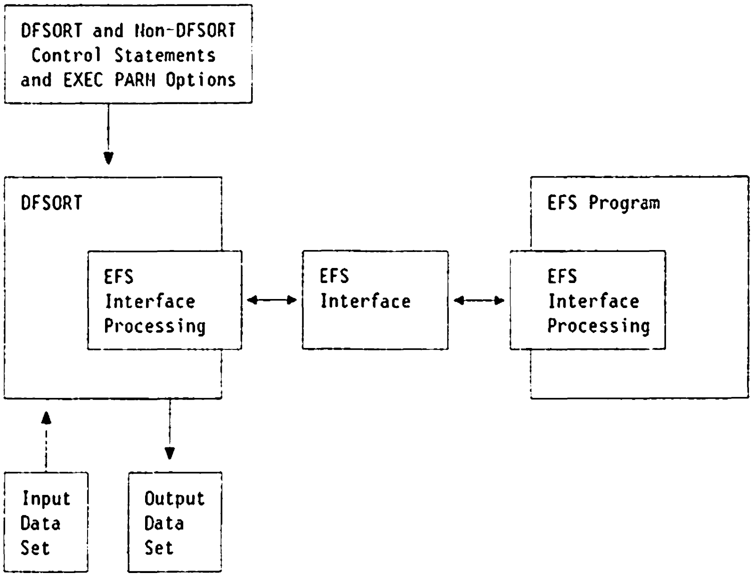


Figure 47. Relationship Between DFSORT and an EFS Program

DFSORT Phases

DFSORT code can be divided into execution phases that consist of components that perform specific tasks, such as distributing input data onto work data sets or writing an output file.

EFS processing can be invoked during the initialization, input, and termination phases of DFSORT. During the initialization phase, DFSORT:

- Processes control statements from SYSIN, SORTCNTL, DFSPARM, or the parameter list in the invoking program

- Processes PARM options from DFSPARM or the EXEC statement
- Opens data sets
- Obtains storage for use during the run.

DFSORT always calls the EFS program during the initialization phase.

During the input phase,·DFSORT reads input records, and performs any INCLUDE or OMIT logic on the records. If the EFS program generates exit routines (EFS01 and EFS02), DFSORT calls them during the input phase.

During the termination phase, DFSORT closes data sets, releases storage, and returns control to the calling program or system. DFSORT always calls the EFS program from the termination phase.

DFSORT Calls to Your EFS Program

DFSORT makes five functional calls (Major Calls 1 through 5) at various phases to transfer information across the EFS interface, between DFSORT and your EFS program. DFSORT can make multiple calls at Major Calls 1 and 2. Refer to Figure 48 on page 318 and Figure 49 on page 319 as you read this section for illustrations of the relationships between program phases and calls during execution.

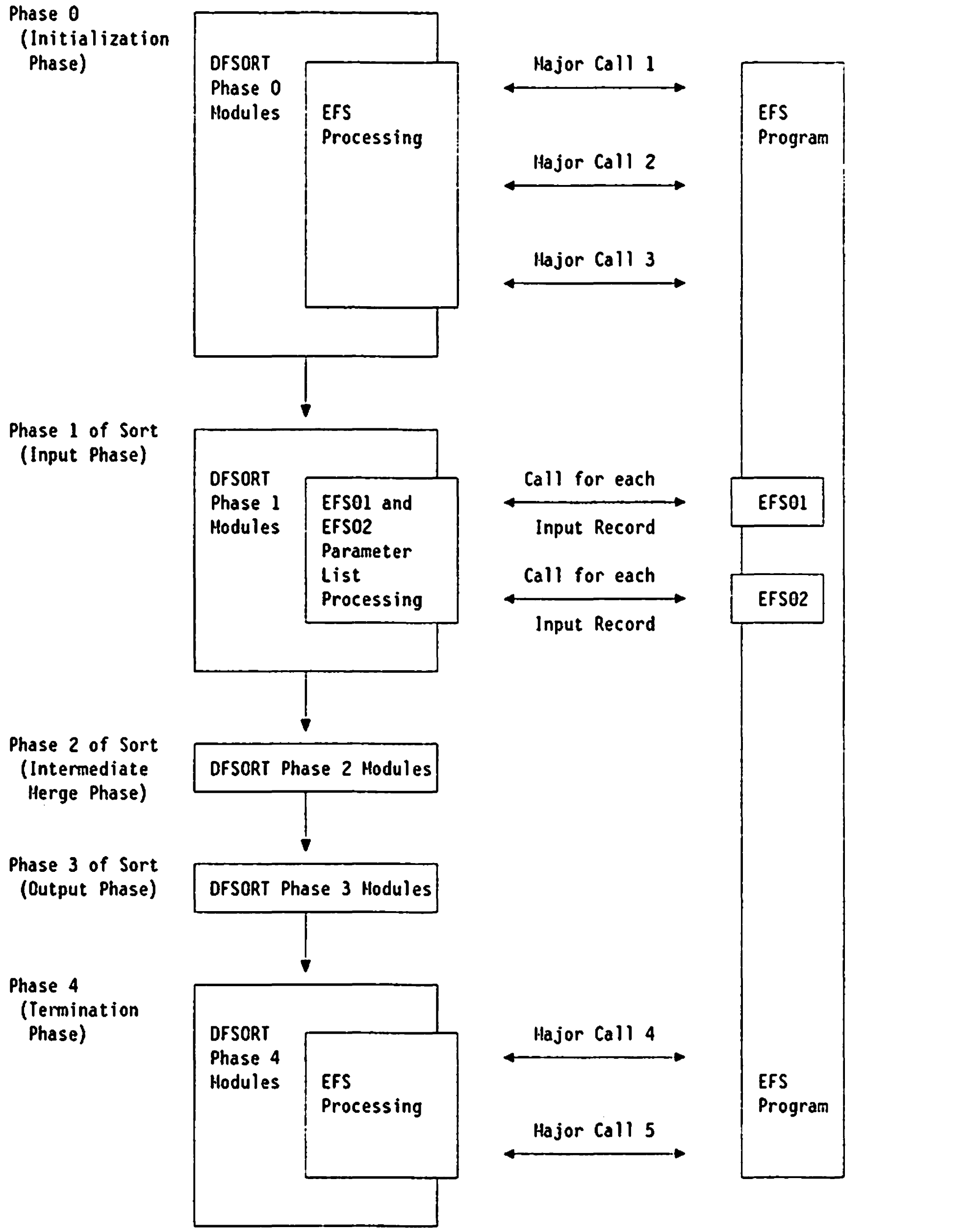


Figure 48. EFS Program Calls for a Sort. The figure also shows the calls to the EFS program EFS01 and EFS02 exit routines.

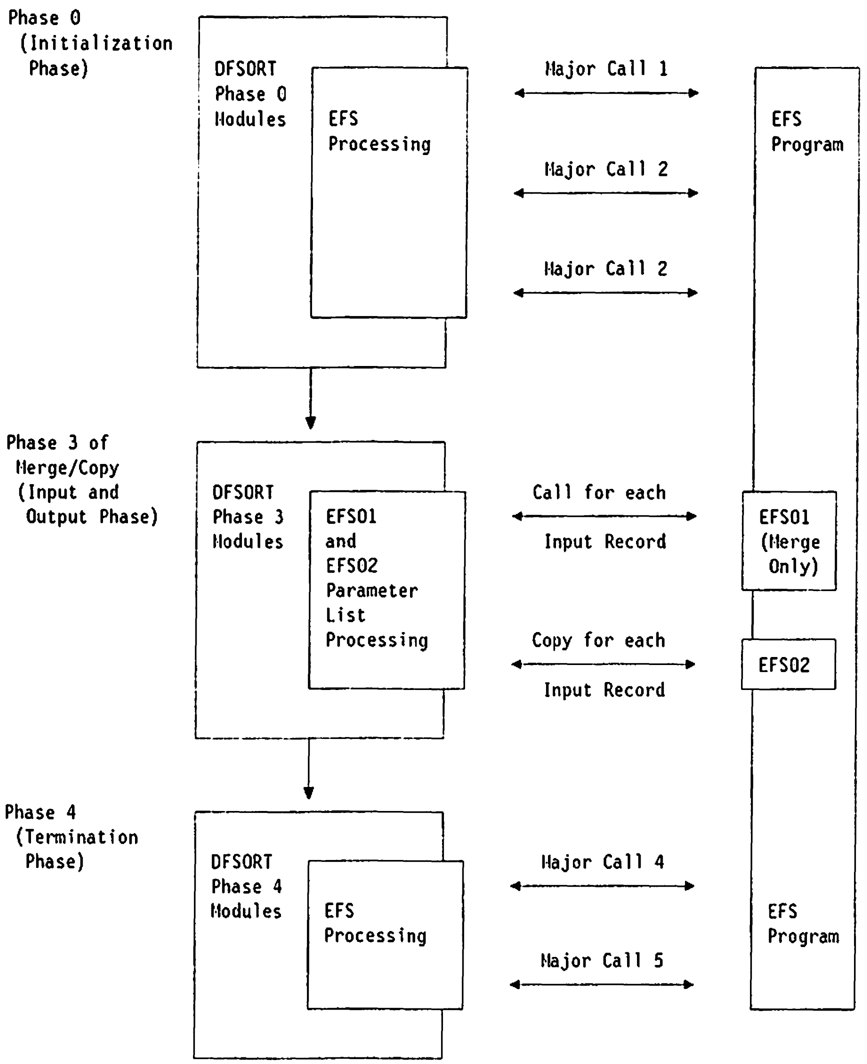


Figure 49. EFS Program Calls for a Merge or Copy. The figure also shows the calls to the EFS program EFS01 and EFS02 exit routines.

Initialization Phase

DFSORT executes Major Calls 1 through 3 during the initialization phase.

Major Call 1: The EFS program can perform initialization processing such as opening data sets and obtaining storage, and can provide user-written messages to DFSORT's message data set. DFSORT obtains information from the EFS program to be used to process the control statements and the two exit routines, EFS01 and EFS02. The EFS01 routine supports sorting or merging user-defined data types with user-defined collating sequences. The EFS02 routine provides logic to include or omit records based on user-defined data types.

Information is passed in both directions between DFSORT and the EFS program across the EFS interface.

At Major Call 1, DFSORT supplies your EFS program with fields in the EFS interface containing:

- An action code indicating that Major Call 1 is in effect
- Informational flags that describe current processing.

When the EFS program returns control to DFSORT, it can supply fields in the EFS interface containing:

- A control statement request list, with a list of DFSORT and non-DFSORT control statement operation definers, or EXEC PARM options needed by the EFS program
- An EFS Program Context area (a private communication area for the EFS program)
- A list containing messages for printing to the message data set
- A return code (in general register 15).

Major Call 2: DFSORT calls your EFS program once for each control statement or EXEC PARM you request. At this call, your EFS program can examine, alter, or ignore control statements before DFSORT processes them, and provide user-written messages to the message data set. DFSORT obtains more information from the EFS program to process the EFS01 and EFS02 exit routines.

At Major Call 2, DFSORT supplies your EFS program with fields in the EFS interface containing:

- An action code indicating that Major Call 2 is in effect
- The original control statement or EXEC PARM option requested by the EFS program
- The length of the original control statement or EXEC PARM option
- Informational flags that describe current processing.

When the EFS program returns control to DFSORT, it can supply fields in the EFS interface containing:

- A modified version of the control statement or EXEC PARM option sent by DFSORT to the EFS program. If you plan to sort or merge user-defined data types (for which you must define a collating sequence), or include or omit user-defined data types, then your EFS program must return new formats for the SORT/MERGE or INCLUDE/OMIT control statements. These new formats (D1 and D2) signal DFSORT to call the EFS01 and EFS02 exit routines you included with your EFS program.
- The length of the altered control statement or EXEC PARM option
- Informational flags signaling DFSORT whether to parse or ignore the control statement or EXEC PARM option.
- A list of messages for DFSORT to print to the message data set.
- A return code (in general register 15).

Major Call 3: At Major Call 3, your EFS program provides DFSORT with user-written messages to print to the message data set. DFSORT can call the EFS program once for the Blockset technique and once for the Peerage/Vale techniques. DFSORT obtains more information at this call from the EFS program to process the EFS01 and EFS02 exit routines.

At Major Call 3, DFSORT supplies your EFS program with fields in the EFS interface containing:

- An action code indicating that Major Call 3 is in effect
- An extract buffer offsets list needed by the EFS01 exit routine
- A record lengths list of input and output records

- Informational flags that describe current processing
- An EFS Program Context area (a private communication area for the EFS program).

When the EFS program returns control to DFSORT, it can supply fields in the EFS interface containing:

- An EFS01 exit routine address
- An EFS02 exit routine address
- A list of messages for printing to the message data set
- A return code in general register 15.

Input Phase

DFSORT executes the two exit routines, EFS01 and EFS02, during the input phase. The EFS01 routine supports sorting or merging user-defined data types with user-defined collating sequences and is called once for each record. The EFS02 routine provides logic to include or omit records on user-defined data types and is called one or more times, according to the include/omit logic.

Information is passed in both directions between DFSORT and the exit routines across the EFS01 and EFS02 parameter lists.

DFSORT supplies the EFS01 routine with fields in the parameter list containing:

- An Extract Buffer Area, which contains an internal version of the input record. The EFS01 routine must move all EFS control fields to this input record. See "EFS02 Exit Routine" on page 342 for more information.
- The input data record.
- An EFS Program Context Area (a private communication area for the EFS program).

When the EFS01 routine returns control to DFSORT, it must return a return code in general register 15.

DFSORT supplies the EFS02 routine with fields in the parameter list containing:

- A Correlator Identifier, which identifies a relational condition containing EFS fields. See "EFS01 Exit Routine" on page 339 for more information.
- The input data record.

When the EFS02 routine returns control to DFSORT, it must return a return code in general register 15.

Termination Phase

DFSORT executes Major Calls 4 and 5 during the termination phase. Only one call is made at each Major Call.

Note: If a system abend occurs while DFSORT's ESTAE recovery routine is in effect, and Major Calls 4 and 5 have not already been executed, the ESTAE routine executes them. If an EFS abend occurs during Major Call 1, the ESTAE routine does not execute Major Calls 4 and 5. See Appendix F, "DFSORT Abend Processing" on page 383 for more information about ESTAE.

Major Call 4: The EFS program provides any user-written messages for printing to the message data set.

At Major Call 4, DFSORT supplies your EFS program with fields in the EFS interface containing:

- An action code indicating that Major Call 1 is in effect
- An EFS Program Context Area (a private communication area for the EFS program).

When the EFS program returns control to DFSORT, it can supply fields in the EFS interface containing:

- A message list containing messages for printing to the message data set
- A return code (in general register 15).

Major Call 5: The EFS program performs any termination processing, such as closing data sets and releasing storage.

At Major Call 5, DFSORT supplies your EFS program with fields in the EFS interface containing:

- An action code indicating that Major Call 5 is in effect
- An EFS Program Context Area (a private communication area for the EFS program).

When the EFS program returns control to DFSORT, it must supply a return code in general register 15.

What You Can Do with EFS

You can design your EFS program to perform seven basic tasks at the initialization, input, and termination phases of DFSORT. Some of the tasks require using the EFS program-generated exit routines EFS01 and EFS02.

Table 21 (Page 1 of 2). Functions of an Extended Function Support (EFS) Program

EFS Program Functions	Initialization Phase	Input Phase	Termination Phase
Opening and initializing	EFS Program		
Examining, altering, or ignoring DFSORT and non-DFSORT control statements prior to processing by DFSORT	EFS Program		
Sorting or merging user-defined data types with user-defined collating sequences		EFS01	
Providing the logic to include or omit records based on user-defined data types		EFS02	

Table 21 (Page 2 of 2). Functions of an Extended Function Support (EFS) Program

EFS Program Functions	Initialization Phase	Input Phase	Termination Phase
Supplying messages to DFSORT for printing to the message data set	EFS Program		EFS Program
Terminating DFSORT	EFS Program	EFS01, EFS02	EFS Program
Closing data sets and housekeeping			EFS Program

Opening and Initializing Data Sets

Your EFS program can open data sets, obtain necessary storage, and perform other forms of initialization needed during a run.

Examining, Altering, or Ignoring Control Statements

At Major Call 1, your EFS program can send a control statement request list to indicate the control statements and EXEC PARM options you want DFSORT to send to your EFS program at Major Call 2.

At Major Call 2, your EFS program can examine, alter, or ignore control statements and EXEC PARM options that DFSORT reads from the EXEC statement, SYSIN, SORTCNTL, DFSPARM, or a parameter list passed from an invoking program. Refer to Figure 50 on page 324 for an illustration of the control statement processing sequence used when an EFS program is activated.

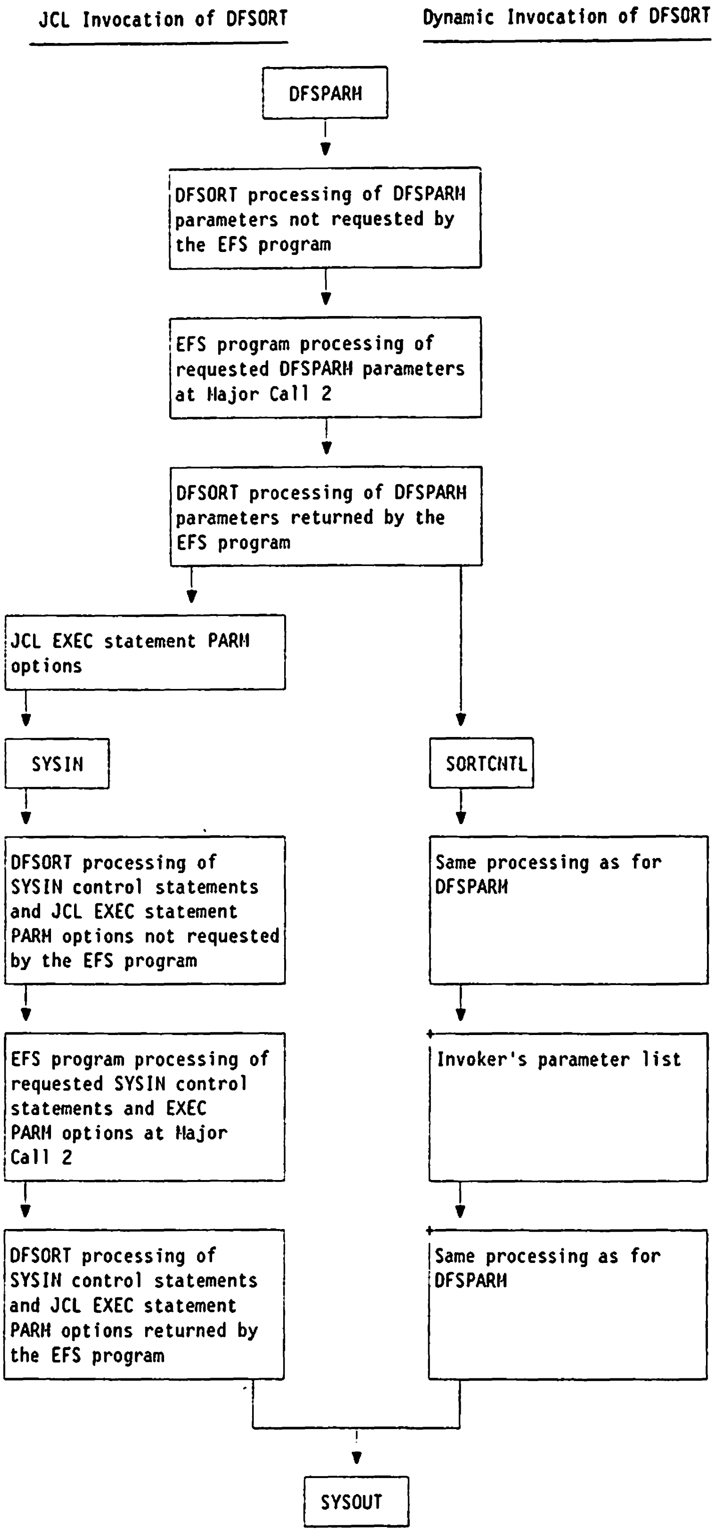

JCL Invocation of DFSORT
Dynamic Invocation of DFSORT
DFSPARM
DFSORT processing of DFSPARM parameters not requested by the EFS program
EFS program processing of requested DFSPARM parameters at Major Call 2
DFSORT processing of DFSPARM parameters returned by the EFS program
JCL EXEC statement PARM options
SYSIN
SORTCNTL
DFSORT processing of SYSIN control statements and JCL EXEC statement PARM options not requested by the EFS program
Same processing as for DFSPARM
EFS program processing of requested SYSIN control statements and EXEC PARM options at Major Call 2
Invoker's parameter list
DFSORT processing of SYSIN control statements and JCL EXEC statement PARM options returned by the EFS program
Same processing as for DFSPARM
SYSOUT

Figure 50. Control Statement Processing Sequence

Processing User-Defined Data Types with EFS Program Exit Routines

You can write your EFS program to provide two exit routines to perform various tasks during execution. Acceptable data types are EBCDIC, ASCII, or user-defined.

Your EFS program exit routines can:

- *Process user-defined data types.* Your EFS program can provide an EFS01 exit routine to alter any control field of an input record. You can also use a user-defined collating sequence for user-defined input records.
- *Include or omit records based on user-defined data types.* Your EFS program can provide an EFS02 exit routine to examine any input field of an input record to determine whether or not to include that record for processing.

Supplying Messages for Printing to the Message Data Set

You can use an EFS program to tailor messages for several purposes:

- To describe new types of operations
- To describe extended field parameters
- To customize the message data set to your site
- To display statistical information about control statements or EXEC PARM options.

You can control whether to print the control statements returned by an EFS program on the message data set with:

- The LISTX operator of the ICEMAC macro (see "Installation Defaults" on page 11).
- The LISTX or NOLISTX operators in the PARM field of the JCL EXEC statement (see "Specifying EXEC Statement PARM Options" on page 21).
- The LIST or NOLIST operators of the OPTION program control statement

Terminating DFSORT

Your EFS program can terminate DFSORT at any of the five Major Calls, and also from either of the two EFS program exit routines during the input phase.

Closing Data Sets and Housekeeping

At Major Call 5, your EFS program can close data sets, free storage and perform any other necessary housekeeping.

Structure of the EFS Interface Parameter List

The EFS interface consists of a variable-length parameter list, and is used to communicate between DFSORT and your EFS program. DFSORT initializes the parameter list to zeros during the initialization phase, except that the list end indicator is set to X'FFFFFFFF'.

The parameter list resides below 16-megabyte virtual, and remains accessible while the EFS program is active, although DFSORT might change its storage location during execution to optimize use of storage. The actual address in register 1 (used to pass the interface parameter list address) can therefore change while DFSORT is running.

Figure 51 illustrates the structure of the EFS interface parameter list. The illustrated portions of the list are explained in order in the following pages. EXEC PARMs are not described in the figure, but are included in processing.

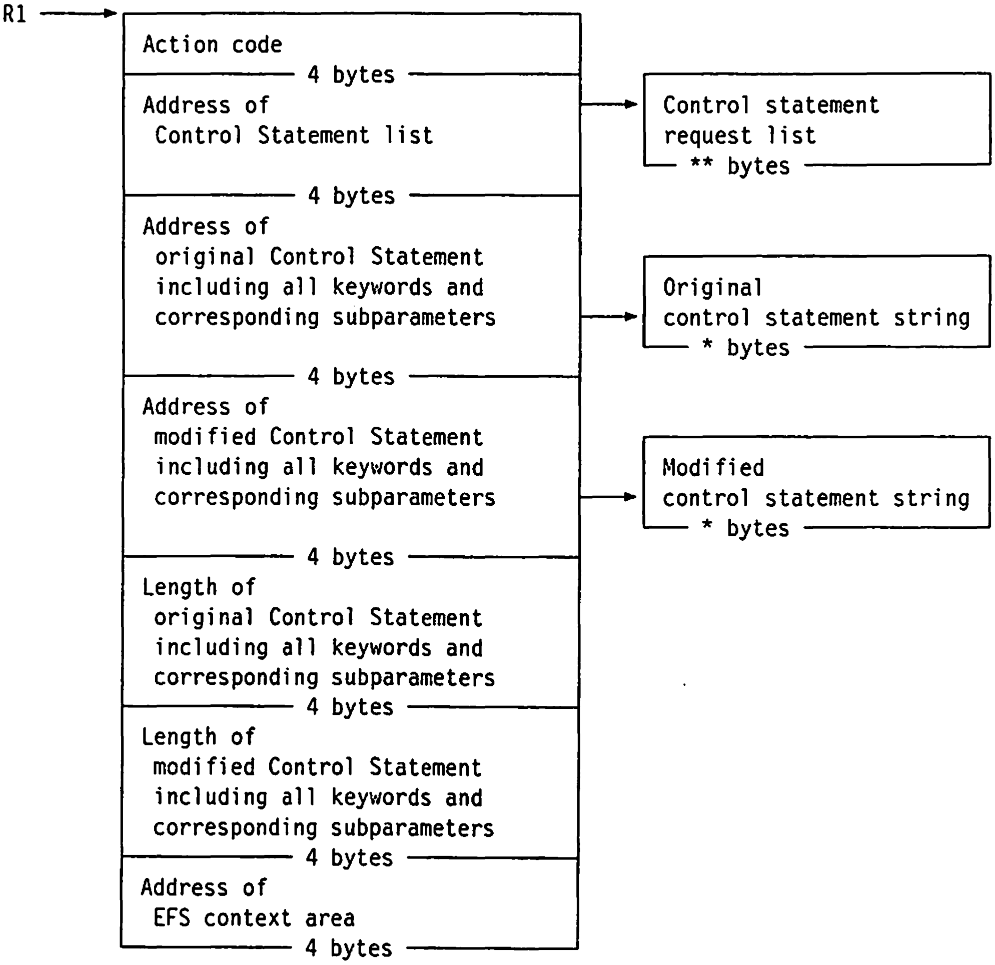
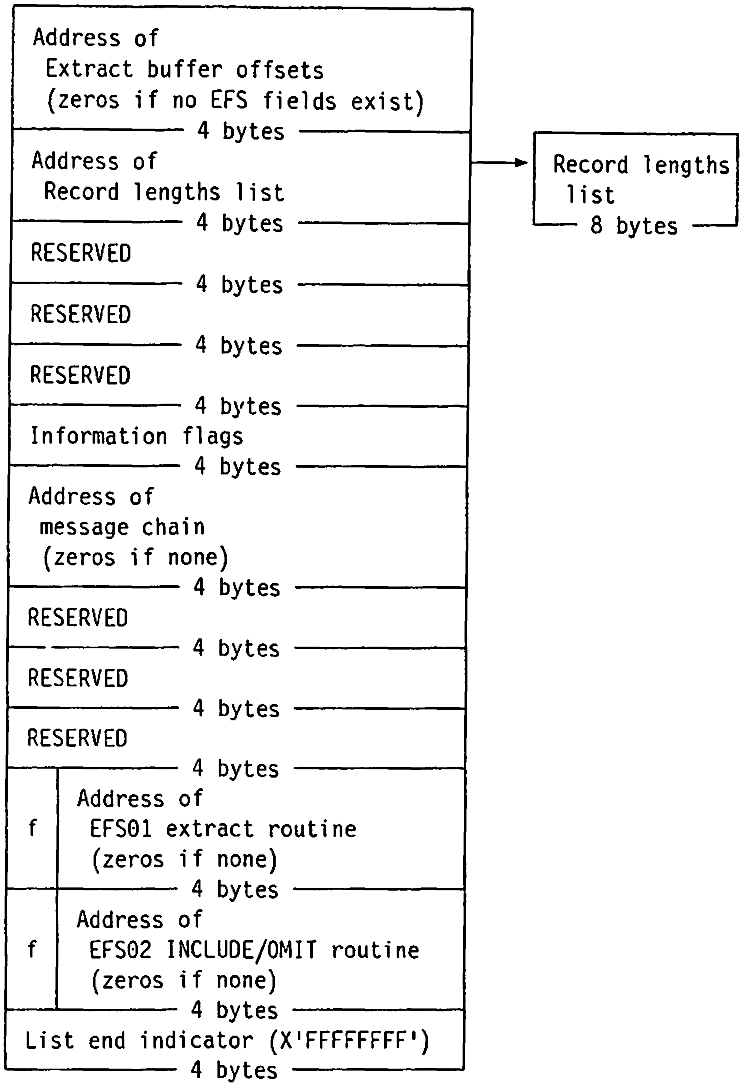

R1
Action code
4 bytes
Address of Control Statement list
4 bytes
Address of original Control Statement including all keywords and corresponding subparameters
4 bytes
Address of modified Control Statement including all keywords and corresponding subparameters
4 bytes
Length of original Control Statement including all keywords and corresponding subparameters
4 bytes
Length of modified Control Statement including all keywords and corresponding subparameters
4 bytes
Address of EFS context area
4 bytes
Control statement request list
** bytes
Original control statement string
* bytes
Modified control statement string
* bytes

Figure 51 (Part 1 of 2). EFS Interface Parameter List



	Field	Length
	Address of Extract buffer offsets (zeros if no EFS fields exist)	4 bytes
	Address of Record lengths list	4 bytes
	RESERVED	4 bytes
	RESERVED	4 bytes
	RESERVED	4 bytes
	Information flags	4 bytes
	Address of message chain (zeros if none)	4 bytes
	RESERVED	4 bytes
	RESERVED	4 bytes
	RESERVED	4 bytes
f	Address of EFS01 extract routine (zeros if none)	4 bytes
f	Address of EFS02 INCLUDE/OMIT routine (zeros if none)	4 bytes
	List end indicator (X'FFFFFFFF')	4 bytes

Record lengths list — 8 bytes

** - Length determined by length fields in list
* - Length determined by corresponding length field

Figure 51 (Part 2 of 2). EFS Interface Parameter List

Action Codes

DFSORT sets one of five action codes before a call to the EFS program:

0 Indicates Major Call 1 to the EFS program. DFSORT sends this action code once.

4 Indicates Major Call 2 to the EFS program. DFSORT might send this action code multiple times at Major Call 2 depending on how many control statements are requested and found. For example, if the SORT, MERGE, and INCLUDE control statements are all supplied in SYSIN and are requested, the EFS program is called twice: once for the SORT control statement (because SORT and MERGE are mutually exclusive, and assuming the SORT statement is specified first, only the SORT statement is taken) and once for the INCLUDE control statement.

8 Indicates Major Call 3 to the EFS program. DFSORT can send this action code once for the Blockset technique and once for the Peerage/Vale technique.

12 Indicates Major Call 4 to the EFS program. DFSORT sends this action code once.

16 Indicates Major Call 5 to the EFS program. DFSORT sends this action code once.

Control Statement Request List

The control statement request list describes the control statements and the PARM options to be sent to the EFS program by DFSORT. The control statement request list consists of control statement operation definers and PARM option names. The maximum length allowed for an operation definer or PARM option name is eight bytes. If the operation definer or PARM option name is longer, DFSORT will use only the first eight bytes. Length field values must not include their own length.

Non-DFSORT operation definers and PARM options must be in EBCDIC format, and the first character must be non-numeric. The format of the control statement request list is:

Chain pointer to next operation definer or EXEC PARM option name, or zero for end of list	Length of operation definer or EXEC PARM option name	Operation definer or EXEC PARM option name (variable length)
4 bytes	2 bytes	* bytes

* indicates that the length is determined by the corresponding length field (maximum of 8 bytes).

Control Statement String Sent to the EFS program

DFSORT scans for the requested control statement from SYSIN, SORTCNTL, DFSPARM, or the invoker's parameter list to create a contiguous control statement string; DFSORT will handle any necessary continuation requirements for control statements from SYSIN, SORTCNTL, or DFSPARM. DFSORT scans for the requested PARM option to create a contiguous PARM option string.

DFSORT places a copy of the requested control statement or PARM option string in a contiguous storage area for the EFS program. No labels are supplied with the control statement; the address of the string always points to the first byte of the appropriate operation definer or PARM option.

DFSORT will send the requested control statement(s) or PARM option(s) to the EFS program as found by DFSORT; DFSORT will provide limited syntax checking of control statements or PARM option(s) before sending them to the EFS program.

In addition to following the rules in "General Coding Rules" on page 55, you must observe the following rules for non-DFSORT control statements:

- DFSORT will recognize a control statement with no operand(s) provided the operation definer (1) is supplied in SYSIN, SORTCNTL, or DFSPARM and (2) is the only operation definer contained on a line.
- *Operation definers supplied through SYSIN, SORTCNTL, DFSPARM, or the extended parameter list and requested by the EFS program will not be recognized if they are longer than eight bytes.*

In addition to observing the rules in your JCL manual, you must observe the following rule for non-DFSORT PARM options:

- *PARM options requested by the EFS program will not be recognized if they are longer than eight bytes.*

DFSORT will send the requested DFSORT or non-DFSORT control statements or PARM options that remain *after DFSORT override rules have been applied.*

If duplicate DFSORT or non-DFSORT control statements or PARM options are supplied through the *same source* (such as SYSIN), then DFSORT will send the first occurrence of the control statement. The second occurrence of the DFSORT or non-DFSORT control statement or PARM option will be ignored by DFSORT.

If duplicate DFSORT or non-DFSORT control statements are supplied through *different sources* (such as extended parameter list, SORTCNTL, and DFSPARM), then DFSORT will send the control statement remaining after different source override rules have been applied, except for the DFSORT OPTION and DEBUG control statements (see "Special Handling of OPTION and DEBUG Control Statements" on page 330).

If mutually exclusive DFSORT control statements (such as SORT/MERGE) are supplied through the *same source* (such as SYSIN), then DFSORT will send the first occurrence of the control statement. The second occurrence of the DFSORT control statement will be ignored by DFSORT.

If mutually exclusive DFSORT control statements (such as SORT/MERGE) are supplied through *different sources* (such as extended parameter list, SORTCNTL, and DFSPARM), then DFSORT will send the control statement remaining after different source override rules have been applied. The DFSORT control statement not sent will be ignored by DFSORT.

Thus the EFS program will not be sent duplicate DFSORT or non-DFSORT control statements (except for the DFSORT OPTION and DEBUG control statements as explained in "Special Handling of OPTION and DEBUG Control Statements" on page 330), or duplicate PARM options.

If the EFS program supplies non-DFSORT operands on the DFSORT OPTION control statement and the OPTION control statement is supplied in the extended parameter list, the EFS program must specify the non-DFSORT operands *after* all DFSORT operands.

DFSORT will free any storage it acquired for the control statement or PARM string.

Note: Blanks and quotes are very important to DFSORT in determining the control statement to send to an EFS program. Do not supply unpaired quotes in the INCLUDE/OMIT control statements, because DFSORT treats data within quotes as a constant, and treats blanks outside of quotes as the major delimiter.

Special Handling of OPTION and DEBUG Control Statements

The override features of both the DFSORT OPTION and DEBUG control statements, when supplied through different sources, require special handling when EFS processing is in effect and either or both control statements are requested by the EFS program.

For example, DFSORT handles override for the OPTION and DEBUG control statements as follows:

- The OPTION control statement supplied in SORTCNTL will selectively override corresponding options on the OPTION control statement supplied in the extended parameter list.
- The DEBUG control statement supplied in SORTCNTL will selectively override corresponding options on the DEBUG control statement supplied in the 24-bit parameter list or the extended parameter list.

Because of these override features, DFSORT cannot simply send the OPTION control statement supplied in SORTCNTL and not send the OPTION control statement supplied in the extended parameter list. For the EFS program to process all possible operands on the OPTION control statements, DFSORT must send the OPTION control statements supplied in both SORTCNTL and the extended parameter list. DFSORT will thus send both the OPTION and DEBUG control statements supplied through different sources. If duplicate OPTION or DEBUG control statements are supplied in the same source and the OPTION or DEBUG control statements are also supplied in different sources, DFSORT will send the first occurrence of both the OPTION and DEBUG control statements supplied through different sources.

Control Statement String Returned by the EFS Program

Your EFS program can alter the control statement or PARM option string and replace it in the original contiguous storage area. If the area is too small, your program must allocate a new contiguous area. If the string is returned in a new storage area, your EFS program will be responsible for freeing the acquired storage.

Your EFS program must set an Informational flag to indicate whether the control statement or PARM option in the string should be parsed or ignored by DFSORT (see "Information Flags" on page 335 for further details).

Rules for Parsing

The content and format of the altered control statement to be parsed must correspond to valid DFSORT values as described in Chapter 3, "Using DFSORT Program Control Statements" on page 53, except when using the FIELDS operand with SORT or MERGE, or the COND operand with INCLUDE or OMIT (see "EFS Formats for SORT, MERGE, INCLUDE, and OMIT Control Statements" on page 331).

You must observe the following rules for control statements to be returned to DFSORT for parsing:

- The operation definer and corresponding operands must be in uppercase EBCDIC format.

- At least one blank must follow the operation definer (SORT, MERGE, RECORD, and so forth). A control statement can start with one or more blanks and can end with one or more blanks. No other blanks are allowed unless the blanks are part of a constant.
- Labels are not allowed; a leading blank, or blanks, before the control statement name is optional.
- No continuation character is allowed.
- Neither comment statements nor comment fields are allowed.

The content and format of the altered EXEC PARM option to be parsed must correspond to valid DFSORT values as described in "Specifying EXEC Statement PARM Options" on page 21.

The following operands will be ignored by DFSORT if returned by an EFS program on the OPTION control statement:

EFS
LIST
NOLIST
LISTX
NOLISTX
MSGDDN
MSGDD
MSGPRT
SORTDD
SORTIN
SORTOUT

The following EXEC PARM options will be ignored by DFSORT if returned by an EFS program:

EFS
LIST
NOLIST
LISTX
NOLISTX
MSGDDN
MSGDD
MSGPRT

EFS Formats for SORT, MERGE, INCLUDE, and OMIT Control Statements

In addition to using the SORT, MERGE, INCLUDE, and OMIT control statements as explained in "Program Control Statements," you can also use two additional formats on the FIELDS and COND parameters. The formats are termed D1 and D2 and apply as follows:

- D1 with the SORT or MERGE FIELDS parameter
- D2 with the INCLUDE or OMIT COND parameter.

Use D1 and D2 to reflect data types that require special processing by EFS program exit routines EFS01 and EFS02, respectively.

D1 Format on FIELDS Operand

The syntax for the SORT and MERGE statements using the D1 format on the FIELDS operand is as follows.

```
                       ┌──────,──────┐
                       ↓             │
►►──┬─SORT──┬──────FIELDS=(─mp,mm,mf,ms─┴─)──────────────────►◄
    └─MERGE─┘
```

Where **Represents**

mp field position within the input record

mm field length

mf the D1 format that designates this field as an EFS control field

ms must be either ascending (A) or descending (D); modification by an E61 exit (E) is not allowed.

Figure 52 gives an example of using the D1 format for a SORT control statement returned to DFSORT by the EFS program.

You must adhere to the following requirements for the D1 format:

- The mp, mm, and ms values returned must be valid SORT or MERGE control statement values, except:
  - The combined value of mp and mm may exceed the record length.
  - CHALT will have no effect on EFS fields and will not limit the length to 256.
  - Value E for ms will not be allowed; EFS fields may not be altered by an E61.
  - FORMAT=D1 will not be allowed.

Original SORT control statement sent to EFSPGM

```
SORT FIELDS=(15,4,FF,A,20,4,CH,A,40,7,FF,D)
```

Altered SORT control statement returned by EFSPGM

```
SORT FIELDS=(15,4,D1,A,20,4,CH,A,40,7,D1,D)
```

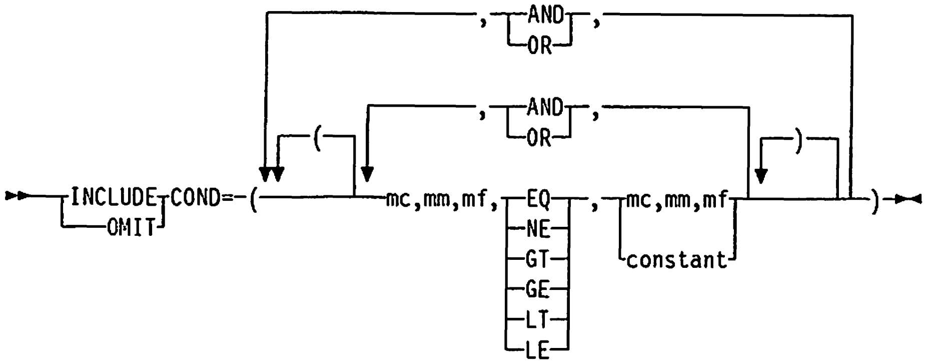
where:

> FF is a user-defined format that is modified to D1
> by the EFS program before returning to DFSORT

Figure 52. D1 Format Returned by an EFS Program

D2 Format on COND Operand

Following is the syntax for the INCLUDE or OMIT statements using the D2 format on the COND operand.



Where	Represents
mc	the correlator identifier, a numeric value used to identify each relational condition
mm	field length
mf	the D2 format designating an EFS field within the relational condition
constant	a valid DFSORT decimal, character, or hexadecimal constant.

Figure 53 on page 334 gives an example of using a correlator identifier and the D2 format for an INCLUDE control statement returned to DFSORT by the EFS program.

Note: The values for the correlator identifiers assigned to each relational condition by the EFS program can be in any chosen order. The example in Figure 53 shows a sequential ordering for the correlator identifiers.

You must adhere to the following requirements for the D2 format:

- The mc, mm, or constant values returned must be valid INCLUDE or OMIT control statement values, except:
  - The combined value of mc and mm may exceed the record length.
  - Any valid DFSORT constant will be allowed.
  - If COND=(mc1,mm1,mf1,comparison operator,mc2,mm2,mf2) is used, both mf1 and mf2 must be D2.
  - CHALT will have no effect on EFS fields.
  - FORMAT=D2 will not be allowed.

Original INCLUDE control statement sent to EFSPGM

```
INCLUDE COND=(15,4,FF,EQ,20,4,FF,AND,40,7,FF,NE,50,7,FF,OR,
              30,2,FF,NE,35,2,FF)
```

Altered INCLUDE control statement returned by EFSPGM

```
INCLUDE COND=(1,4,D2,EQ,1,4,D2,AND,2,7,D2,NE,2,7,D2,OR,
              3,2,D2,NE,3,2,D2)
```

Where:

- FF is a user-defined format and modified to D2 by the EFS program before returning to DFSORT.,
- The first relational condition specified, (1,4,D2,EQ,1,4,D2), uses correlator identifier value 1 to identify this relational condition.
- The second relational condition specified, (2,7,D2,NE,2,7,D2), uses correlator identifier value 2 to identify this relational condition.
- The third relational condition specified, (3,2,D2,NE,3,2,D2), uses correlator identifier value 3 to identify this relational condition

Figure 53. Correlator Identifier and D2 Format Returned by an EFS Program

Length of Original Control Statement

The control statement includes the first byte of the control statement through the last operand of the control statement or, if only an operation definer is supplied, the length of the operation definer. DFSORT does not send labels supplied with the control statement.

Length of the Altered Control Statement

The length includes the first byte of the control statement through the last operand of the control statement. If leading blanks are provided, the length includes the first leading blank.

EFS Program Context Area

The EFS program context area is a private communication area that can be set up and used by the EFS program as appropriate. DFSORT sends the context area address to the EFS program at each Major Call and at each call to EFS01 and EFS02.

The EFS program is responsible for obtaining (at Major Call 1) and releasing (at Major Call 5) the necessary storage for the EFS program context area.

Extract Buffer Offsets List

A linked list of offsets into the extract buffer will be passed to your EFS program. The offsets show the starting positions into the buffer area of any EFS control fields specified on the SORT or MERGE FIELDS operand. The offsets are sent only for EFS control fields and are sent in the same order as specified on the FIELDS operand. If no EFS control fields exist, the address to the offsets is zero.

DFSORT frees any storage it acquired for the extract buffer offsets list. The format of the extract buffer offsets list is:

Chain pointer to the next offset or zero for end of list	Offset n
4 bytes	4 bytes

Record Lengths List

The record lengths list is a linked list containing the input and output record lengths. You must be aware of the possible change in record size during execution (for example, with an E15 user exit).

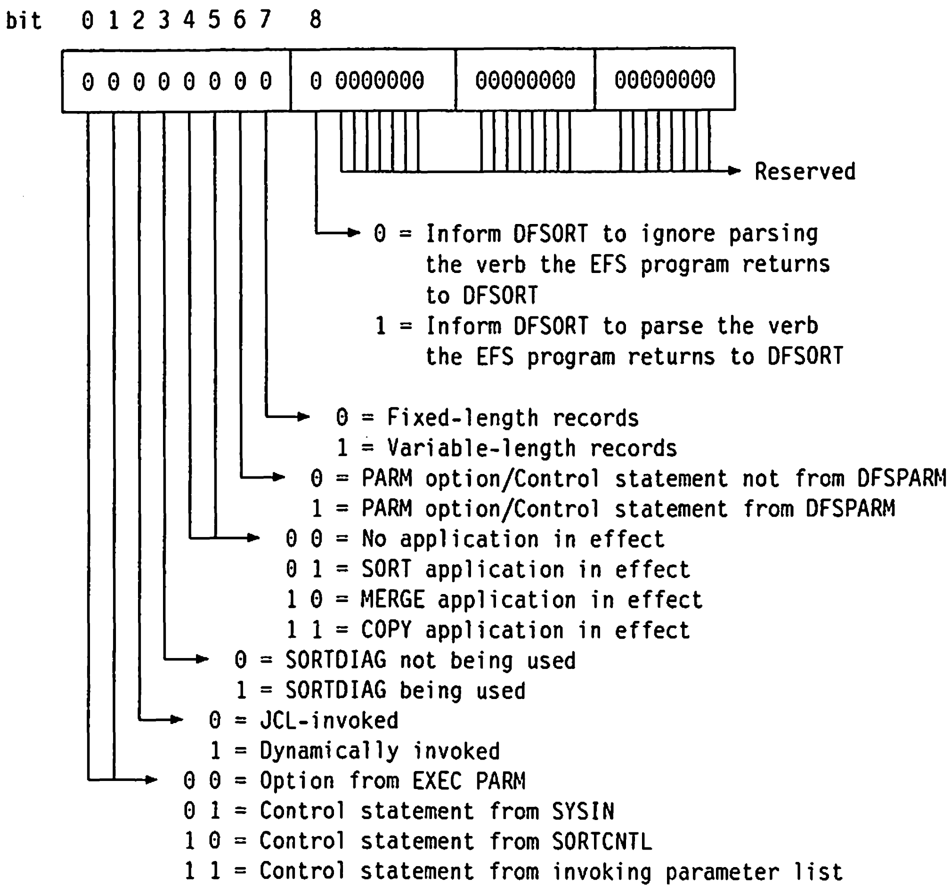
The input and output record lengths are sent to the EFS program for informational use only. DFSORT ignores any changes to the values made to the record lengths list returned by the EFS program.

DFSORT frees any storage it acquired for the record lengths list. The format of the record lengths list is:

Input record length	Output record length
4 bytes	4 bytes

Information Flags

The information flags are defined as follows:



0 = Inform DFSORT to ignore parsing the verb the EFS program returns to DFSORT
1 = Inform DFSORT to parse the verb the EFS program returns to DFSORT

0 = Fixed-length records
1 = Variable-length records

0 = PARM option/Control statement not from DFSPARM
1 = PARM option/Control statement from DFSPARM

0 0 = No application in effect
0 1 = SORT application in effect
1 0 = MERGE application in effect
1 1 = COPY application in effect

0 = SORTDIAG not being used
1 = SORTDIAG being used

0 = JCL-invoked
1 = Dynamically invoked

0 0 = Option from EXEC PARM
0 1 = Control statement from SYSIN
1 0 = Control statement from SORTCNTL
1 1 = Control statement from invoking parameter list

Bit	**Description**
Bits 0 and 1	Indicates the source of the control statement being processed. Information flags 0 and 1 are set by DFSORT before a call(s) to the EFS program at Major Call 2 (multiple calls are possible at Major Call 2).
Bit 2	Indicates how DFSORT was invoked. Information flag 2 is set by DFSORT before Major Call 1 to the EFS program.
Bit 3	Indicates whether diagnostic messages are to be printed. Information flag 3 will be set by DFSORT before Major Call 1 to the EFS program.
Bits 4 and 5	Indicate the DFSORT function being executed. Information flags 4 and 5 will be set by DFSORT before each call(s) at Major Call 2 and Major Call 3 to the EFS program (multiple calls are possible at Major Call 2 and Major Call 3).
Bit 6	Indicates the source of PARM options and control statements from DFSPARM. Information flag 6 will be set by DFSORT before each call or calls at Major Call 2 to the EFS program (multiple calls are possible at Major Call 2).

Bit 7

Indicates whether fixed-length records or variable-length records are to be processed. Information flag 7 is set by DFSORT before each call(s) at Major Call 3 to the EFS program (multiple calls are possible at Major Call 3).

Bit 8

Set by the EFS program to inform DFSORT whether to parse or ignore the control statement returned by the EFS program. Printing of the control statement will be managed by the LISTX/NOLISTX parameters (see "OPTION Control Statement" on page 97 for further details). Information flag 8 is set by the EFS program before returning to DFSORT from each call(s) at Major Call 2 (multiple calls are possible at Major Call 2).

Message List

Your EFS program can return informational or critical messages. A return code of 0 in general register 15 indicates an informational message while a return code of 16 indicates a critical message. If the EFS program has no messages to send after a Major Call, it must zero the message list address in the EFS interface parameter list.

At Major Call 2, if the EFS program finds a syntax error in a control statement, it can return an offset relative to the start of the string to indicate the location of the error. DFSORT will first print the control statement in error and then will print another line containing a dollar symbol ($) at the location indicated by the offset.

Because DFSORT associates the relative offset with a critical message, the EFS program must return with a return code of 16 in general register 15. If a relative offset is returned for an EXEC PARM, the relative offset will be ignored. The EFS program must free any storage it acquired for its messages.

The length field values must not include their own length.

The message list format follows.

Chain pointer to next message or zero for list end	Relative offset (to syntax error) or zero	Length of the message text	Message text (variable length)
4 bytes	2 bytes	2 bytes	* bytes

* indicates that the length is determined by the corresponding length field.

DFSORT imposes no restrictions on the format of the message(s) returned by an EFS program. If you wish, you can use the DFSORT message format so that messages in the message data set are consistent in appearance. For a description of the message format used by DFSORT, see *DFSORT Messages and Codes.*

EFS Program Exit Routines

If you specify EFS control fields (D1 format) or EFS fields (D2 format), DFSORT calls the EFS01 or EFS02 exit routines, respectively, to process those fields. The routines are generated by your EFS program, which can return the following information about them at Major Call 3:

- The address of an extract routine, EFS01, which is used to extract the control fields of an input record to a new internal record before a sort or merge takes place; EFS01 is not applicable to a copy application.
- The address of an INCLUDE or OMIT routine, EFS02, which is used to process comparison logic for including or omitting records.

During the termination phase, the EFS program must free any storage used by these routines.

EFS01 and EFS02 Function Description

Each DFSORT control statement describes to DFSORT the type of operation to be performed on input data. Through the EFS interface, DFSORT will allow an EFS program to provide exit routines to perform functions beyond the capabilities of DFSORT control statements.

The EFS program will be allowed to provide exit routine EFS01 to supplement the function of the DFSORT SORT/MERGE control statements and provide exit routine EFS02 to perform the function of the DFSORT INCLUDE/OMIT control statements.

When preparing your EFS program exit routines, remember:

- The routines must follow standard linkage conventions.
- The general registers used by DFSORT for linkage and communication of parameters follow operating system conventions (see Figure 54).
- The routines must use the described interfaces (see "EFS01 Parameter List" on page 341 and "EFS02 Parameter List" on page 343).

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 can be used to save contents of registers used by the EFS program exit routine. 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 the EFS program exit routine. This register can be used as the base register for EFS program exit routine. This register is also used by the EFS program exit routine to pass return codes to DFSORT.

Figure 54. DFSORT Register Convention

EFS01 Exit Routine

Processing of user-defined data types with the EFS01 exit routine requires using the function of altering control statements. EFS program requirements at Major Calls 1 and 2 are:

> At Major Call 1, the EFS program must provide the control statement request list with the SORT or MERGE operation definer. See "Control Statement Request List" on page 328 for further details.
>
> At Major Call 2, the EFS program must return a new format, D1, on the SORT or MERGE control statement which informs DFSORT to call the EFS program EFS01 exit routine, (the control fields defined with the D1 format are also known as EFS control fields). See "EFS Formats for SORT, MERGE, INCLUDE, and OMIT Control Statements" on page 331 for further details. The EFS program must also return the final position, length, and sequence order. DFSORT uses the final position and length to create a list of offsets.

At Major Call 3, DFSORT sends the EFS program a list of offsets into a buffer. These offsets indicate where in the buffer the EFS program must have the EFS01 exit routine move the data indicated by the EFS control fields. See "Extract Buffer Offsets List" on page 335 for further details. The buffer, known as an extract buffer, contains an internal record created by DFSORT and is used by DFSORT in the SORT or MERGE process. At Major Call 3, the EFS program must return the address of the EFS01 exit routine to DFSORT.

During the input phase, DFSORT calls the EFS01 exit routine for each input record.

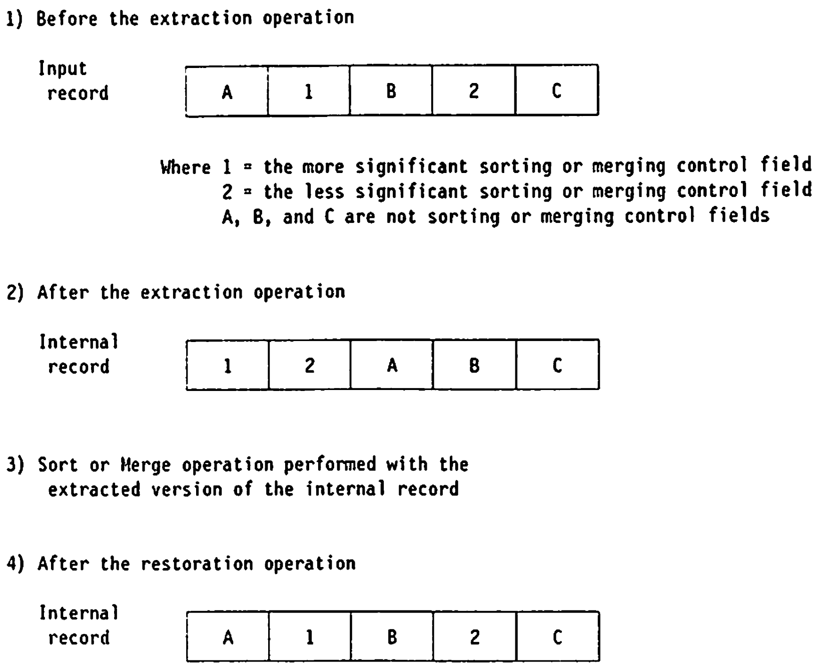
The internal operation performed by DFSORT for SORT/MERGE is:

1. The first operation involves creating an internal record by reorganizing a copy of the input record so that the control fields specified for each record (as described by the FIELDS operand) are placed at the beginning of each record. This process is known as control field extraction.
2. DFSORT then performs the sorting/merging operation using the internal record.
3. The third operation is the reverse of the first operation; the internal record is restored to the original input record format. This process is known as control field restoration.

Figure 55 on page 340 shows a picture of the internal record after the extraction process and the restoration process.

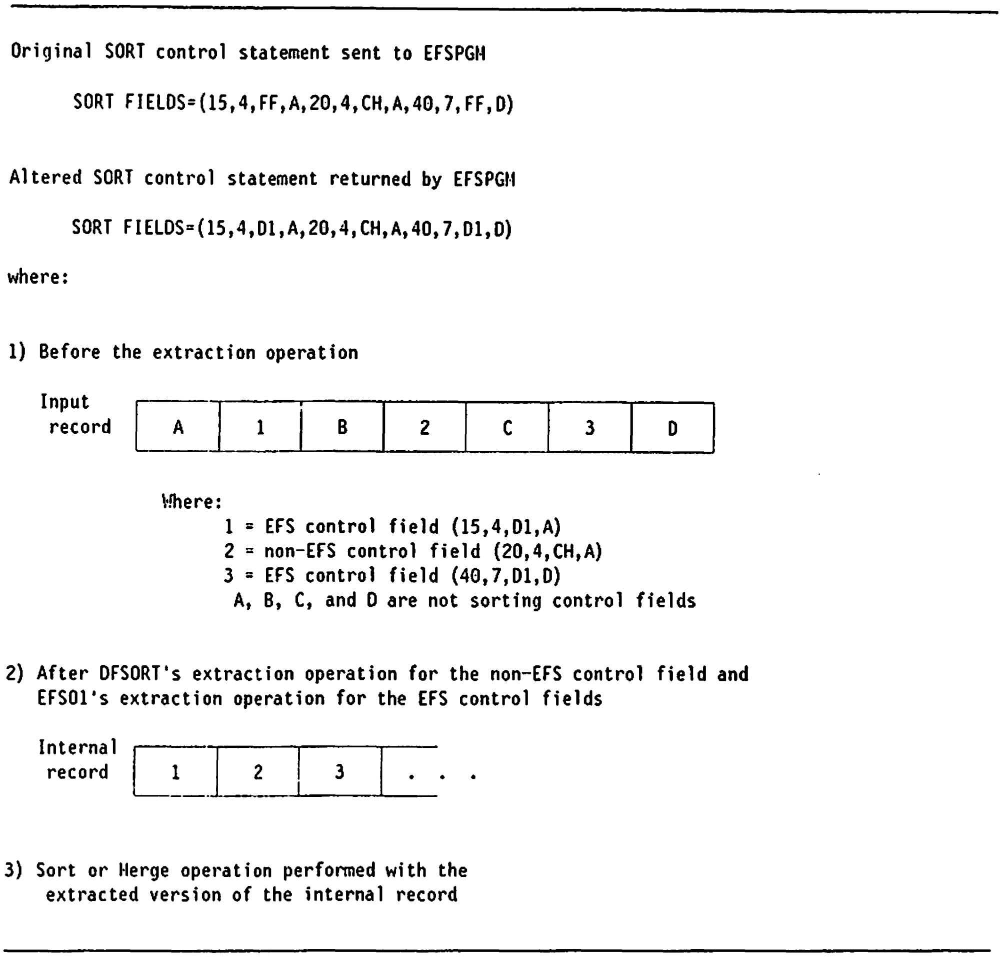
EFS01 must perform the first operation, control field extraction, for any EFS control fields specified on the SORT/MERGE FIELDS operand (note that EFS01 will not be called for a Copy application). The EFS01 exit routine must move all data indicated by the EFS control fields, in the SORT or MERGE FIELDS operand, from the input record to the extract buffer area as specified by the offsets in the extract buffer offsets list. For each EFS control field, the total number of bytes moved by EFS01 into the buffer area is equal to the total number of bytes specified in the *mm* parameter of the altered SORT or MERGE operand. Records are ordered according to the altered *ms* parameter.

The EFS01 exit routine is called to extract all EFS control fields to the extract buffer area each time a new record is brought into the input phase. Figure 56 on page 341 repeats Figure 52 on page 332 to illustrate the extraction operation required by EFS01.



```
1) Before the extraction operation

   Input      ┌───┬───┬───┬───┬───┐
    record    │ 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

2) After the extraction operation

   Internal   ┌───┬───┬───┬───┬───┐
    record    │ 1 │ 2 │ A │ B │ C │
              └───┴───┴───┴───┴───┘

3) Sort or Merge operation performed with the
    extracted version of the internal record

4) After the restoration operation

   Internal   ┌───┬───┬───┬───┬───┐
    record    │ A │ 1 │ B │ 2 │ C │
              └───┴───┴───┴───┴───┘
```

Figure 55. Internal Record Created by DFSORT



```
Original SORT control statement sent to EFSPGM

     SORT FIELDS=(15,4,FF,A,20,4,CH,A,40,7,FF,D)

Altered SORT control statement returned by EFSPGM

     SORT FIELDS=(15,4,D1,A,20,4,CH,A,40,7,D1,D)
where:

1) Before the extraction operation

  Input    ┌───┬───┬───┬───┬───┬───┬───┐
   record  │ A │ 1 │ B │ 2 │ C │ 3 │ D │
           └───┴───┴───┴───┴───┴───┴───┘

             Where:
                  1 = EFS control field (15,4,D1,A)
                  2 = non-EFS control field (20,4,CH,A)
                  3 = EFS control field (40,7,D1,D)
                   A, B, C, and D are not sorting control fields

2) After DFSORT's extraction operation for the non-EFS control field and
   EFS01's extraction operation for the EFS control fields

  Internal ┌───┬───┬───┬────
   record  │ 1 │ 2 │ 3 │ . . .
           └───┴───┴───┴────

3) Sort or Merge operation performed with the
    extracted version of the internal record
```

Figure 56. EFS01 Extraction Operation

EFS01 Parameter List

DFSORT sends three words to the EFS01 exit routine each time it is entered:

- The address of the extract buffer area
- The address of the input record
- The address of the EFS program context area.

DFSORT places the address of a parameter list in register 1. The list begins on a fullword boundary and is three fullwords long. The format of the parameter list is:

Bytes 1 through 4
Address of the Extract Buffer Area
Address of the Input Record
Address of the EFS Program Context Area

The EFS01 exit routine must return one of the following return codes in general register 15:

0 The extraction of the EFS control field was successful.

16 The extraction of the EFS control field was unsuccessful; terminate DFSORT.

EFS02 Exit Routine

Including or omitting records based on user-defined data types with the EFS02 exit routine requires using the function of altering control statements. EFS program requirements at Major Calls 1 and 2 are:

> At Major Call 1, the EFS program must provide the control statement request list with the INCLUDE or OMIT operation definer. See "Control Statement Request List" on page 328 for further details.
>
> At Major Call 2, the EFS program must return a new format, D2, on the INCLUDE or OMIT control statement that informs DFSORT to call the EFS program EFS02 exit routine (the fields defined with the D2 format are also known as EFS fields). See "EFS Formats for SORT, MERGE, INCLUDE, and OMIT Control Statements" on page 331 for further details. The EFS program must also return the final length and, in place of the position value, must send an identifier (known as a correlator identifier) that identifies a specific relational condition. For each relational condition containing EFS fields, there must be a unique correlator identifier to identify the particular relational condition. See "EFS Formats for SORT, MERGE, INCLUDE, and OMIT Control Statements" on page 331 for further details.

At Major Call 3, the EFS program must return the address of the EFS02 exit routine to DFSORT.

During the input phase, DFSORT will call the EFS02 exit routine for each input record according to the evaluation defined by the AND, OR, or parentheses. Thus, the EFS02 exit routine must use the correlator identifier to determine the current relational condition being performed.

The internal operation performed by DFSORT for INCLUDE/OMIT is to use the specified comparison logic on each record to determine if the record satisfies the INCLUDE/OMIT requirements.

Note: EFS02 must perform this operation for any EFS fields specified on the INCLUDE/OMIT COND operand.

The EFS02 exit routine is called to perform the INCLUDE or OMIT comparison logic for each relational condition containing an EFS field. During the input phase, DFSORT will call the EFS02 exit routine for each input record according to the evaluation defined by the AND, OR, or parentheses. The EFS02 exit routine must use the correlator identifier to determine the current relational condition being performed. EFS02 must perform the comparison logic for the current relational condition as identified by the correlator identifier. Figure 57 on page 343 repeats Figure 53 on page 334 to illustrate an example of the calling sequences to an EFS02 by DFSORT.

```
Original INCLUDE control statement sent to EFSPGH

       INCLUDE COND=(15,4,FF,EQ,20,4,FF,AND,40,7,FF,NE,50,7,FF,OR,
                     30,2,FF,NE,35,2,FF)

Altered INCLUDE control statement returned by EFSPGH

       INCLUDE COND=(1,4,D2,EQ,1,4,D2,AND,2,7,D2,NE,2,7,D2,OR,
                     3,2,D2,NE,3,2,D2)

where the calling sequence to EFS01 may be summarized
with the following tables:
```

Relational condition with	EFS02 returns a return code of 0=True or 4=False	DFSORT action when the next logical operator is:
		AND
Correlator Identifier 1	True	Call EFS02 with Correlator Id 2
	False	Call EFS02 with Correlator Id 3

Relational condition with	EFS02 returns a return code of 0=True or 4=False	DFSORT action when the next logical operator is:
		OR
Correlator Identifier 2	True	Include the record
	False	Call EFS02 with Correlator Id 3

Relational condition with	EFS02 returns a return code of 0=True or 4=False	DFSORT action when the next logical operator is:
		None
Correlator Identifier 3	True	Include the record
	False	Omit the record

Figure 57. Calling Sequence to EFS02 by DFSORT

EFS02 Parameter List

DFSORT sends three words to the EFS02 exit routine each time it is entered:

- The address of the correlator identifier
- The address of the input record
- The address of the EFS program context area.

DFSORT places the address of a parameter list in register 1. The list begins on a fullword boundary and is three fullwords long. The format of the parameter list is:

Byte 1	Byte 2	Byte 3	Byte 4
00	00	00	Correlator identifier
Address of the input record			
Address of the EFS program context area			

The EFS02 exit routine must return one of the following return codes in general register 15:

0 True

The record passed the INCLUDE or OMIT test for the relational condition of an EFS field. If applicable, processing continues with the next relational condition. Otherwise, DFSORT accepts the record if INCLUDE is specified or omits the record if OMIT is specified.

4 False

The record did not pass the INCLUDE or OMIT test for the relational condition of an EFS field. If applicable, processing continues with the next relational condition. Otherwise, DFSORT omits the record if INCLUDE is specified or includes the record if OMIT is specified.

16 Terminate

An error occurred in processing the INCLUDE or OMIT logic; terminate DFSORT.

MVS/XA and MVS/ESA Support of EFS Program Exit Routines

If the EFS program exit routine executes in an MVS/XA or an MVS/ESA environment, DFSORT supplies the following features to allow residence above or below 16-megabyte virtual and use of either 24-bit or 31-bit addressing:

f (bit 0 of EFS program exit routine address)

0 Enter the EFS program exit routine with 24-bit addressing in effect.

1 Enter the EFS program exit routine with 31-bit addressing in effect.

The EFS program exit routine can 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 EFS program context area address (which DFSORT sends to the EFS program exit routine unchanged), DFSORT handles the EFS program exit routine parameter list addresses (that is, the pointer to the EFS program exit routine parameter list and the addresses in the parameter list) as follows:

- If the EFS program exit routine is entered with 24-bit addressing in effect, DFSORT can pass clean (zeros in the first 8 bits) 24-bit addresses and/or 31-bit addresses to the EFS program exit routine. The EFS program exit routine must return clean 24-bit addresses if the EFS program exit routine returns to DFSORT with 31-bit addressing in effect.

- If the EFS program exit routine is entered with 31-bit addressing in effect, DFSORT can pass clean 24-bit addresses and/or 31-bit addresses to the EFS program exit routine. The EFS program exit routine must return 31-bit addresses or clean 24-bit addresses.

EFS Program Return Codes You Must Supply

Your EFS program must pass one of two return codes to DFSORT:

0 Continue Processing

If you want DFSORT to continue processing for this Major Call, return with a return code of zero in general register 15.

16 Terminate DFSORT

If you want DFSORT to terminate processing for this Major Call, return with a return code of 16 in general register 15.

If the EFS program returns a return code of 16 from a Major Call prior to Major Call 4 or one of its generated exit routines returns a return code of 16, DFSORT will skip interim Major Calls, where applicable, to the EFS program or exit routine, and will call the EFS program at Major Call 4 and at Major Call 5.

.Multiple calls are possible at Major Call 2 and Major Call 3. If the EFS program returns with a return code of 16 from one of the multiple calls at Major Call 2, subsequent calls at Major Call 2, if applicable, will be completed. If the EFS program returns with a return code of 16 from one of the multiple calls at Major Call 3, subsequent calls at Major Call 3, if applicable, will not be completed.

If the EFS program returns a return code of 16 at Major Call 4, DFSORT will still call the EFS program at Major Call 5.

Record Processing Order

The order of record processing when using :i1/record processing order EFS is similar to processing without it. Figure 58 on page 346 illustrates the record processing sequence for a sort or merge and Figure 59 on page 347 illustrates the record processing sequence for a copy when EFS processing is in effect.

The figures illustrate the same points as described in Figure 2 on page 6 with the following exceptions:

- When record processing is done for an INCLUDE or OMIT control statement, an EFS02 exit routine is called to perform the comparison logic for the relational conditions with EFS fields.
- When record processing is done for a SORT or MERGE control statement, an EFS01 exit routine is called to perform the extraction process for EFS control fields.

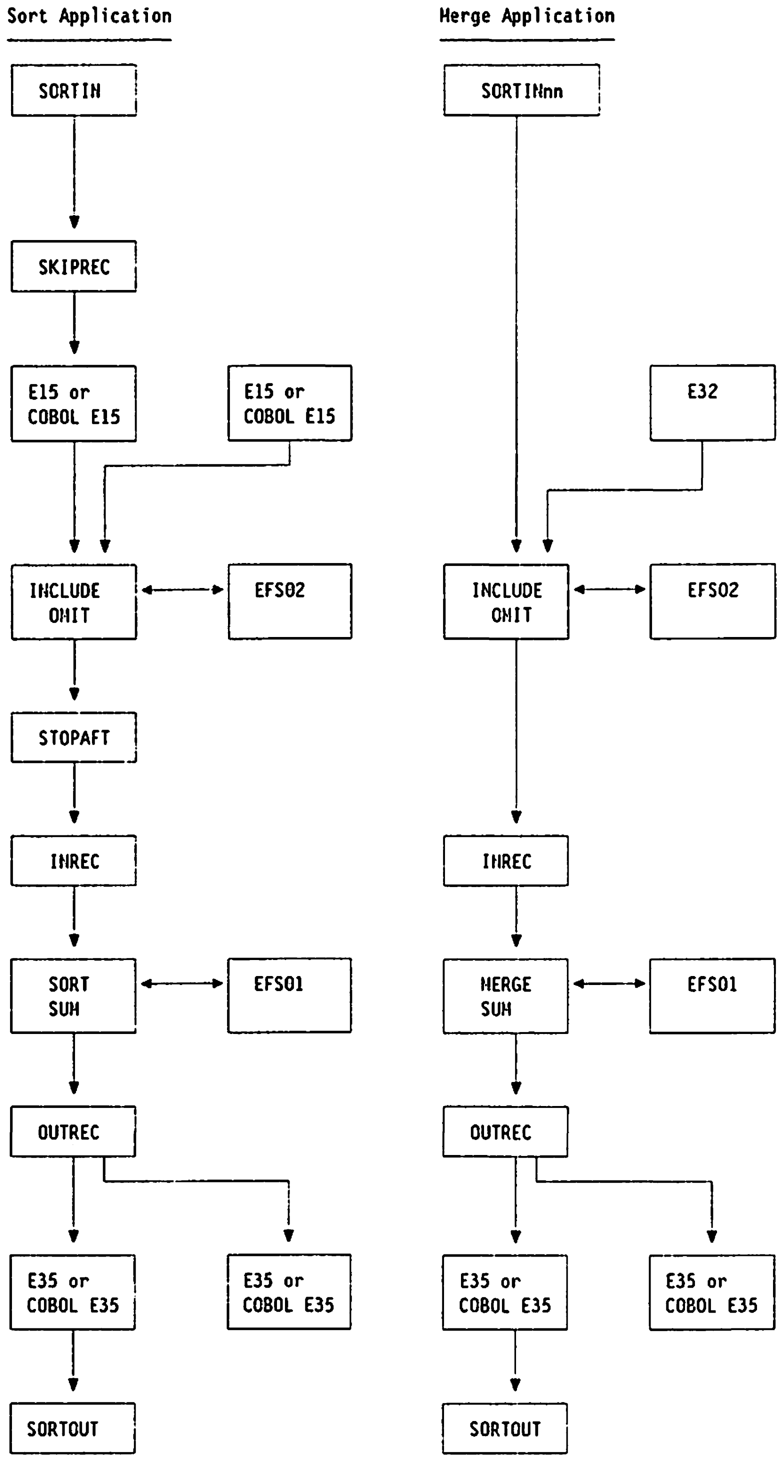


Figure 58. EFS Record Processing Sequence for a Sort or Merge

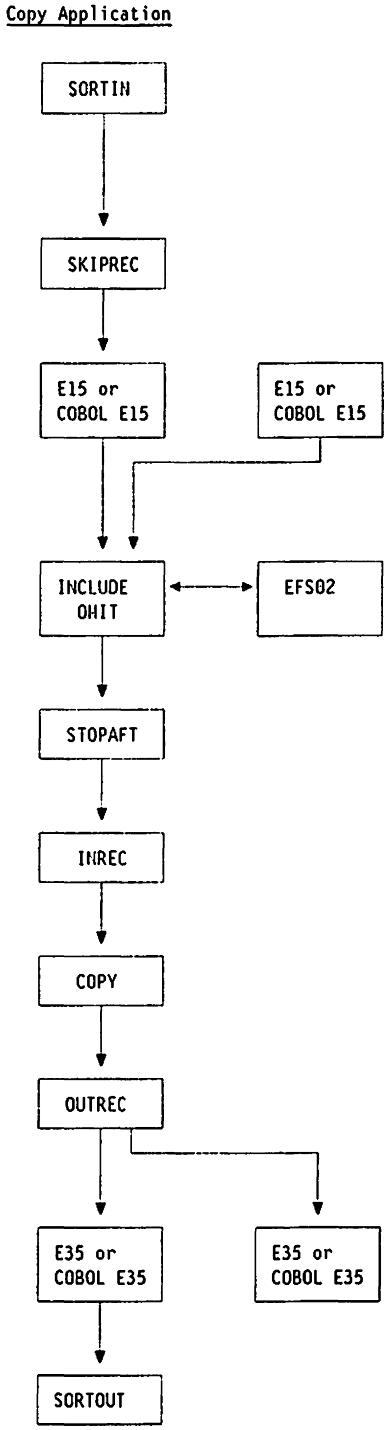


Figure 59. EFS Record Processing Sequence for a Copy

How to Request a SNAP Dump

You can request a SNAP dump for diagnostic purposes before or after any Major Call except Major Call 1. Use either the EFSDPBFR parameter or the EFSDPAFT parameter on the DEBUG statement.

See "DEBUG Control Statement" on page 62 for the correct syntax.

EFS Program Example

The following example shows how an EFS program can be used to change control statements at execution time.

The following information is assumed for this example DFSORT run:

- The EFS program "EFSPGM" resides in the same library as the DFSORT modules.
- The JCL statements for the job are:

```
//EXAMPLE1 JOB A12345,'J. SMITH'
//S1 EXEC  PGM=SORT,PARM='EFS=EFSPGM'
//SYSOUT   DD  SYSOUT=A
//SORTIN   DD  DSNAME=SMITH.INPUT,DISP=SHR,
//     UNIT=3380,SPACE=(TRK,(15,2)),VOL=SER=XYZ003,
//     DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//SORTOUT  DD  DSNAME=SMITH.OUTPUT,DISP=(NEW,KEEP),
//     UNIT=3380,SPACE=(TRK,(15,2)),VOL=SER=XYZ003
//SYSIN    DD  *
    SORT FIELDS=(5,20,CH,A,13,5,BI,D)
    OPTION STOPAFT=30,DYNALLOC=3330
/*
```

DFSORT Initialization Phase:

Major Call 1

Prior to Major Call 1, DFSORT sets the following fields in the EFS interface parameter list:

- Action code=0

  Major Call 1 is in effect.

- Informational bit flag 2=0

  The DFSORT run is JCL-invoked.

- Informational bit flag 3=0

  SORTDIAG is not in effect.

DFSORT executes Major Call 1 to EFS program EFSPGM and EFSPGM sets the following fields in the EFS interface parameter list:

- Control Statement Request List

  Contains the OPTION operation definer which indicates to DFSORT that the OPTION control statement is requested by EFSPGM.

- EFSPGM Program Context Area

  EFSPGM will be using the context area.

- Message List=0

  EFSPGM has no messages for DFSORT to print to the message data set. General register 15 is set to zero.

Major Call 2

Prior to Major Call 2, DFSORT sets the following fields in the EFS interface parameter list:

- Action code=4

  Major Call 2 is in effect.

- Informational bit flag 4=0 and informational bit flag 5=0

  No application is in effect.

EFSPGM requested the OPTION control statement. DFSORT makes a call to an EFS program for each control statement requested; in this case, one. DFSORT also sets the following fields in the EFS interface parameter list:

- Informational bit flag 0=0 and informational bit flag 1=1

  The requested control statement came from SYSIN.

- The ***original*** OPTION control statement, including all operands and corresponding subparameters

  ```
  OPTION STOPAFT=30,DYNALLOC=3330
  ```

- The length of the *original* OPTION control statement, including all operands and corresponding subparameters

  The original control statement string is 31 bytes long.

DFSORT executes Major Call 2 to EFS program EFSPGM and EFSPGM sets the following fields in the EFS interface parameter list:

- Informational bit flag 8=1

  DFSORT must parse the control statement returned by EFSPGM.

- The *altered* OPTION control statement, including all operands and subparameters

  ```
  OPTION STOPAFT=30,DYNALLOC=3380,EQUALS
  ```

- The length of the *altered* OPTION control statement, including all operands and subparameters

  The altered control statement string is 38 bytes long.

- Message List=0

  EFSPGM has no messages for DFSORT to print to the message data set. General register 15 is set to zero.

Figure 60 on page 350 shows the original control statement sent to EFS program EFSPGM and the altered control statement returned by EFS program EFSPGM.

Original OPTION control statement sent to EFSPGM

```
    OPTION STOPAFT=30,DYNALLOC=3330
```

Altered OPTION control statement returned by EFSPGM

```
    OPTION STOPAFT=30,DYNALLOC=3380,EQUALS
```

where:

STOPAFT=30 is the original operand and value
DYNALLOC=3380 is the original operand with a new value
EQUALS option has been added

Figure 60. Original and Altered Control Statements

Major Call 3

Prior to Major Call 3, DFSORT sets the following fields in the EFS interface parameter list:

- Action Code=8

  Major Call 3 is in effect.

- Informational bit flag 4=0 and informational bit flag 5=1

  A sort application is in effect.

- Informational bit flag 7=0

  Fixed-length records are being processed.

- Record Lengths List values=80

  The LRECL of the input and output data sets is 80. Because the SORTOUT LRECL was not specified, DFSORT defaulted to the SORTIN LRECL for the SORTOUT LRECL.

- Extract Buffer Offsets List=0

  No EFS control fields were specified on the SORT control statement.

DFSORT executes Major Call 3 to EFS program EFSPGM, and EFSPGM sets the following fields in the EFS interface parameter list:

- EFS01 address=0

  Because the SORT control statement has no EFS control fields, the EFS01 exit routine is not used.

- EFS02 address=0

  Because no INCLUDE control statement was supplied (with EFS fields), the EFS02 exit routine is not used.

- Message List=0

  EFSPGM has no messages for DFSORT to print to the message data set. General register 15 is set to zero.

DFSORT Termination Phase

Major Call 4

Prior to Major Call 4, DFSORT sets the following fields in the EFS interface parameter list:

- Action Code=12

  Major Call 4 is in effect.

DFSORT executes Major Call 4 to EFS program EFSPGM and EFSPGM sets the following fields in the EFS interface parameter list:

- Message List=0

  EFSPGM has no messages for DFSORT to print to the message data set.

And general register 15 is set to zero.

Major Call 5

Prior to Major Call 5, DFSORT sets the following fields in the EFS interface parameter list:

- Action Code=16

  Major Call 5 is in effect.

DFSORT executes Major Call 5 to EFS program EFSPGM and EFSPGM does not set any fields in the EFS interface parameter list but sets general register 15 to zero.

Appendix C. Specification/Override of DFSORT Options

Listed below are the places in DFSORT where you can specify various options that will override the IBM-supplied standard defaults. 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

- DFSPARM data set
  - PARM options
  - DEBUG and OPTION control statements
  - Other control statements.
- EXEC statement PARM options
- SYSIN data set
  - DEBUG and OPTION control statements
  - Other control statements.
- Installation macro (ICEMAC JCL).

Dynamically Invoked DFSORT

- DFSPARM data set
  - PARM options
  - DEBUG and OPTION control statements
  - Other control statements.
- 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).

Note: An EFS program or an installation initialization exit (ICEIEXIT) routine can also be used to override options. ICEIEXIT changes override any corresponding changes made by an EFS program.

Main Features of Sources of DFSORT Options

There are five sources of options in which you can override IBM-supplied standard defaults. To help you decide which is most efficient for you, compare their main features using the following lists:

DFSPARM Data Set

- Use with JCL or dynamic invocation.
- Overrides all other sources.
- Accepts all DFSORT program control statements, and all EXEC PARM options, including those OPTION statement parameters ignored by SYSIN and SORTCNTL.
- Permits comment statements, blank statements, and remarks.

EXEC Statement PARM Options

- Use with JCL invocation only.
- Accepts all EXEC PARM options, including those equivalent to the OPTION statement parameters ignored by SYSIN and SORTCNTL.

SORTCNTL Data Set

- Use with dynamic invocation only.
- Accepts all DFSORT program control statements.
- Ignores these OPTION statement parameters: EFS, LIST, NOLIST, LISTX, NOLISTX, MSGPRT, MSGDDN, SORTDD, SORTIN, and SORTOUT.
- Permits comment statements, blank statements, and remarks.
- Using multiple parameter lists to rename the SORTCNTL data set permits different control statements to be used for a program that invokes DFSORT more than once.

SYSIN Data Set

- Use with JCL invocation only.
- Accepts all DFSORT program control statements.
- Ignores these OPTION statement parameters: EFS, LIST, NOLIST, LISTX, NOLISTX, MSGPRT, MSGDDN, SORTDD, SORTIN, and SORTOUT.
- Permits comment statements, blank statements, and remarks.
- Can contain user exit routines in object deck format for link-editing.

Parameter lists

- Use with dynamic invocation only.
- Extended parameter list accepts all DFSORT program control statements, including those OPTION statement parameters ignored by SYSIN and SORTCNTL.
- 24-bit parameter list accepts a subset of DFSORT program control statements.
- Using multiple parameter lists to rename the SORTCNTL data set permits different control statements to be used for a program that invokes DFSORT more than once.
- Can be used to pass the addresses of any exits that your program has placed in main storage.

Note: The extended parameter list can perform a superset of the functions in the 24-bit parameter list.

Override Tables

The following tables show the possible sources of specification and order of override for individual options.

- The order of override between sources of specification is from left to right. A specification overrides all specifications to its right.
- The order of override within a source is from top to bottom. A specification overrides all specifications below it.
- EXEC PARM options you can specify in the DFSPARM data set are preceded by the word "PARM" in the DFSPARM columns of the tables to distinguish them from control statement options.
- The Function columns indicate which functions (S=sort, M=merge, or C=copy) can use the option.

- Although alias names are available for many of the options, they are not shown here.

JCL-Invoked DFSORT

Table 22 shows where each sort, merge, or copy option may be specified when DFSORT is invoked through JCL.

Table 22 (Page 1 of 4). JCL DFSORT Option Specification/Override

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL	Description of Option	Function
NO	NO	NO	ABCODE	ABEND code	S,M,C
ALTSEQ CODE	NO	ALTSEQ CODE	ALTSEQ	Alternate sequence	S,M
PARM ARESALL OPTION ARESALL	ARESALL	OPTION ARESALL	ARESALL	System storage above 16-megabyte virtual	S
PARM BSAM DEBUG BSAM	BSAM	DEBUG BSAM	NO	Force BSAM	S,M,C
DEBUG NOASSIST	NO	DEBUG NOASSIST	NO	Bypass Sorting Instructions	S
DEBUG BUFFERS	NO	DEBUG BUFFERS	NO	Placement of buffers	S
OPTION CHALT|NOCHALT	NO	OPTION CHALT|NOCHALT	CHALT	CH field sequence	S,M
OPTION CHECK|NOCHECK	NO	OPTION CHECK|NOCHECK	CHECK	Record count check	S,M,C
PARM CINV|NOCINV OPTION CINV|NOCINV	CINV|NOCINV	OPTION CINV|NOCINV	CINV	Control interval access	S,M,C
OPTION COBEXIT	NO	OPTION COBEXIT	COBEXIT	COBOL library	S,M,C
DEBUG CTRx	NO	DEBUG CTRx	NO	ABEND record count	S,M
DEBUG ABSTP	NO	DEBUG ABSTP	NO	Abnormal stop	S,M,C
DEBUG ESTAE|NOESTAE	NO	DEBUG ESTAE|NOESTAE	ESTAE	ESTAE routine	S,M,C
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	DYNALOC[1]	Dynamic SORTWKs	S
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	DYNAUTO	Automatic DYNALLOC	S
PARM EFS OPTION EFS	EFS	NO[2]	EFS	EFS program specified	S,M,C
PARM EQUALS|NOEQUALS OPTION EQUALS|NOEQUALS SORT|MERGE EQUALS|NOEQUALS	EQUALS|NOEQUALS	OPTION EQUALS|NOEQUALS SORT|MERGE EQUALS|NOEQUALS	EQUALS	Equal record order	S,M
DEBUG EQUCOUNT	NO	DEBUG EQUCOUNT	NO	Equal key count message	S
PARM EXCPVR OPTION EXCPVR	EXCPVR	OPTION EXCPVR	EXCPVR	EXCPVR for I/O	S,C
PARM ABEND|NOABEND DEBUG ABEND|NOABEND	ABEND|NOABEND	DEBUG ABEND|NOABEND	ERET	Error action	S,M,C

Table 22 (Page 2 of 4). JCL DFSORT Option Specification/Override

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL	Description of Option	Function
NO	NO	NO	EXITCK	E15/E35 return code checking	S,M,C
INCLUDE|OMIT COND/FORMAT	NO	INCLUDE|OMIT COND/FORMAT	NO	Include|Omit fields	S,M,C
PARM E15=COB PARM E35=COB MODS Exx	E15=COB E35=COB	MODS Exx	NO	Exit Exx (xx=11,15-19,31,35,37-39, and 61)	S,M,C[3]
NO	NO	NO	GENER	IEBGENER name	C
PARM HIPRMAX OPTION HIPRMAX	HIPRMAX	OPTION HIPRMAX	HIPRMAX	Hipersorting	S
NO	NO	NO	IEXIT	ICEIEXIT	S,M,C
INREC FIELDS	NO	INREC FIELDS	NO	INREC fields	S,M,C
OUTREC FIELDS	NO	OUTREC FIELDS	NO	OUTREC fields	S,M,C
SORT|MERGE| FIELDS|FORMAT	NO	SORT|MERGE FIELDS|FORMAT	NO	Control fields	S,M
OPTION COPY SORT|MERGE FIELDS	NO	OPTION COPY SORT|MERGE FIELDS	NO	Copy records	C
SUM FIELDS/FORMAT	NO	SUM FIELDS/FORMAT	NO	Sum fields	S,M
MERGE FILES	NO	MERGE FILES	NO	Merge input files	M
PARM FILSZ OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	FILSZ	OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	NO	File size	S,M
OPTION CKPT[4] SORT|MERGE CKPT[4]	NO	OPTION CKPT[4] SORT|MERGE CKPT[4]	IGNCKPT	Checkpoints	S,M
RECORD LENGTH	NO	RECORD LENGTH	NO	Record lengths	S,M,C
PARM LIST|NOLIST OPTION LIST|NOLIST	LIST|NOLIST	NO[2]	LIST	Print DFSORT control statements[5]	S,M,C
PARM LISTX|NOLISTX OPTION LISTX|NOLISTX	LISTX|NOLISTX	NO[2]	LISTX	Print control statements returned by an EFS program[5]	S,M,C
NO	NO	NO	MAXLIM	Maximum storage below 16-megabyte virtual[6]	S,M,C
NO	NO	NO	MINLIM	Minimum storage	S,M,C
PARM MSGDDN OPTION MSGDDN	MSGDDN	NO[2]	MSGDDN	Alternate message data set	S,M,C
NO	NO	NO	MSGCON	Write messages on master console	S,M,C

Table 22 (Page 3 of 4). JCL DFSORT Option Specification/Override

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL	Description of Option	Function
PARM MSGPRT OPTION MSGPRT	MSGPRT|FLAG	NO[2]	MSGPRT	Print messages	S,M,C
OPTION NOBLKSET	NO	OPTION NOBLKSET	NO	Bypass Blockset	S,M
NO	NO	NO	NOMSGDD	Action when message data set missing	S,M,C
PARM OUTREL|NOOUTREL OPTION NOOUTREL	OUTREL|NOOUTREL	OPTION NOOUTREL	OUTREL	Release SORTOUT space	S,M,C
OPTION NOOUTSEC	NO	OPTION NOOUTSEC	OUTSEC	SORTOUT secondary allocation	S,M,C
NO	NO	NO	OVERRGN	Storage over REGION	S,M,C
NO	NO	NO	PARMDDN	Alternate DDNAME for DFSPARM	S,M,C
PARM RESALL OPTION RESALL	RESALL	OPTION RESALL	RESALL	System reserved storage[6]	S,M,C
PARM SIZE OPTION MAINSIZE	SIZE	OPTION MAINSIZE	SIZE	Storage	S,M,C
NO	NO	NO	SMF	SMF records	S,M,C
PARM SKIPREC OPTION SKIPREC SORT|MERGE SKIPREC	SKIPREC	OPTION SKIPREC SORT|MERGE SKIPREC	NO	Skip records	S,C
OPTION SORTDD	NO	NO[2]	NO	DDNAME prefix	S,M,C
OPTION SORTIN[7]	NO	NO[2]	NO	Alternate input DDNAME	S,C
NO	NO	NO	SORTLIB	Conventional module library	S,M
OPTION SORTOUT[8]	NO	NO[2]	NO	Alternate output DDNAME	S,M,C
PARM STIMER|NOSTIMER OPTION NOSTIMER	STIMER|NOSTIMER	OPTION NOSTIMER	STIMER	Use of STIMER	S,M,C
PARM STOPAFT OPTION STOPAFT SORT|MERGE STOPAFT	STOPAFT	OPTION STOPAFT SORT|MERGE STOPAFT	NO	Input limit	S,C
NO	NO	NO	SVC	DFSORT SVC Information	S,M,C
NO	NO	NO	TEXIT	ICETEXIT	S,M,C
RECORD TYPE	NO	RECORD TYPE	NO	Record format	S,M,C
NO	NO	NO	TMAXLIM	Maximum storage above and below 16-megabyte virtual[6]	S

Table 22 (Page 4 of 4). JCL DFSORT Option Specification/Override

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL	Description of Option	Function
OPTION VERIFY|NOVERIFY	NO	OPTION VERIFY|NOVERIFY	VERIFY	Sequence check	S,M
NO	NO	NO	VIO	SORTWK virtual I/O	S
OPTION VLSHRT|NOVLSHRT	NO	OPTION VLSHRT|NOVLSHRT	VLSHRT	Variable records do not contain all specified control fields	S,M
PARM WRKREL|NOWRKREL OPTION NOWRKREL	WRKREL|NOWRKREL	OPTION NOWRKREL	WRKREL	Release SORTWK space	S
PARM WRKSEC|NOWRKSEC OPTION NOWRKSEC	WRKSEC|NOWRKSEC	OPTION NOWRKSEC	WRKSEC	SORTWK secondary allocation	S
NO	NO	NO	ZDPRINT	ZD SUM results	S,M

Notes to Table 22 on page 356:

1	Does not request dynamic allocation; only supplies defaults.
2	Not used in SYSIN.
3	All functions do not apply to all exits. See Figure 28 on page 186 and Figure 29 on page 187 for applicable exits.
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	Not used unless MAINSIZE=MAX is in effect.
7	Overrides SORTDD for the SORT input DDNAME.
8	Overrides SORTDD for the SORT output DDNAME.

Dynamically-Invoked DFSORT with the Extended Parameter List

Table 23 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.

Control statements (other than DEBUG and OPTION) you specify in DFSPARM completely override corresponding control statements specified in any other source. DEBUG, OPTION, and EXEC PARM options selectively override corresponding options in control statements specified in any other source.

Control statements (other than DEBUG and OPTION) specified in the SORTCNTL data set completely override corresponding control statements specified through the extended parameter list. DEBUG and OPTION options selectively override corresponding options in control statements specified in any other source.

SORT and MERGE and INCLUDE and OMIT, are considered to be corresponding control statements.

Table 23 (Page 1 of 5). Extended Parameter List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV	Description of Option	Function
NO	NO	NO	ABCODE	ABEND code	S,M,C
ALTSEQ CODE	ALTSEQ CODE	Offset 16 entry ALTSEQ CODE	ALTSEQ	Alternate sequence	S,M
PARM ARESALL OPTION ARESALL	OPTION ARESALL	OPTION ARESALL	ARESALL	System storage above 16-megabyte virtual	S
OPTION ARESINV	OPTION ARESINV	OPTION ARESINV	ARESINV	Storage above 16-megabyte virtual for invoking program	S
PARM BSAM DEBUG BSAM	DEBUG BSAM	DEBUG BSAM	NO	Force BSAM	S,M,C
DEBUG NOASSIST	DEBUG NOASSIST	DEBUG NOASSIST	NO	Bypass Sorting Instructions	S
DEBUG BUFFERS	DEBUG BUFFERS	DEBUG BUFFERS	NO	Placement of buffers	S
OPTION CHALT|NOCHALT	OPTION CHALT|NOCHALT	OPTION CHALT|NOCHALT	CHALT	CH field sequence	S,M
OPTION CHECK|NOCHECK	OPTION CHECK|NOCHECK	OPTION CHECK|NOCHECK	CHECK	Record count check	S,M,C
PARM CINV|NOCINV OPTION CINV|NOCINV	OPTION CINV|NOCINV	OPTION CINV|NOCINV	CINV	Control interval access	S,M,C
OPTION COBEXIT	OPTION COBEXIT	OPTION COBEXIT	COBEXIT	COBOL library	S,M,C
DEBUG CTRx	DEBUG CTRx	DEBUG CTRx	NO	ABEND record count	S,M
DEBUG ABSTP	DEBUG ABSTP	DEBUG ABSTP	NO	Abnormal stop	S,M,C
DEBUG ESTAE|NOESTAE	DEBUG ESTAE|NOESTAE	DEBUG ESTAE|NOESTAE	ESTAE	ESTAE routine	S,M,C
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC[2]	OPTION DYNALLOC SORT DYNALLOC	DYNALOC[1]	Dynamic SORTWKs	S
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	DYNAUTO	Automatic DYNALLOC	S
PARM EFS OPTION EFS	NO[3]	OPTION EFS	EFS	EFS program specified	S,M,C
PARM EQUALS|NOEQUALS OPTION EQUALS|NOEQUALS SORT|MERGE EQUALS|NOEQUALS	OPTION EQUALS|NOEQUALS SORT|MERGE EQUALS|NOEQUALS[2]	OPTION EQUALS|NOEQUALS SORT|MERGE EQUALS|NOEQUALS	EQUALS	Equal record order	S,M

Table 23 (Page 2 of 5). Extended Parameter List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV	Description of Option	Function
DEBUG EQUCOUNT	DEBUG EQUCOUNT	DEBUG EQUCOUNT	NO	Equal key count message	S
PARM EXCPVR OPTION EXCPVR	OPTION EXCPVR	OPTION EXCPVR	EXCPVR	EXCPVR for I/O	S,C
PARM ABEND|NOABEND DEBUG ABEND|NOABEND	DEBUG ABEND|NOABEND	DEBUG ABEND|NOABEND	ERET	Error action	S,M,C
NO	NO	NO	EXITCK	E15/E35 return code checking	S,M,C
INCLUDE|OMIT COND/FORMAT	INCLUDE|OMIT COND/FORMAT	INCLUDE|OMIT COND/FORMAT	NO	Include|Omit fields	S,M,C
PARM E15=COB MODS E15[4]	MODS E15[4]	Offset 4 entry[4] MODS E15[4]	NO	Exit E15	S,C
MODS E18[4]	MODS E18[4]	Offset 24 entry[4] MODS E18[4]	NO	Exit E18	S
NO	NO	Offset 4 entry	NO	Exit E32	M
PARM E35=COB MODS E35[4]	MODS E35[4]	Offset 8 entry[4] MODS E35[4]	NO	Exit E35	S,M,C
MODS E39[4]	MODS E39[4]	Offset 28 entry[4] MODS E39[4]	NO	Exit E39	S,M,C
MODS Exx	MODS Exx	MODS Exx	NO	Exit Exx (xx=11,16,17,19, 31,37,38, and 61)	S,M,C[5]
NO	NO	NO	GENER	IEBGENER name	C
PARM HIPRMAX OPTION HIPRMAX	OPTION HIPRMAX	OPTION HIPRMAX	HIPRMAX	Hipersorting	S
NO	NO	NO	IEXIT	ICEIEXIT	S,M,C
INREC FIELDS	INREC FIELDS	INREC FIELDS	NO	INREC fields	S,M,C
OUTREC FIELDS	OUTREC FIELDS	OUTREC FIELDS	NO	OUTREC fields	S,M,C
SORT|MERGE FIELDS/FORMAT	SORT|MERGE FIELDS/FORMAT	SORT|MERGE FIELDS/FORMAT	NO	Control fields	S,M
OPTION COPY SORT|MERGE FIELDS	OPTION COPY SORT|MERGE FIELDS[2]	OPTION COPY SORT|MERGE FIELDS	NO	Copy records	C
SUM FIELDS/FORMAT	SUM FIELDS/FORMAT	SUM FIELDS/FORMAT	NO	Sum fields	S,M
MERGE FILES	MERGE FILES	MERGE FILES	NO	Merge input files	M

Table 23 (Page 3 of 5). Extended Parameter List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV	Description of Option	Function
PARM FILSZ OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE[2]	OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	NO	File size	S,M
OPTION CKPT[6] SORT|MERGE CKPT[6]	OPTION CKPT[6] SORT|MERGE CKPT[2,6]	OPTION CKPT[6] SORT|MERGE CKPT[6]	IGNCKPT	Checkpoints	S,M
RECORD LENGTH	RECORD LENGTH	RECORD LENGTH	NO	Record lengths	S,M,C
PARM LIST|NOLIST OPTION LIST|NOLIST	NO[3]	OPTION LIST|NOLIST	LIST	Print DFSORT control statements[7]	S,M,C
PARM LISTX|NOLISTX OPTION LISTX|NOLISTX	NO[3]	OPTION LISTX|NOLISTX	LISTX	Print control statements returned by an EFS program[7]	S,M,C
NO	NO	NO	MAXLIM	Maximum storage below 16-megabyte virtual[8]	S,M,C
NO	NO	NO	MINLIM	Minimum storage	S,M,C
PARM MSGDDN OPTION MSGDDN	NO[3]	OPTION MSGDDN	MSGDDN	Alternate message DDNAME	S,M,C
NO	NO	NO	MSGCON	Write messages on master console	S,M,C
PARM MSGPRT OPTION MSGPRT	NO[3]	OPTION MSGPRT	MSGPRT	Print messages	S,M,C
OPTION NOBLKSET	OPTION NOBLKSET	OPTION NOBLKSET	NO	Bypass Blockset	S,M
NO	NO	NO	NOMSGDD	Action when message data set missing	S,M,C
PARM OUTREL|NOOUTREL OPTION NOOUTREL	OPTION NOOUTREL	OPTION NOOUTREL	OUTREL	Release SORTOUT space	S,M,C
OPTION NOOUTSEC	OPTION NOOUTSEC	OPTION NOOUTSEC	OUTSEC	SORTOUT secondary allocation	S,M,C
NO	NO	NO	OVERRGN	Storage over REGION	S,M,C
NO	NO	NO	PARMDDN	Alternate DDNAME for DFSPARM	S,M,C
PARM RESALL OPTION RESALL	OPTION RESALL	OPTION RESALL	RESALL	System reserved storage[8]	S,M,C

Table 23 (Page 4 of 5). Extended Parameter List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV	Description of Option	Function
OPTION RESINV	OPTION RESINV	OPTION RESINV	RESINV	Program reserved storage[8]	S,M,C
PARM SIZE OPTION MAINSIZE	OPTION MAINSIZE	OPTION MAINSIZE	SIZE	Storage	S,M,C
NO	NO	NO	SMF	SMF records	S,M,C
PARM SKIPREC OPTION SKIPREC SORT|MERGE SKIPREC	OPTION SKIPREC SORT|MERGE SKIPREC[2]	OPTION SKIPREC SORT|MERGE SKIPREC	NO	Skip records	S,C
OPTION SORTDD	NO[3]	OPTION SORTDD	NO	DDNAME prefix	S,M,C
OPTION SORTIN[9]	NO[3]	OPTION SORTIN[9]	NO	Alternate input DDNAME	S,C
NO	NO	NO	SORTLIB	Conventional module library	S,M
OPTION SORTOUT[10]	NO[3]	OPTION SORTOUT[10]	NO	Alternate output DDNAME	S,M,C
PARM STIMER|NOSTIMER OPTION NOSTIMER	OPTION NOSTIMER	OPTION NOSTIMER	STIMER	Use of STIMER	S,M,C
PARM STOPAFT OPTION STOPAFT SORT|MERGE STOPAFT	OPTION STOPAFT SORT|MERGE STOPAFT[2]	OPTION STOPAFT SORT|MERGE STOPAFT	NO	Input limit	S,C
NO	NO	NO	SVC	DFSORT SVC information	S,M,C
NO	NO	NO	TEXIT	ICETEXIT	S,M,C
RECORD TYPE	RECORD TYPE	RECORD TYPE	NO	Record format	S,M,C
NO	NO	NO	TMAXLIM	Maximum storage above and below 16-megabyte virtual[8]	S
OPTION VERIFY|NOVERIFY	OPTION VERIFY|NOVERIFY	OPTION VERIFY|NOVERIFY	VERIFY	Sequence check	S,M
NO	NO	NO	VIO	SORTWK virtual I/O	S
OPTION VLSHRT|NOVLSHRT	OPTION VLSHRT|NOVLSHRT	OPTION VLSHRT|NOVLSHRT	VLSHRT	Variable records do not contain all specified control fields	S,M
PARM WRKREL|NOWRKREL OPTION NOWRKREL	OPTION NOWRKREL	OPTION NOWRKREL	WRKREL	Release SORTWK space	S

Table 23 (Page 5 of 5). Extended Parameter List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV	Description of Option	Function
PARM WRKSEC|NOWRKSEC OPTION NOWRKSEC	OPTION NOWRKSEC	OPTION NOWRKSEC	WRKSEC	SORTWK secondary allocation	S
NO	NO	NO	ZDPRINT	ZD SUM results	S,M

Notes to Table 23 on page 361:

[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] Not used in SORTCNTL.

[4] DFSORT terminates if the exit is specified via the parameter list entry and the exit is specified in a MODS statement.

[5] All functions do not apply to all exits. See Figure 28 on page 186 and Figure 29 on page 187 for applicable exits.

[6] Not used if Blockset is selected and IGNCKPT=YES was specified.

[7] Not used if MSGPRT=NONE is in effect; in this case control statements are not printed.

[8] Not used unless MAINSIZE=MAX is in effect.

[9] Overrides SORTDD for the sort input DDNAME.

[10] Overrides SORTDD for the sort output DDNAME.

Dynamically-Invoked DFSORT with the 24-Bit Parameter List

Table 24 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.

Control statements (other than DEBUG and OPTION) you specify in DFSPARM completely override corresponding control statements specified in any other source. DEBUG, OPTION, and EXEC PARM options selectively override corresponding options in control statements in any other source.

Control statements (other than DEBUG) specified in the SORTCNTL data set completely override corresponding control statements specified through the 24-bit parameter list. DEBUG options selectively override corresponding options in control statements specified in any other source.

SORT and MERGE and INCLUDE and OMIT, are considered to be corresponding control statements.

Table 24 (Page 1 of 4). 24-Bit List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV	Description of Option	Function
NO	NO	NO	ABCODE	ABEND code	S,M,C
ALTSEQ CODE	ALTSEQ CODE	X'F6' entry ALTSEQ CODE	ALTSEQ	Alternate sequence	S,M
PARM ARESALL OPTION ARESALL	OPTION ARESALL	NO	ARESALL	System storage above 16-megabyte virtual	S
OPTION ARESINV	OPTION ARESINV	NO	ARESINV	Storage above 16-megabyte virtual for invoking program	S
PARM BSAM DEBUG BSAM	DEBUG BSAM	DEBUG BSAM	NO	Force BSAM	S,M,C
DEBUG NOASSIST	DEBUG NOASSIST	DEBUG NOASSIST	NO	Bypass Sorting Instructions	S
DEBUG BUFFERS	DEBUG BUFFERS	DEBUG BUFFERS	NO	Placement of buffers	S
OPTION CHALT|NOCHALT	OPTION CHALT|NOCHALT	NO	CHALT	CH field sequence	S,M
OPTION CHECK|NOCHECK	OPTION CHECK|NOCHECK	NO	CHECK	Record count check	S,M,C
PARM CINV|NOCINV OPTION CINV|NOCINV	OPTION CINV|NOCINV	NO	CINV	Control interval access	S,M,C
OPTION COBEXIT	OPTION COBEXIT	NO	COBEXIT	COBOL library	S,M,C
DEBUG CTRx	DEBUG CTRx	DEBUG CTRx	NO	ABEND record count	S,M
DEBUG ABSTP	DEBUG ABSTP	DEBUG ABSTP	NO	Abnormal stop	S,M,C
DEBUG ESTAE|NOESTAE	DEBUG ESTAE|NOESTAE	DEBUG ESTAE|NOESTAE	ESTAE	ESTAE routine	S,M,C
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	SORT DYNALLOC	DYNALOC[1]	Dynamic SORTWKs	S
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	SORT DYNALLOC	DYNAUTO	Automatic DYNALLOC	S
PARM EFS OPTION EFS	NO[2]	NO	EFS	EFS program specified	S,M,C
PARM EQUALS|NOEQUALS OPTION EQUALS|NOEQUALS SORT|MERGE EQUALS|NOEQUALS	OPTION EQUALS|NOEQUALS SORT|MERGE EQUALS|NOEQUALS	SORT|MERGE EQUALS|NOEQUALS	EQUALS	Equal record order	S,M

Table 24 (Page 2 of 4). 24-Bit List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV	Description of Option	Function
DEBUG EQUCOUNT	DEBUG EQUCOUNT	DEBUG EQUCOUNT	NO	Equal key count message	S
PARM EXCPVR OPTION EXCPVR	OPTION EXCPVR	NO	EXCPVR	EXCPVR for I/O	S,C
PARM ABEND|NOABEND DEBUG ABEND|NOABEND	DEBUG ABEND|NOABEND	DEBUG ABEND|NOABEND	ERET	Error action	S,M,C
NO	NO	NO	EXITCK	E15/E35 return code checking	S,M,C
INCLUDE|OMIT COND/FORMAT	INCLUDE|OMIT COND/FORMAT	NO	NO	Include|Omit fields	S,M,C
PARM E15=COB MODS E15[3]	MODS E15[3]	Offset 18 entry[3] MODS E15[3]	NO	Exit E15	S,C
NO	NO	Offset 18 entry	NO	Exit E32	M
PARM E35=COB MODS E35[3]	MODS E35[3]	Offset 22 entry[3] MODS E35[3]	NO	Exit E35	S,M,C
MODS Exx	MODS Exx	MODS Exx	NO	Exit Exx (xx=11,16-19, 31,37-39, and 61)	S,M,C[4]
PARM HIPRMAX OPTION HIPRMAX	OPTION HIPRMAX	NO	HIPRMAX	Hipersorting	S
NO	NO	NO	GENER	IEBGENER name	C
NO	NO	NO	IEXIT	ICEIEXIT	S,M,C
INREC FIELDS	INREC FIELDS	NO	NO	INREC fields	S,M,C
OUTREC FIELDS	OUTREC FIELDS	NO	NO	OUTREC fields	S,M,C
SORT|MERGE FIELDS/FORMAT	SORT|MERGE FIELDS/FORMAT	SORT|MERGE FIELDS/FORMAT	NO	Control fields	S,M,C
OPTION COPY SORT|MERGE FIELDS	OPTION COPY SORT|MERGE FIELDS	SORT|MERGE FIELDS	NO	Copy records	C
SUM FIELDS/FORMAT	SUM FIELDS/FORMAT	NO	NO	Sum fields	S,M
MERGE FILES	MERGE FILES	X'04' entry MERGE FILES	NO	Merge input files	M
PARM FILSZ OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	SORT|MERGE FILSZ|SIZE	NO	File size	S,M
OPTION CKPT[5] SORT|MERGE CKPT[5]	OPTION CKPT[5] SORT|MERGE CKPT[5]	SORT|MERGE CKPT[5]	IGNCKPT	Checkpoints	S,M

Table 24 (Page 3 of 4). 24-Bit List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV	Description of Option	Function
RECORD LENGTH	RECORD LENGTH	RECORD LENGTH	NO	Record lengths	S,M,C
PARM LIST|NOLIST OPTION LIST|NOLIST	NO[2]	NO	LIST	Print DFSORT control statements[6]	S,M,C
PARM LISTX|NOLISTX OPTION LISTX|NOLISTX	NO[2]	NO	LISTX	Print control statements returned by an EFS program[6]	S,M,C
NO	NO	NO	MAXLIM	Maximum storage below 16-megabyte virtual[7]	S,M,C
NO	NO	NO	MINLIM	Minimum storage	S,M,C
PARM MSGDDN OPTION MSGDDN	NO[2]	X'03' entry	MSGDDN	Alternate message DDNAME	S,M,C
NO	NO	NO	MSGCON	Write messages on master console	S,M,C
PARM MSGPRT OPTION MSGPRT	NO[2]	X'FF' entry	MSGPRT	Print messages	S,M,C
OPTION NOBLKSET	OPTION NOBLKSET	NO	NO	Bypass Blockset	S,M
NO	NO	NO	NOMSGDD	Action when message data set missing	S,M,C
PARM OUTREL|NOOUTREL OPTION NOOUTREL	OPTION NOOUTREL	NO	OUTREL	Release SORTOUT space	S,M,C
OPTION NOOUTSEC	OPTION NOOUTSEC	NO	OUTSEC	SORTOUT secondary allocation	S,M,C
NO	NO	NO	OVERRGN	Storage over REGION	S,M,C
NO	NO	NO	PARMDDN	Alternate DDNAME for DFSPARM	S,M,C
PARM RESALL OPTION RESALL	OPTION RESALL	NO	RESALL	System reserved storage[7]	S,M,C
OPTION RESINV	OPTION RESINV	X'01' entry	RESINV	Program reserved storage[7]	S,M,C
PARM SIZE OPTION MAINSIZE	OPTION MAINSIZE	X'00' entry	SIZE	Storage	S,M,C
NO	NO	NO	SMF	SMF records	S,M,C

Table 24 (Page 4 of 4). 24-Bit List DFSORT Option Specification/Override

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV	Description of Option	Function
PARM SKIPREC OPTION SKIPREC SORT|MERGE SKIPREC	OPTION SKIPREC SORT|MERGE SKIPREC	SORT|MERGE SKIPREC	NO	Skip records	S,C
OPTION SORTDD	NO[2]	Prefix entry	NO	DDNAME prefix	S,M,C
OPTION SORTIN[8]	NO[2]	NO	NO	Alternate input DDNAME	S,C
NO	NO	NO	SORTLIB	Conventional module library	S,M
OPTION SORTOUT[9]	NO[2]	NO	NO	Alternate output DDNAME	S,M,C
PARM STIMER|NOSTIMER OPTION NOSTIMER	OPTION NOSTIMER	NO	STIMER	Use of STIMER	S,M,C
PARM STOPAFT SORT|MERGE STOPAFT OPTION STOPAFT	OPTION STOPAFT SORT|MERGE STOPAFT	SORT|MERGE STOPAFT	NO	Input limit	S,C
NO	NO	NO	SVC	DFSORT SVC information	S,M,C
NO	NO	NO	TEXIT	ICETEXIT	S,M,C
RECORD TYPE	RECORD TYPE	RECORD TYPE	NO	Record format	S,M,C
NO	NO	NO	TMAXLIM	Maximum storage above and below 16-megabyte virtual[7]	S
OPTION VERIFY|NOVERIFY	OPTION VERIFY|NOVERIFY	NO	VERIFY	Sequence check	S,M
NO	NO	NO	VIO	SORTWK virtual I/O	S
OPTION VLSHRT|NOVLSHRT	OPTION VLSHRT|NOVLSHRT	X'FD' entry	VLSHRT	Variable records do not contain all specified control fields	S,M
PARM WRKREL|NOWRKREL OPTION NOWRKREL	OPTION NOWRKREL	NO	WRKREL	Release SORTWK space	S
PARM WRKSEC|NOWRKSEC OPTION NOWRKSEC	OPTION NOWRKSEC	NO	WRKSEC	SORTWK secondary allocation	S
NO	NO	NO	ZDPRINT	ZD SUM results	S,M

Notes to Table 24 on page 367:

[1] Does not request dynamic allocation; only supplies defaults.

[2] Not used in SORTCNTL.

[3] DFSORT terminates if the exit is specified via the parameter list entry and the exit is specified in a MODS statement.

[4] All functions do not apply to all exits. See Figure 28 on page 186 and Figure 29 on page 187 for applicable exits.

[5] Not used if Blockset is selected and IGNCKPT=YES was specified.

[6] Not used if MSGPRT=NONE or MSGPRT=CRITICAL is in effect; in this case control statements are not printed.

[7] Not used unless MAINSIZE=MAX is in effect.

[8] Overrides SORTDD for the sort input DDNAME.

[9] Overrides SORTDD for the sort output DDNAME.

Appendix D. Data Format Examples

The format descriptions refer to the assembled data formats as used with IBM System/370. If, for example, a data variable is declared in PL/I as FIXED DECIMAL, it is the compiled format of the variable that must be given in the 'f' field of the SORT control statement, not the PL/I declared format. In this case, the 'f' field would be PD (packed decimal) because the PL/I compiler converts fixed decimal to packed decimal form.

Format	Description
CH	(character EBCDIC, unsigned). Each character is represented by its 8-bit EBCDIC code. Example: AB7 becomes C1 C2 F7 Hexadecimal 11000001 11000010 11110111 Binary
ZD	(zoned decimal, signed). Each digit of the decimal number is converted into its 8-bit EBCDIC representation. The sign indicator replaces the first four bits of the low order byte of the number. Example: -247 becomes 2 4 - 7 Decimal F2 F4 D7 Hexadecimal 11110010 11110100 11010111 Binary The number +247 becomes F2 F4 C7 11110010 11110100 11000111
PD	(packed decimal, signed). Each digit of the decimal number is converted into its 4-bit binary equivalent. The sign indicator is put into the rightmost four bits of the number. Example: -247 becomes 2 4 7 - Decimal 24 7D Hexadecimal 00100100 01111101 Binary The number +247 becomes 247C in hexadecimal.
FI	(fixed point, signed). The complete number is represented by its binary equivalent in either halfword or 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.

Format	Description
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
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

Format	Description
ASL	(signed numeric, ASCII, leading separate sign). Similar to the CSL format but with decimal data assembled into ASCII code. Example: +247 punched into card becomes + 2 4 7 Punched numeric data 2B 32 34 37 Hexadecimal 0101011 00110010 00110100 00110111 Binary ASCII code Similarly -247 becomes 2D 32 34 37 hexadecimal
AST	(signed numeric, ASCII, trailing separate sign). This gives the same bit representation as the ASL format, except that the sign is punched after the number. Example: 247+ becomes 32 34 37 2B hexadecimal

[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 are found in the *OS/VS—DOS/VSE—VM/370 Assembler Language Manual*, Section G.

Appendix E. EBCDIC and ISCII/ASCII Collating Sequences

EBCDIC

Figure 61 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 61 (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	@	Commercial At
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 61 (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 61 (Part 3 of 3). EBCDIC Collating Sequence

ISCII/ASCII

Figure 62 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	Bit		
Sequence	Configuration	Symbol	Meaning
0	00000000		Null
32	00100000	SP	Space
33	00100001	!	Exclamation point
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 62 (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 slash
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 62 (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 62 (Part 3 of 3). ISCII/ASCII Collating Sequence

Appendix F. DFSORT Abend Processing

This appendix explains how DFSORT processes an abend. It is intended to help you get the dump you need in order to diagnose the error causing the abend.

All abend dumps produced by DFSORT are system abend dumps that can be processed by standard dump analysis programs. A dump will be generated if you have included a SYSUDUMP, SYSABEND, or SYSMDUMP DD statement in your jobstream. The actual output of the system dump depends on the system parameters specified in the IEADMP00, IEAABD00 or IEADMR00 members of SYS1.PARMLIB by your installation.

DFSORT ESTAE Recovery Routine

At the beginning of each run, DFSORT establishes an ESTAE recovery routine to trap system or user abends for Blockset and Peer/Vale applications. You can delete the routine by specifying ICEMAC ESTAE=NO during installation, or by specifying NOESTAE on the DEBUG control statement. We recommend that you always run with ESTAE in effect so that in the event of an abend the following benefits are available:

- In general, you get dumps closer to the time of the abend.
- You get additional information useful in diagnosing the problem causing the abend.
- If you have activated SMF, an ICETEXIT routine, or an EFS program, DFSORT attempts to continue processing. That is, an SMF record is created, the termination exit is called, and/or major calls 4 and 5 are made to the EFS program before the application terminates. Of course, if one of these functions caused the abend, that function will not complete successfully.

At the end of its recovery routine, DFSORT always returns control to the system to allow termination to continue. The system will then invoke the next higher level ESTAE recovery routine.

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 3, "Using DFSORT Program Control Statements" on page 53).

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 "Reading List" on page 389.

DFSORT Abend Categories

There are two categories of abends for DFSORT: unexpected abends and user abends issued by DFSORT

- Unexpected abends

  These are system abends or user abends not issued by DFSORT. The abend code in these cases is the system abend code or the user abend code. See the *DFSORT Diagnosis Guide* for information about detecting common user errors and reporting DFSORT program failures.

- User abends issued by DFSORT

  DFSORT will issue user abends under the following circumstances:

  - The ABEND or ABSTP option is in effect and DFSORT encounters an error that prevents completion of the run.
  - DFSORT detects an error in its internal logic.

  See *DFSORT Messages and Codes* for complete information about user abends issued by DFSORT.

Abend Recovery Processing for Unexpected Abends

DFSORT normally has an ESTAE recovery routine established to trap system or user exit routine abends for Blockset and Peer/Vale applications. If an abend occurs, the system will pass control to this routine. The DFSORT ESTAE recovery routine functions are shown below:

- Abend dump

  The recovery routine will first have the system issue an abend dump to capture the environment at the time the error occurred.

- Termination functions

  DFSORT tries to accomplish the following tasks when they are specified, whether the program terminates normally or abnormally.

  - Calls 4 and 5 to an EFS program
  - Create the SMF record
  - Call the ICETEXIT routine

  The DFSORT recovery routine executes any of the functions specified above if they have not already been executed at the time of the abend.

- Abend information message

  For unexpected system or user exit routine abends, the DFSORT recovery routine issues message ICE185A giving information about when the abend occurred. The description of this message is in *DFSORT Messages and Codes*.

- Snap dumps

  The DFSORT recovery routine provides snap dumps of the DFSORT communication area COMMON and the system diagnostic work area SDWA. The snap dumps are written to a dynamically allocated data set whether or not a SYSUDUMP (or SYSABEND or SYSUDUMP) DD statement is included in the jobstream.

- Copy system diagnostic work area

  If an invoking program passes the address of an SDWA area in the 24-bit or extended parameter list, DFSORT will copy the first 104 or 112 bytes of the system diagnostic work area into the user SDWA area. See Chapter 6, "Invoking DFSORT from a Program" on page 231 for more information.

- Continuation of an application after successful output

  If an unexpected abend occurs after the sort, merge, or copy application completed successfully, DFSORT issues message ICE186A and completes the cleanup and termination functions it would have normally completed. A sort, merge, or copy application is completed successfully when the output of the application has been written to a data set or passed to a user exit without error. The output data set written by DFSORT is closed. The run is successful except for the function causing the abend. Message ICE186A says that the DFSORT output is usable even though the run has abended. You can then decide to use the output or rerun the job.

- DFSORT returns control to the system at the end of its abend recovery processing so that recovery routines can be invoked.

The DFSORT abend recovery routine functions described above may not be performed after an abend if NOESTAE is in effect. The DFSORT ESTAE recovery routine is always established at the beginning of a run. It is deleted early in DFSORT processing if NOESTAE is in effect.

Processing of Error Abends with A-Type Messages

When DFSORT encounters a critical error, it issues an A-type message and terminates. You can specify that DFSORT is to terminate the application with an abend through the ABEND or ABSTP options.

If abend termination is in effect and DFSORT encounters a critical error, DFSORT first causes an abend dump to capture the environment at the time of the error. Then, it issues the A-type message. It also executes the termination functions described earlier before terminating with an abend. The abend code will be the error message number, or a number between 1-99, as determined during installation with the ICEMAC ABCODE option.

With NOESTAE and ABEND in effect, the abend dump is produced after the A-type message is printed and other termination functions are executed. As a result, the dump produced may not reflect the conditions at the time of the error. It may not include the module that encountered the error.

With NOESTAE and ABSTP in effect, the correct module will be dumped but the A-type message will not be issued.

CTRx Abend processing

The CTRx option can be used to diagnose a problem by requesting that DFSORT abend during record input or output processing. See the DEBUG control statement in Chapter 3, "Using DFSORT Program Control Statements" on page 53. DFSORT will cause an 0C1 abend when the CTRx count is satisfied. The DFSORT ESTAE recovery routine will process the abend in the same way as it does for an unexpected abend described earlier.

Abend Recovery Processing and Invoking Programs

The DFSORT ESTAE recovery routine will return control to the system which will pass control to any ESTAE recovery routine(s) established by invoking programs.

As described earlier, the DFSORT ESTAE recovery routine will save the first 104 or 112 bytes of the system diagnostic work area in the invoking program's SDWA area if the address of the area is passed to DFSORT.

Since PL/I normally has an ESPIE in effect to intercept program checks (0Cx abend codes), the DFSORT ESTAE recovery routine is not entered after these errors unless you have specified NOSPIE. DFSORT abend recovery processing will occur for all other types of abends.

Invocations from COBOL programs or use of COBOL exits can result in more than one abend dump.

Summary of Changes for Previous Releases of DFSORT

Release 10, January 1988

- Better performance through internal enhancements to the Blockset technique, which reduce required SORTWK space, CPU time, and EXCPs.
- DFSORT Panels, an interactive menu-driven facility supported under ISPF and ISMF, that lets you use DFSORT on-line with minimal experience.
- New ICEMAC options:
  - With DYNAUTO, you can specify automatic dynamic allocation of intermediate data sets.
  - With ABCODE, you can customize critical error ABEND codes using either the critical message numbers as before, or your own choice of a number between 1 and 99.
  - With EXITCK, you can prevent termination when E15/E35 user routines use certain "invalid" return codes.
  - With NOMSGDD, you can prevent termination when a needed message data set is not present.
- The MODS statement allows you to specify that user exit routines are in the STEPLIB/JOBLIB or link library data sets.
- A new subparameter for the DYNALLOC execution option lets you suppress dynamic allocation of intermediate data sets.
- New parameters added to the EXEC statement let you specify options formerly available only on ICEMAC or control statements. Alternate parameter names (for example, MSGDD for MSGDDN) are also available.
- For increased flexibility, DFSORT now accepts:
  - Semicolons interchangeably with commas in the EXEC parm string and within control statements.
  - Blank statements.
  - FB, VB, and DB for RECORD TYPE.
  - Duplicate message data set DD statements (the first is used).
  - SORTWKxx DD statements for a merge job (ignored).
  - Empty VSAM input data sets for sort and copy jobs.
  - VSAM input and non-VSAM output without the RECORD TYPE parameter.
  - Shorter output LRECL for VLR data sets.
  - The STOPAFT parameter on the SORT and MERGE statements.
  - LS, TS, OL, and OT as format types on the SORT, MERGE, INCLUDE, and OMIT statements.
- Automatic estimation of VSAM input data set size for intermediate storage optimization.
- DFSORT uses the job stepname rather than the procedure stepname in messages and SMF field ICESTPNM.

Release 9, April 1987

- Improved performance:
  - For sort, merge, and copy jobs from internal enhancements to the Blockset techniques.
  - For fixed-length, nonspanned, VSAM data sets from more efficient input processing in the Blockset techniques.
- Installation-written exit (ICEIEXIT and ICETEXIT) routines can be used to exercise control over individual jobs and to collect job statistics.
- Use of the more efficient DFSORT copy function for qualifying IEBGENER jobs.
- Improved capability to run two different releases of DFSORT at the same time.
- Additional statistical information in the SMF record.
- Capability to make zoned-decimal positive-sum fields printable.
- User-written Extended Function Support (EFS) programs can be used to:
  - Examine, alter, and/or ignore DFSORT and non-DFSORT control statements or EXEC PARM options.
  - Support sorting or merging of user-defined data types with user-defined collating sequences.
  - Provide the logic to include or omit records based on user-defined data types.
  - Provide user supplied messages to DFSORT for printing to the message data set.
- Beginning with Release 9, DFSORT will no longer support the OS/VS1 environment. Program services for OS/VS1 that were announced for Release 8.0 will be available for twelve months after the release of 9.

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/VS2 MVS JCL, GC28-0692

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/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/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/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

System Macros and Facilities, Volume 1, GC28-1151

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/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

ISPF

Interactive System Productivity Facility Dialog Management Services, SC34-4021

Interactive System Productivity Facility/Program Development Facility Program Reference, SC34-4024

ISMF

MVS/Extended Architecture Interactive Storage Management Facility: User's Guide, GC26-4266

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/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/VS2 MVS System Programming Library: Debugging Handbook, Volume 1, GC28-0708, Volume 2, GC28-0709, and Volume 3, GC28-0710

Index

DFSORT
Application Programming: Guide

SC33-4035-14

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.

If you have applied any technical newsletters (TNLs) to this book, please list them here: ____________________

Comments (please include specific chapter and page references) :

If you want a reply, please complete the following information:

Name ____________________ Date ____________________

Company ____________________ Phone No. (____) ____________________

Address ____________________

____

Thank you for your cooperation. No postage is 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 them directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

SC33-4035-14

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 MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Programming Publishing
P.O. Box 49023
San Jose, CA 95161-9023

Fold and tape

Please do not staple

Fold and tape

IBM®

Program Number
5740-SM1

File Number
S370-33

SC33-4035-14

Printed in U.S.A.