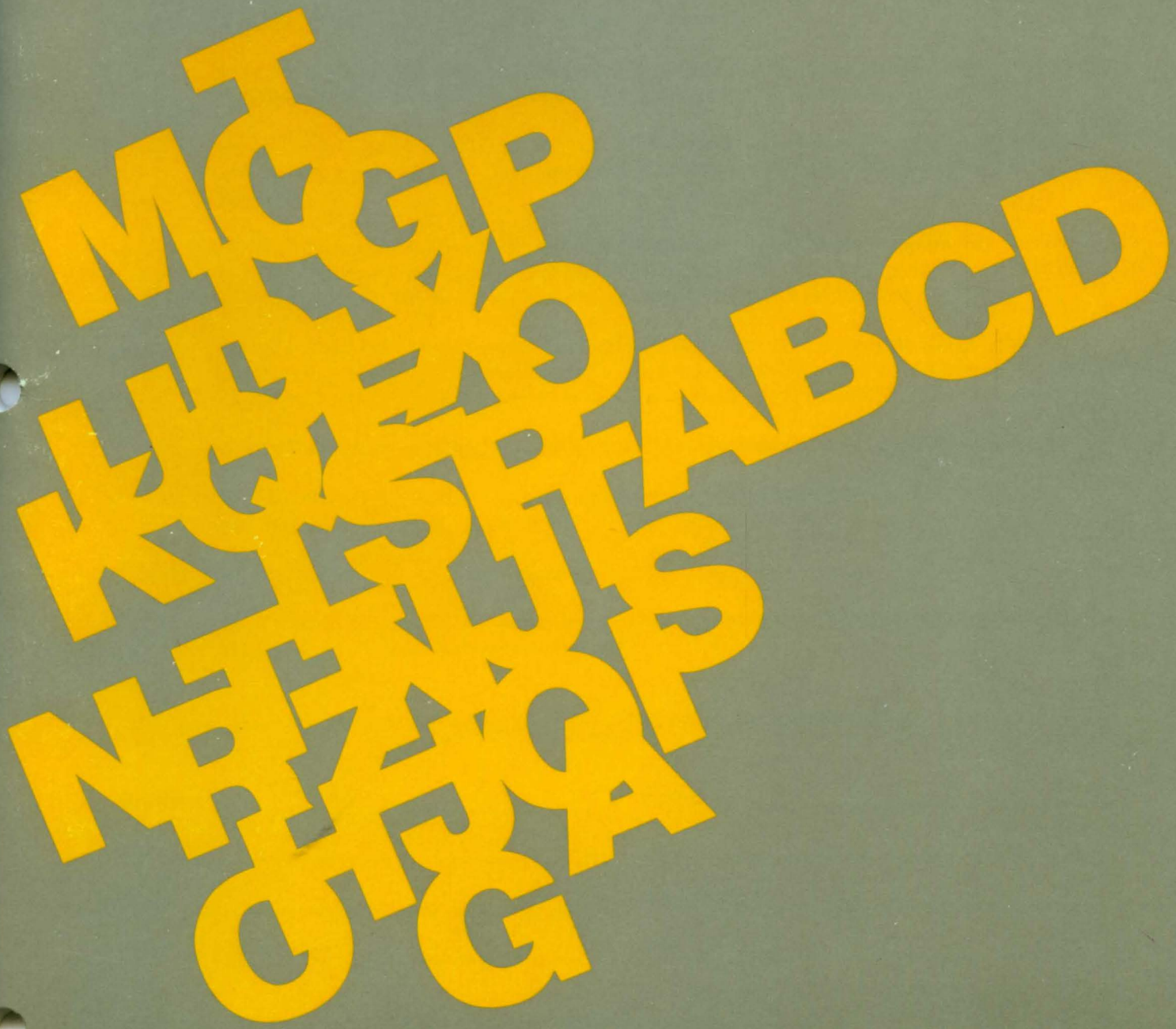
IBM

DFSORT
General Information

Program Product

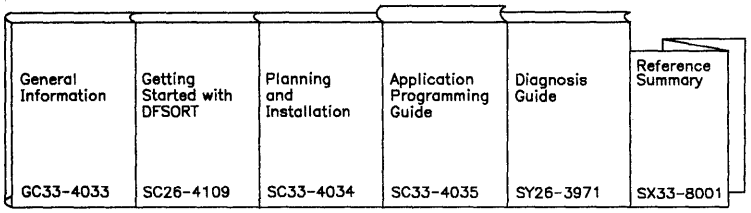


Order Number:
GC33-4033-12

Program Number:
5740-SM1

Release Number:
8.0

The DFSORT Library



Order Today

Just fill in the appropriate quantity to order IBM DFSORT publications.

☐ *DFSORT General Information* is an introduction to DFSORT. It is a source of general information for people involved in planning, managing, system support, or programming at an installation.

☐ *Getting Started with DFSORT* is for the new DFSORT user. It shows you step-by-step how to perform sort, merge, and copy applications and teaches you some techniques for using DFSORT efficiently.

☐ *DFSORT Planning and Installation* is written for experienced system programmers, has planning considerations and general instructions for installing DFSORT. It is designed to be used with the *DFSORT Program Directory*, which has more detailed installation instructions.

☐ *DFSORT Application Programming: Guide* enables the programmer to prepare all the input necessary to perform a sort, merge or copy application. It provides detailed information on how to use program control statements and how to estimate storage used by DFSORT during its execution.

☐ *DFSORT Diagnosis Guide* provides instructions for diagnosing and resolving program failures.

☐ *DFSORT Reference Summary* is a quick reference providing program control statements, job control statements, and parameter lists.

To order the IBM DFSORT product, simply check the box below.

☐ Yes, I want to improve my sort performance. Please send me DFSORT Release 8.0 (5740-SM1), which includes a ***FREE*** one month test period.

Fill in the section below or call us, toll free, at 800-IBM-2468 (Software and Education).

Signature _____ Company _____

Name _____ Address _____

Phone _____ City, State _____

IBM Customer Number _____ ZIP _____

Note: Your IBM Customer Number is required for billing purposes.

GC33-4033-12

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

DRO/Software
IBM Corporation
1 Culver Road
Dayton, New Jersey 08810

Fold and tape

Please do not staple

Fold and tape

IBM®

IBM

DFSORT General Information

Program Product

Order Number:
GC33-4033-12

Program Number:
5740-SM1

Release Number:
8.0

Thirteenth Edition (January 1986)

This is a major revision of, and makes obsolete, GC33-4033-11.

This edition applies to Release 8.0 of IBM DFSORT, Program Product 5740-SM1, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

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

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

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

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

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

Preface

This publication provides an introduction to the IBM DFSORT (Data Facility Sort) Program Product 5740-SM1. It is intended as a source of general information for people involved in planning, managing, system support, or programming at an installation.

The publication gives a general description of the DFSORT program and its relationship to the operating systems and the machine environments. It also outlines compatibility considerations.

The reader is assumed to be familiar with the terminology and concepts of the operating system used.

Organization of Manual

This manual contains the following chapters:

Chapter 1, " Introduction" on page 1 describes the program, its performance, and its features.

Chapter 2, " Program Description" on page 9 describes the functional organization of the DFSORT program and the types of records, data set formats, devices that can be used, and other related information, including a control statement summary.

Chapter 3, " System Interface and Requirements" on page 25 describes the relationship of DFSORT to its operating systems, the minimum hardware configuration required for its residence and execution, and how it is installed.

Chapter 4, " Performance" on page 31 tells how you can aid the program's optimization toward higher performance.

Notational Conventions

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

The control statement command-format summary shown in "Program Description" uses these conventions:

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

Summary of Amendments

Release 8.0, January 1986

New features added to DFSORT include:

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

Release 7.1, June 1985

New enhancements added to DFSORT are the ability to:

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

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

- Processing of multivolume SORTOUT data sets with the EXCP access method rather than with the BSAM access method, whenever possible.

- Dynamical link-editing of user exit routines. Note that the SORT cataloged procedure has been changed to include link-edit DD statements.
- Writing program messages to the master console.

Release 7.0, January 1985

New features added to DFSORT are:

- For MVS/XA, the ability to reduce the processor time for sorting done in large storages by using IBM System/370 Extended Architecture Sorting Instructions.
- Virtual storage constraint relief (VSCR) for MVS/XA:
  - Improved performance because certain buffers and modules can now be placed above 16-megabyte virtual when sorting fixed-length records.
  - The ability to specify:
    - The upper limit of the amount of main storage above and below 16-megabyte virtual available to DFSORT (TMAXLIM).
    - The number of bytes reserved above 16-megabyte virtual for system use (ARESALL).
    - The number of bytes reserved above 16-megabyte virtual for a program that invokes DFSORT (ARESINV).
- COBOL-related enhancements:
  - The ability to invoke DFSORT from VS COBOL II programs.
  - The ability to use the VS COBOL II FASTSRT compile-time option, which enhances performance.
  - The ability to write E15 and E35 exit routines in COBOL.
  - The ability to specify an alternative message data set when invoking DFSORT with JCL. This is especially useful when exits are written in COBOL.
- Support of the IBM 3480 Magnetic Tape Subsystem.
- Removal of the upper limit of 48K bytes for the RESALL option.
- The ability to specify the maximum number of records to be accepted for sorting.

Contents

Chapter 1. Introduction **1**
How Well Does DFSORT Perform? 2
What Does DFSORT Do? 4
Functions 5
Features 5

Chapter 2. Program Description **9**
Control Fields and Collating Sequences 9
Control Fields 9
Collating Sequences 9
Alternative Collating Sequence 10
Control Statements 10
Control Statement Parameters 11
Control Statement Examples 13
Input/Output 15
Record Limitations 17
Statistical Data Collection 18
Program Initiation 18
Invoking DFSORT from a Program 19
Converting to the Extended Parameter List 20
Overriding Parameter List Options 20
Overriding Installation Defaults 22
Using Job Control Language (JCL) 22
User-Written Routines at Program Exits 23
Functions of Routines at User Exits 23

Chapter 3. System Interface and Requirements **25**
Relationship to OS/VS and MVS/XA 25
Minimum Hardware Requirements 25
Program Distribution 27
Program Installation 27

Chapter 4. Performance **31**
Using System/370-XA Sorting Instructions 31
Planning Applications 32
Efficient Blocking 33
Variable-Length Records 33
User-Written Exit Routines 33
Being Generous with Main Storage 33
Using Efficient Sort/Merge Techniques 34
Sort Techniques 34
Merge Techniques 34
Using Work Storage Devices Efficiently 35
Direct-Access Work Storage Devices 35

Device Data Transfer Rate 35
Tape Work Storage Devices 37
Specifying Input/Output Data Set Characteristics Correctly 37
Data Set Size 37
Variable-Length Records 37
Direct Access 38
Tape 38
Using JCL to Initiate DFSORT 39
Using Options That May Enhance Performance 39
COBEXIT 40
FASTSRT 40
INCLUDE or OMIT, STOPAFT, and SKIPREC 40
INREC and OUTREC 41
SUM 41

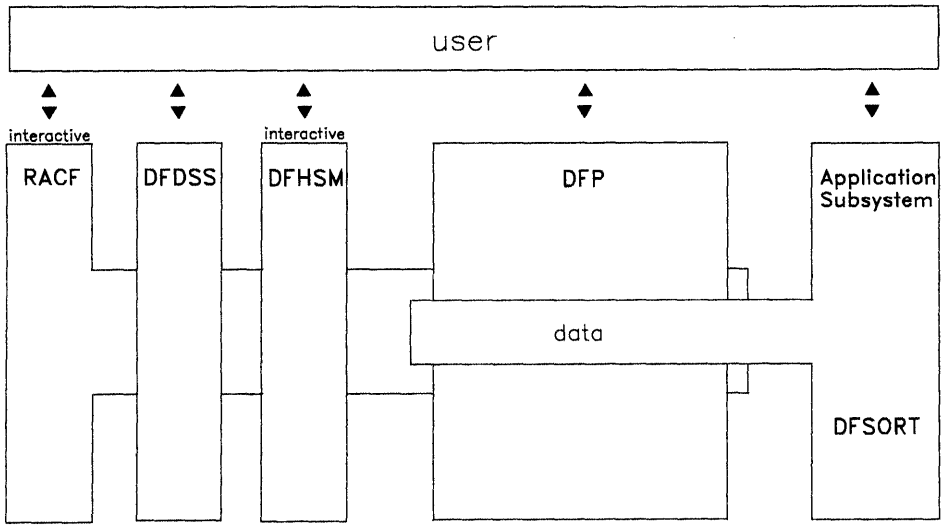
Index 43

Figures

1. Performance Improvement under MVS/XA 3
2. Performance Improvement under MVS/370 3
3. An Input/Output Overview 16
4. Data Set Characteristics of Input and Output Records 17
5. Overview of DFSORT Program Initiation from Four Sources 19
6. Converting to the Extended Parameter List 20
7. Functions of Routines at Program Exits 23
8. Information about Work Data Sets 26
9. Comparative Data Transfer Rates of Disk Work Storage Devices 36
10. Number of Tracks per Cylinder for Direct-Access Devices 38
11. External Work Storage Requirements of the Various Tape Techniques .. 39
12. Performance is Improved with FASTSRT 40

Chapter 1. Introduction

DFSORT is a member of a family of products that perform essential data processing tasks. These products are designed to work together so that the total benefits derived from the family may exceed the sum of the benefits of each individual product.



DFDSS Data Facility Data Set Services is a high performance data mover. It performs data backup and recovery, DASD conversion, and DASD space management tasks.

DFHSM Data Facility Hierarchical Storage Manager is the manager of inactive data. It is a key product for automating functions that involve space and availability.

DFP The Data Facility Product is the manager of active data and is a base for hardware and software. It manages programs, devices, and data.

DFSORT Data Facility Sort is a high performance data arranger that provides an efficient and flexible way to handle sorting, merging, and copying applications.

Together, DFDSS, DFHSM, DFP, and DFSORT make up the Data Facility family of products, the foundation for response to expanding storage management requirements. These products, along with the Resource Access Control Facility (RACF - the manager of data security and resource authorization) form the

strategic base from which IBM is evolving to system-managed storage in the MVS/Extended Architecture (MVS/XA) environment.

DFSORT is an important component of the Data Facility family with the potential to individually contribute significant benefits to data processing resources. DFSORT supports input and output data sets on any device supported by VSAM or QSAM. IBM's high performance direct access or tape storage devices can be used for intermediate storage.

DFSORT is a frequently invoked program and major user of system resources. In fact, it is not uncommon for the program to be invoked 4000 to 5000 times a week in a large installation. Typically it consumes from 10 to 20 percent of the processor resources and from 15 to 25 percent of the channel resources. Any improvement in a program that uses the system resources so heavily can result in significant dollar savings. Designed specifically for such performance improvements, DFSORT:

- Exploits device geometry, processor memory, and processor cache
- Exploits MVS/XA extended addressing
- Utilizes MVS/XA sorting instructions
- Supports VS COBOL II FASTSRT compile-time option

How Well Does DFSORT Perform?

Because reductions in CPU time, elapsed time, and the number of EXCPs issued by the program are such important considerations, the results from DFSORT test cases are noteworthy.

The following two figures show the improvement of Release 8.0 over Release 7.1 when sorting variable-length records.

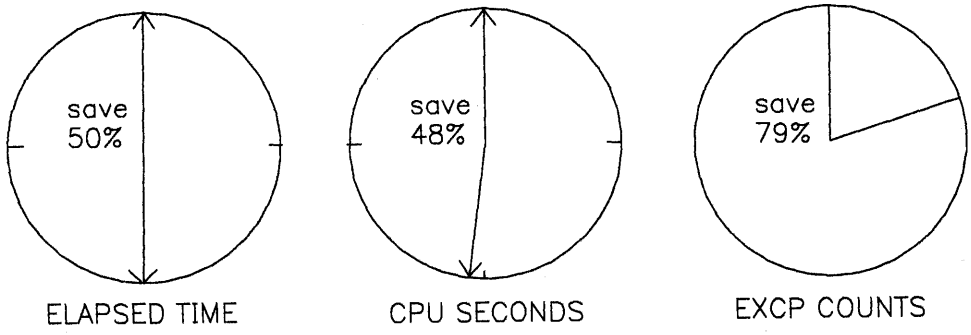


Figure 1. Performance Improvement under MVS/XA. This figure shows measured performance improvements for variable-length record sorts in Release 8.0, when compared to equivalent sorts in Release 7.1. Release 7.1 used JCL region sizes from 256Kb to 1Mb. Release 8.0 used the 2Mb virtual storage default.

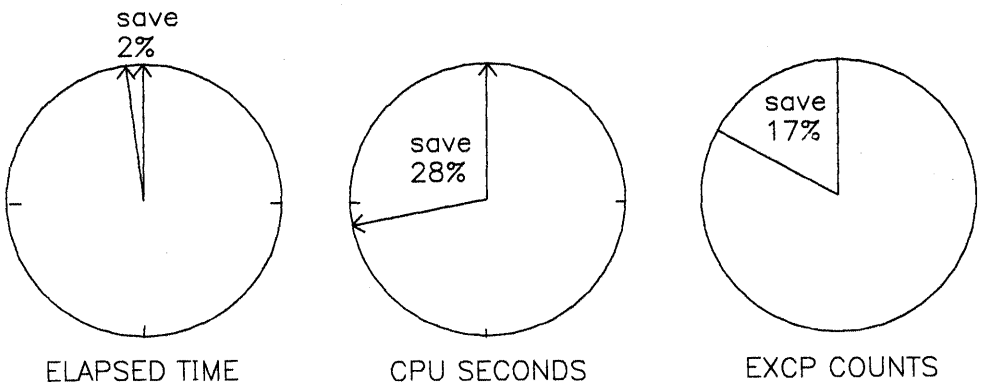


Figure 2. Performance Improvement under MVS/370. This figure shows the measured performance improvements for variable-length record sorts in Release 8.0, when compared to equivalent sorts in Release 7.1.

When sorting fixed-length record files, Release 8.0 has shown performance improvements over Release 7.1 to be approximately an 8 percent reduction of CPU time in the MVS/XA environment when using the two-megabyte virtual storage default.

Note: Many factors affect the performance of sort products: device speed, CPU model, record size, file size, region size, and so forth. IBM, therefore, does not warrant or represent that users will experience the same changes in performance indicated by the test results.

When comparing DFSORT Release 8.0 to 7.1, the tests were executed on a partitioned dyadic configuration of an IBM 3084 Model QX processor having 32 megabytes of storage. All data sets were on IBM 3380 Direct Access Storage Devices. Record sizes, file sizes, and JCL region sizes were selected to be representative of many user environments.

A number of factors contribute to the overall improved performance of DFSORT Release 8.0 when compared to Release 7.1:

- The significant enhancement of VLR (Variable-Length Record) Blockset
- The utilization of extended addressing capability and use of virtual storage above the 16-megabyte address boundary for VLR sorts on MVS/XA systems
- The use of DFSORT's highest performing processing technique, Blockset, for VSAM record processing
- The efficient use of processor cache memory which improves performance of fixed-length record sorts
- The removal of some restrictions that prevented the use of the more efficient Blockset technique

What Does DFSORT Do?

DFSORT offers five techniques for sorting, merging, and copying (hereafter referred to as "processing"): Fixed-Length Records (FLR) Blockset, Variable-Length Records (VLR) Blockset, Peerage, Vale, and Conventional techniques.

FLR- and VLR-Blockset are two high-performance techniques for processing fixed-length records and variable-length records. Because FLR-Blockset and VLR-Blockset are DFSORT's most efficient techniques, the program selects them whenever possible.

If conditions for selection of the Blockset techniques are not met, DFSORT will use either the Peerage, Vale, or Conventional technique, depending on several factors, including whether you are sorting or merging, whether you are working with FLR or VLR records, and whether you use disk or tape for intermediate storage. For a discussion of sorting and merging techniques, see Chapter 4, " Performance" on page 31.

Functions

DFSORT will:

Sort

The primary function of DFSORT is to sort data sets. Input can be blocked or unblocked sequential data sets containing fixed- or variable-length records on any device that can be used with QSAM or VSAM. Output can be either QSAM or VSAM output, regardless of the input form, but it must be of the same type (fixed or variable) as the input.

While sorting you can perform record-level editing operations such as reformatting and including or excluding certain records from the input file.

Merge

You can merge as many as 16 previously sorted QSAM or VSAM input data sets (but not both types together). Output can be either QSAM or VSAM output, regardless of the input form, but it must be of the same type (fixed or variable) as the input.

Record-level editing is supported.

Copy

The COPY function enables you to copy data sets without performing any sorting or merging operations.

Record-level editing is supported.

Input can be blocked or unblocked sequential data sets containing fixed- or variable-length records on any device that can be used with QSAM or VSAM. Output can be either QSAM or VSAM output, regardless of the input form, but it must be of the same type (fixed or variable) as the input.

For sort, merge, and copy, VSAM output data sets must be predefined using the Access Method Services facility.

Features

The features of DFSORT include:

Control Statements and Options

- Control statements provide flexibility in controlling DFSORT applications. They facilitate record-level editing operations.
- All control statements can be specified in the SYSIN or SORTCNTL data set regardless of what processing technique is used by DFSORT. (SORTCNTL is valid only when DFSORT is dynamically invoked by another program.)
- Control statements can be used instead of complex programming logic.

- You can temporarily override most specifications made during installation of DFSORT.

Program Exits

User-written routines can be used to perform a variety of operations, such as deleting, inserting, and altering records.

Operation in MVS/XA Mode

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

DFSORT places certain modules and buffers above 16-megabyte virtual leaving more space below for user programs. More and larger buffers provide greater optimization opportunities. For MVS/XA users who install DFSORT resident, most of the Blockset modules will be placed above 16-megabyte virtual in the extended link pack area.

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

Long Records

The maximum record length for variable-length records is 32756 bytes, and, for fixed-length records, is 32760 bytes.

Auxiliary Storage

These features involve auxiliary storage:

Mixed Device Types

Disk work data sets can be allocated on mixed device types, and space need not be contiguous. As many as 32 work data sets can be allocated for a disk sort.

Automatic Release

Disk work space and output data set space that is no longer needed after all records have been read in may automatically be released and thus given back to the system before processing ends. The DFSORT program can be installed with or without this option.

Automatic Secondary Allocation

DFSORT automatically allocates secondary extents on your disk work and output data sets if there are insufficient primary extents. This minimizes the probability of exceeding intermediate storage capacity and prevents DFSORT from terminating before the whole file is processed. The DFSORT program can be installed with or without this option.

Dynamic Allocation (MVS)

You can assign to DFSORT the task of dynamically allocating needed work space. (This will relieve you of the necessity of calculating and specifying, through JCL, the amount of intermediate work space needed by the program.) The program, by use of the dynamic allocation facility of the operating system, allocates work space for the current sort application.

Main (virtual) Storage Sort

Small data sets that can be contained within main storage can be sorted without any work data sets, subject to some conditions for the Blockset sorting techniques.

Collating Options

DFSORT provides several collating options:

Long Control Fields

Control fields may be contiguous separated, or may overlap. The control fields may occur anywhere within the first 4092 bytes of a data record but their total length must not exceed 4092 bytes.

EQUALS Preservation

Input order of records with equal control fields can be preserved.

Alternative Sequence

You can specify a deviation from the standard EBCDIC collating sequence (alternative collating sequence) when DFSORT is installed. You can also, optionally, designate an alternative collating sequence for character format.

System Facilities

Two system facilities provided by DFSORT are:

SMF Records

Additional SMF records can be used to gather statistical information about DFSORT applications.

Checkpoint/Restart

The system checkpoint/restart facility can be used with the Peerage, Vale, and Conventional techniques.

Messages

- Messages include a diagnostic message trace character to identify the source of the message in the code.
- Information messages are issued containing statistical information (such as the average record length for variable-length record sort applications) about the sort run.
- You can determine the level of messages issued (diagnostic, informative, critical, or none).
- You can control whether control statements will be printed.
- You can control whether DFSORT messages will be written to the master console.

Chapter 2. Program Description

This chapter describes the types of records, data set formats, and devices that can be used with DFSORT. It describes the collating sequences used, the program control statements, and program initiation methods.

It also gives a functional description of the exits provided in the program for entry into routines written by the user for particular applications. In addition, it tells how you can collect statistical data.

Control Fields and Collating Sequences

Control Fields

Control fields are portions of records, designated by the user, which are used by the program to sort or merge records into a specified sequence.

DFSORT determines the sequence of data records by using one or more control fields, which must appear in the same relative position in each record. The sequence for each control field can be ascending or descending.

Starting with the first control field, the contents of the control fields are compared when all previous comparisons have resulted in an equal condition.

For a sort or a merge, the order of equally collating records is preserved when the EQUALS option is active. When the EQUALS option is not active, the order of equally collating records is not preserved.

Control fields may be contiguous, separated, or may overlap. The control fields may occur anywhere within the first 4092 bytes of a data record but their total length must not exceed 4092 bytes.

Collating Sequences

The sequence of output records is determined by the standard EBCDIC collating sequence, the ISCII/ASCII collating sequence, or a user-specified collating sequence.

Alternative Collating Sequence

The EBCDIC collating sequence can be modified at program installation time by specifying the ALTSEQ option, or at execution time with the ALTSEQ program control statement. Modifying the collating sequence means that one or more characters in the sequence can be changed.

The ALTSEQ translation table applies to the AQ field format. The user can now specify, at either installation or execution time, whether the ALTSEQ translation table should also apply to the CH field format.

Control Statements

DFSORT control statements provide details about how the records are to be processed. They also provide information about the control fields and collating sequence in addition to information such as a description of the input data, whether the order of equally collating records is to be preserved from input to output, whether any user-written routines are to be included during program execution, and so on.

You can select one of three applications.

SORT Can be used to specify a sort or copy application.

MERGE Can be used to specify a merge or copy application.

OPTION Can be used to specify a copy application. Also, it can be used to modify certain program options specified at installation time or specified on the SORT or MERGE control statement, and to supply other optional information.

Other control statements include:

ALTSEQ Specifies modifications to the standard EBCDIC collating sequence.

DEBUG Specifies an abend and/or a return code 16 on a critical error, and/or specifies use of BSAM for input/output.

END Causes DFSORT to discontinue reading SYSIN or SORTCNTL.

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

INREC Specifies how records will be reformatted before they are processed.

MODS Normally required when program exits are to be used.

OMIT Specifies that records whose fields do not meet certain criteria will be deleted.

OUTREC Specifies how records will be reformatted before they are output.

RECORD Required when record lengths will be changed during execution of the program, when there is no SORTIN DD statement, or when input is a VSAM data set.

SUM Specifies that summary fields in records with equal control fields will be summarized in one of the records, and that the other records will be deleted.

Note: The INPFIL and OUTFIL control statements, which are used by other IBM sort/merge programs, are ignored.

Control Statement Parameters

A list of the statements and their parameters is given below:

```
Statement  Parameters

ALTSEQ     CODE=(fftt...,fftt)
DEBUG      [ABEND|NOABEND]
           [,ABSTP]
           [,BSAM]
           [,BUFFERS={ANY|BELOW}]
           [,CTRx=n]
           [,FMTABEND]
           [,NOASSIST]
END        None
INCLUDE    {COND=(logical expression)|
           COND=(logical expression),FORMAT=f}
INREC      FIELDS=([s,]p,m[,a]...
           [,s][,p,m[,a]][,s])
MERGE      {FIELDS=(p,m,f,s...,p,m,f,s)|
           FIELDS=(p,m,s...,p,m,s),FORMAT=f|
           FIELDS=COPY}
           [,CKPT]
           [,EQUALS|,NOEQUALS]
           [,FILES=n]
           [,FILSZ=x|,SIZE=y]
MODS       exit=(n,m,s[,e])...,exit=(n,m,s[,e])
OMIT       {COND=(logical expression)|
           COND=(logical expression),FORMAT=f}
OPTION     [ARESALL={n|nK}]
           [,ARESINV={n|nK}]
           [,CHALT|,NOCHALT]
           [,CHECK|,NOCHECK]
           [,CKPT]
           [,COBEXIT={COB1|COB2}]
           [,COPY]
           [,DYNALLOC[={d|(d)|(,n)|(d,n)}]]
           [,EQUALS|,NOEQUALS]
           [,FILSZ=x|,SIZE=y]
           [,LIST|,NOLIST]
           [,MAINSIZE={n|nK|MAX}]
           [,MSGDDN=ddname]
           [,MSGPRT={ALL|NONE|CRITICAL}]
           [,NOBLKSET]
           [,NOOUTREL]
           [,NOOUTSEC]
           [,NOSTIMER]
           [,NOWRKREL]
           [,NOWRKSEC]
           [,RESALL={n|nK}]
           [,RESINV={n|nk}]
           [,SKIPREC=z]
           [,SORTDD=cccc]
           [,SORTIN=ddname]
           [,SORTOUT=ddname]
           [,STOPAFT=n]
           [,VERIFY|,NOVERIFY]
           [,VLSHRT|,NOVLSHRT]
OUTREC     FIELDS=([s,]p,m[,a]...
           [,s][,p,m[,a]][,s])
RECORD     [LENGTH=(L1,L2,L3,L4,L5,L6,L7)]
           [,TYPE=x]
```

```
SORT        {FIELDS=(p,m,f,s...,p,m,f,s)|
            FIELDS=(p,m,s...,p,m,s),FORMAT=f|
            FIELDS=COPY}
            [,CKPT]
            [,DYNALLOC[={d|(d)|(,n)|(d,n)}]]
            [,EQUALS|,NOEQUALS]
            [,FILSZ=x|,SIZE=y]
            [,SKIPREC=z]
SUM         {FIELDS=(p,m,f...,p,m,f)|
            FIELDS=(p,m...,p,m),FORMAT=f|
            FIELDS=NONE}
```

Control Statement Examples

The following control statements could be used in a sort application:

Example 1: SORT

```
 SORT      FIELDS=(14,2,D,5,8,A),FORMAT=AQ
 RECORD    TYPE=V,LENGTH=(120,,,80,90)
*****************************************
*                                       *
*         CHANGE COLLATING SEQUENCE     *
*                                       *
*****************************************
 ALTSEQ    CODE=5BEA
 OPTION    FILSZ=E20000
```

The SORT statement tells the program to sort the input data according to two control fields. The major control field is located at byte 14 and is two bytes long. The major control field will be in descending sequence in the output data set. The minor control field begins at the fifth byte in the record and is eight bytes long. (The first 4 bytes of a variable-length record contain the record descriptor word (RDW).) The minor control field will be in ascending sequence in the output. Both control fields are in EBCDIC character format, and the collating sequence is to be modified.

The RECORD statement tells the program that the input data set contains variable-length records. The maximum record length is 120 bytes, and the minimum is 80 bytes. The average record length in the input data set is 90 bytes.

The comment statements are printed but otherwise ignored.

The ALTSEQ statement specifies that X'5B' ('$') is to collate after X'E9' ('Z').

The OPTION statement tells the program that the estimated size of the data set is 20000 records.

For compatibility with previous releases, some of the options available on the OPTION statement can also be coded on the SORT or MERGE statement. However, if you code the same options on both the OPTION and SORT or MERGE statements, those on the OPTION statement will be used. It is preferable to code all your options on the OPTION statement only.

Example 2: INCLUDE

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

This statement will include 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.

Example 3: INREC

```
INREC FIELDS=(1,32)
```

This statement reformats the input record before it is sorted. The reformatted input/output record will contain only bytes 1 to 32 of the input record.

Example 4: OUTREC

```
OUTREC FIELDS=(1,42,4X,100,5)
```

This statement specifies that the output record will contain bytes 1 to 42 of the input record, 4 blanks, and bytes 100 to 104 of the input record.

Example 5: SUM

```
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.

Input/Output

Input to DFSORT consists of JCL and program control information, and records to be processed. Output consists of a data set containing the processed records, a return code, optional program control information (such as a listing of program control statements and messages), and an SMF record as defined at installation time. Input and output record types (fixed or variable) must be the same.

User Input and Output

When user-written exit routines are used, output can contain records generated by the routines themselves, as well as records altered or inserted by the routines. If desired, the output data set can contain only records created by user routines. User routines can have input and output data sets independent of those involved in DFSORT processing. (See "User-Written Routines at Program Exits" on page 23.)

Data Set Identification

The data sets used for DFSORT are identified using the operating system's JCL. DD statements with special ddnames describe the physical and logical organization of the input, output, and work data sets for DFSORT. These and other JCL statements are discussed under "Program Initiation" on page 18.

Figure 3 on page 16 is an overview of input and output for a DFSORT application.

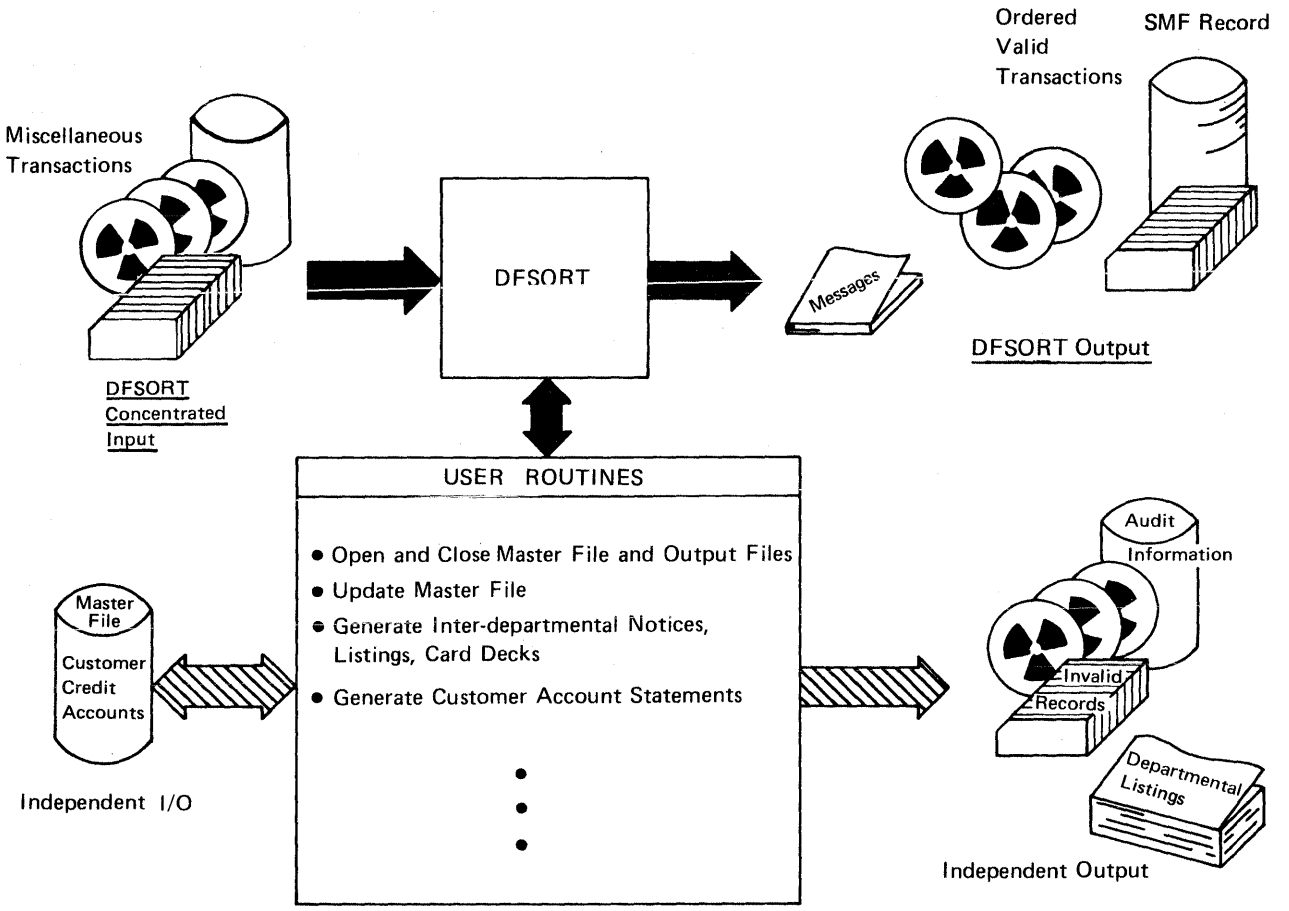


Figure 3. An Input/Output Overview

Record Limitations

The characteristics of input and output records are shown in Figure 4.

Data set	Input to a Sort or Copy	Input to a Merge	Output
Organization	Sequential (QSAM or VSAM).	Sequential (QSAM or VSAM).	Sequential (QSAM or VSAM).
Number	Up to 255 concatenated QSAM or 1 VSAM.	2 through 16.	1 (can be multivolume).
Content	Unsorted or sorted records.	Records previously sorted on the same basis as that required for output.	Sorted or copied records.
Blocking	Blocked or unblocked. Data sets can have different block sizes if the largest is specified first.	Blocked or unblocked. Data sets can have different block sizes.	Blocked or unblocked.
Code	EBCDIC, ISCII, or ASCII.	EBCDIC, ISCII, or ASCII.	EBCDIC, ISCII, or ASCII.
Record Format	Fixed or variable (including spanned), but must all be the same.	Fixed or variable (including spanned), but must all be the same.	Fixed or variable (including spanned).
Record Length	Min: 18 bytes if tape devices used; otherwise, 1 byte. Max: see Figure 7.	Min: QSAM/VSAM, Max: QSAM/VSAM (but see under "Record Limitations").	Same as for input.

Figure 4. Data Set Characteristics of Input and Output Records

The maximum record length the program can handle depends on the amount of main storage available to it. For sorting, it also depends on the type and number of intermediate storage devices. Maximum record lengths are shown in Figure 4. With the standard disk sorting techniques, if the input file is small enough to be contained in main storage, no intermediate storage will be used.

When spanned records are processed, a work area is used to assemble the records. As a result, the available space for buffers and for processing is decreased and the maximum record length the program can accept becomes less for a given main storage size. The maximum record length for variable-length records is 32756 bytes, and, for fixed-length records, 32760 bytes.

Statistical Data Collection

If you want to collect statistical data concerning execution time, record distribution, and so on, you can use the SMF installation option. SMF is a parameter operand of the ICEMAC installation macro.

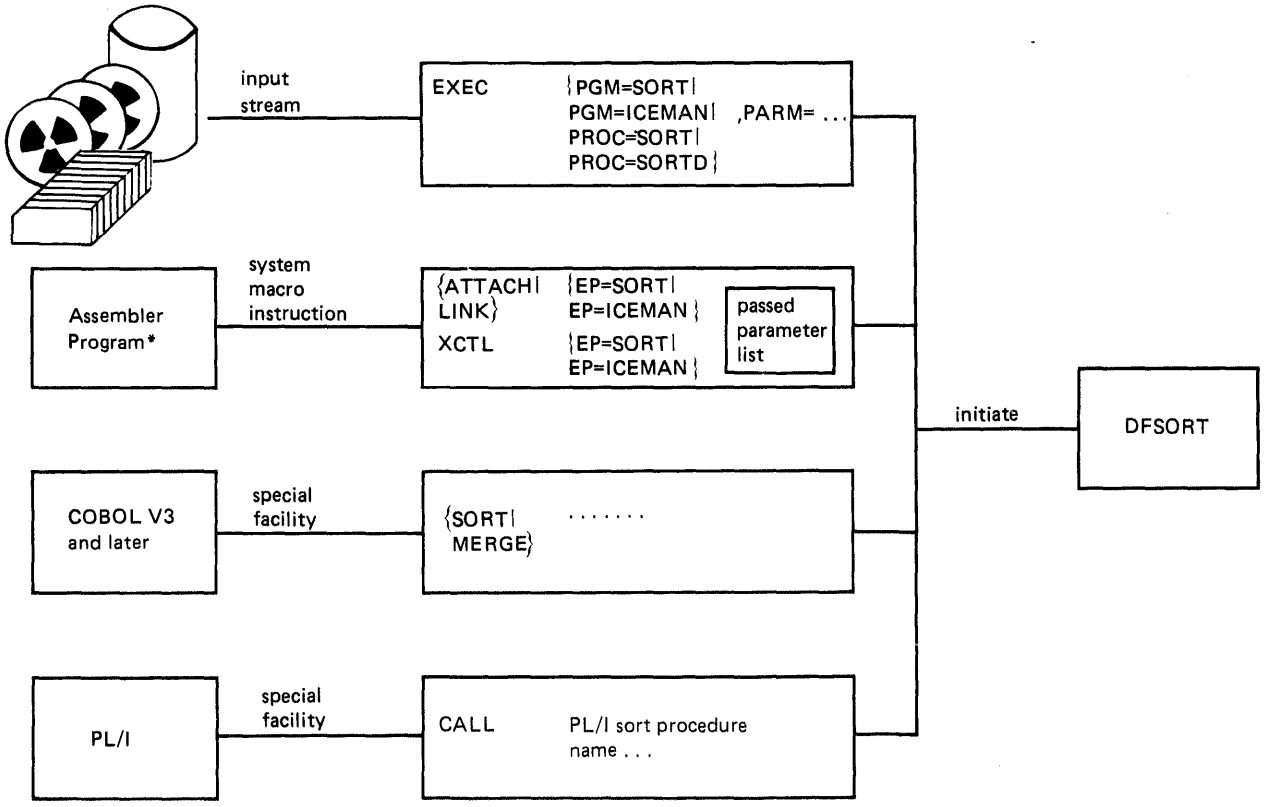
If SMF is specified, DFSORT causes an SMF record to be written for each application that completes successfully. If an SMF record is desired, either a short or full SMF record can be produced by means of the SMF parameter on the ICEMAC installation option. A full SMF record will be produced by DFSORT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

Note: If you want DFSORT to produce SMF type-16 records under the MVS operating system, an SVC routine for DFSORT must be installed. If SMF records under the VS1 operating system are desired, an SVC routine for DFSORT must be installed.

Program Initiation

As Figure 5 on page 19 shows, DFSORT can be initiated from:

- A job stream entered from tape, disk, or cards.
- An assembler language program that invokes DFSORT.
- A COBOL program that invokes DFSORT using the COBOL SORT or MERGE verb.
- A PL/I program that invokes DFSORT using sorting subroutines.



*If DFSORT is called by any type of LOAD and BALR interface, the invoking program is overlaid.

Figure 5. Overview of DFSORT Program Initiation from Four Sources

Invoking DFSORT from a Program

Programs written in Basic Assembler Language, OS/VS COBOL, VS COBOL II, or OS/VS PL/I can invoke DFSORT.

Note: If you have installed VS COBOL II and want to make use of its features, make sure that your invoking programs have been recompiled.

When invoking DFSORT from a program, you can pass DFSORT control statements by using the SORTCNTL data set. Later, you can add or change control statements in the SORTCNTL data set without recompiling the invoking program.

Converting to the Extended Parameter List

Programs that invoke DFSORT dynamically pass control statements and other options using either a 24-bit or extended parameter list. Programs that use the 24-bit parameter list can be converted to use the extended parameter list by using a combination of the parameters in the extended parameter list and the control statements in its control statement area. Figure 6 shows the relationship between the two parameter lists.

24-Bit List	Equivalent in Extended List
SORT | MERGE, RECORD, MODS, ALTSEQ control statements	Corresponding control statements
Address of E15 or E32 routine	Address of E15 or E32 routine
Address of E35 routine	Address of E35 routine
Main storage value	MAINSIZE option of the OPTION control statement
Reserved main storage value	RESINV option of the OPTION control statement
Message ddname	MSGDDN option of the OPTION control statement
Number of input files to a merge-only	FILES option of the MERGE control statement
ALTSEQ translation table	ALTSEQ translation table
STAE information	STAE information
Message type option	MSGPRT option of the OPTION control statement
Option characters for ddnames	SORTDD option of the OPTION control statement
Short records	VLSHRT option of the OPTION control statement

Figure 6. Converting to the Extended Parameter List

Overriding Parameter List Options

Control statements other than DEBUG and OPTION specified in the SORTCNTL data set will completely override corresponding control statements specified in the 24-bit or extended parameter list.

Note: SORT and MERGE, and INCLUDE and OMIT, are considered to be corresponding control statements.

For example:

```
SORT FIELDS=(23,10,CH,A),NOEQUALS,SKIPREC=100
```

specified in the SORTCNTL data set will completely override:

```
SORT FIELDS=(23,10,CH,A),DYNALLOC,EQUALS
```

specified by means of the parameter list; that is, the statement specified in the parameter list will be ignored, so DYNALLOC and EQUALS will also be ignored for this run.

When specified in the SORTCNTL data set, the OPTION statement will *selectively* override corresponding options specified in the 24-bit or extended parameter list.

For example:

```
OPTION   NOEQUALS,DYNALLOC=(3330,1),SIZE=12352
```

specified in the SORTCNTL data set will selectively override:

```
SORT     FIELDS=(23,10,A,10,5,D),FORMAT=CH,DYNALLOC=3350
OPTION   NOCHECK,FILSZ=E12000,ARESALL=2K
```

specified in the extended parameter list; that is, DYNALLOC=3350 and FILSZ=E12000 will be ignored, but NOCHECK will be used.

Note: The OPTION statement cannot be specified by using the 24-bit parameter list, but some of the equivalent options can be specified in the parameter list fields and on the SORT or MERGE statements.

When specified in the SORTCNTL data set, the DEBUG statement also selectively overrides corresponding options specified in the 24-bit or extended parameter list.

```
DEBUG ABEND
```

specified in the SORTCNTL data set will selectively override:

```
DEBUG NOABEND,BSAM
```

specified in a parameter list; that is, NOABEND will be ignored, but BSAM will be used.

Overriding Installation Defaults

You can temporarily override most specifications made during installation of DFSORT by using the following at execution time:

- PARM field of the JCL EXEC statement (when DFSORT is invoked by JCL)
- PARAMETER list passed to DFSORT (when DFSORT is invoked by another program)
- Control statements specified in the SYSIN data set (when DFSORT is invoked by JCL)
- Control statements specified in the SORTCNTL data set (when DFSORT is invoked by a program)

Note: For details on overriding installation defaults, see *DFSORT Application Programming: Guide.*

Using Job Control Language (JCL)

In addition to the standard JCL statements required for normal program execution, DFSORT uses other dedicated JCL ddnames. These ddnames are as follows:

```
SORTLIB      SORTINnn
SORTIN       SORTCKPT
SORTOUT      SORTCNTL
SORTWKnn     SORTDKnn
SORTDIAG     SORTMODS
```

Note: You should not specify SORTDKnn in your JCL.

In some circumstances, cataloged procedures (SORT or SORTD) can be used to supply certain required statements.

If DFSORT is dynamically invoked, the prefix SORT in some of these ddnames, and the ddnames SORTIN and SORTOUT, can be changed.

User-Written Routines at Program Exits

User-written routines may receive control at predesignated points (exits) in DFSORT. The available exits and the functions for which they can be used are shown in Figure 7.

Linkage Conventions and Programming Languages

User-written routines must follow standard linkage conventions. All user-written routines can be written in any language that provides the ability to pass the location/address of a record or parameter list in register 1 and pass the return code values in register 15. Examples of such languages are assembler and PL/I. (PL/I users, however, must use the special SORT subroutine facilities of their language.)

E15 and E35 routines can also be written in COBOL.

Dynamic link-editing of user exit routines is supported for user exit routines written in any language that can pass the location/address of a record or parameter in register 1 and a return code in register 15. (For complete details, see *DFSORT Application Programming: Guide.*)

Functions of Routines at User Exits

Figure 7 summarizes the functions of user exit routines, and the exits and phases with which they may be associated.

Exit Routine Functions	Input Phase	Output Phase
Open user data sets/initialize	E11, E15	E31, E35
Insert records	E15	E32, E35
Delete/Alter records	E15	E35
Terminate the program	E15	E35
Summarize records		E35[1]
Determine action when intermediate storage insufficient	E16[2]	

Figure 7 (Part 1 of 2). Functions of Routines at Program Exits

Exit Routine Functions	Input Phase	Output Phase
Handle special I/O conditions:		
Input (incl. handling labels, read errors, EOF)	E18	E38[2]
VSAM password insertion, journaling, and other VSAM exits	E18	E38[2]
Output (incl. handling labels, write errors)	E19[2]	E39
VSAM password insertion, journaling, and other VSAM exits		E39
Modify control fields	E61	
Close user data sets/housekeeping	E15, E17	E35, E37

Figure 7 (Part 2 of 2). Functions of Routines at Program Exits

Notes to Figure 7:

1 The SUM control statement may be used instead of your own routine to summarize records.

2 Not valid for a disk work data set sort; it is ignored if specified.

Chapter 3. System Interface and Requirements

This chapter describes the relationship of DFSORT to its operating system, the minimum hardware configuration required for its residence and execution, and how it is installed.

Relationship to OS/VS and MVS/XA

The program operates under OS/VS1, MVS/370, and MVS/XA. Additionally, the above operating systems run as guests under VM.

- DFSORT must be initiated according to operating system conventions.
- Any data sets processed by DFSORT must be defined according to operating system standards.

Minimum Hardware Requirements

DFSORT is designed to operate with:

- All IBM processors supported by OS/VS1, MVS/370, or MVS/XA (System/370-XA sorting instructions must be activated to be used.)
- Any device supported by OS/VS1, MVS/370, or MVS/XA for program residence
- Any device supported by QSAM or VSAM for input and output
- Direct access or tape storage devices for intermediate storage

Permanent Requirements

When installing DFSORT, you will need direct-access space for the DFSORT cataloged procedures, DFSORT link modules, and other DFSORT program modules. For details, see *DFSORT Installation*. The modules can reside on any of the following devices: 2305, 2314/2319, 3330/3333, 3340/3344, 3350, 3375, or 3380.

If DFSORT is installed resident, it will require space in the link pack area; for MVS/XA, it will also require space in the extended link pack area.

Intermediate Storage Requirements

Most sorting operations need work data sets, as described in Figure 8. See also "Using Work Storage Devices Efficiently" on page 35.

Work Data Sets	Comments
When needed	Never used for a merge or copy. Always used for a sort unless the whole input file can be contained in available main storage.
Type	Disk, tape, or virtual I/O. The types cannot be mixed, but need not be he same as input or output.
When on tape	Can be on 2400 and/or 3400 series tapes. • 3 to 32 tape units may be used. • 7-track tape can be used, but only if input is also on 7-track tape.
When on disk	Work areas can be on a mixture of any of the following devices: - 2314/2319 - 3330/3333 - 3340/3344 - 3350 - 3375 - 3380 - 3850[1] The minimum number of areas is 0 and the maximum is 32. Noncontiguous allocations can be used and secondary allocation will be used automatically if needed. All system supported models of the 3880 are also supported for use with 3380.
When executing under MVS	Can be dynamically allocated by means of a parameter on the SORT or OPTION control statement.
When VIO	For performance reasons, VIO is not recommended.

Figure 8. Information about Work Data Sets

Note to Figure 8:

1 For performance reasons, the 3850 is not recommended.

For more details, see *DFSORT Application Programming: Guide.*

Note: The number of SORTWK data sets specified is limited to 32; if more than 32 SORTWK data sets are specified, only the first 32 will be used. The JCL for existing programs that do not allocate enough intermediate storage space using the first 32 SORTWK data sets must be changed.

Performance Considerations

The amount of storage available directly affects the performance of DFSORT; this should, therefore, be taken into consideration when planning your hardware configuration (see Chapter 4, " Performance" on page 31). The minimum main storage requirement is 88K bytes.

Program Distribution

DFSORT is distributed as a multifile installation tape that can be installed using System Modification Program, Release 4 (SMP), or System Modification Program, Extended (SMP/E).

Program Installation

Installation of DFSORT can be done any time after your operating system is generated.

DFSORT can be "tuned" to meet your installation's special requirements by setting various values in the ICEMAC installation macro and recompiling the macro. For further information on the installation process, refer to *DFSORT Installation.*

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

Parameters	Functions
ALTSEQ	Alters the normal EBCDIC collating sequence.
ARESALL	Specifies, for MVS/XA, the number of bytes reserved above 16-megabyte virtual for system use.
ARESINV	Specifies, for MVS/XA, 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	Suppresses record count checking for sorting applications that use the E35 user exit routine without a SORTOUT data set.
COBEXIT	Indicates whether the E15 and E35 routines will be 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 on MVS systems when DYNALLOC is specified at execution time (on either SORT or OPTION statement) without these values.

EQUALS Preserves the input order of equally collating records.

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

EXCPVR Uses EXCPVR for SORTWK I/O.

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

INV | JCL Indicates whether the 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 Lists program control statements.

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

MINLIM Sets a minimum limit to 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.

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

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

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

RESALL Reserves storage for system and application use.

RESDNTx Indicates for OS/VS1 systems whether DFSORT modules reside in the pageable supervisor area.

RESINV Reserves space for programs invoking DFSORT.

SIZE Sets maximum amount of main storage.

SMF Produces SMF records.

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

SVC Specifies a user SVC number for DFSORT.

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

VERIFY Verifies sequence of output records.

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

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

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

WRKSEC Specifies whether or not DFSORT should use automatic secondary allocation for temporary SORTWK data sets.

Chapter 4. Performance

DFSORT automatically optimizes performance by analyzing the information given to it. This automatic optimization results in setting variables (such as buffer sizes) and selecting the proper sorting or merging technique.

You can improve DFSORT's performance by:

- Using System/370-XA sorting instructions.
- Planning your application development (including data formats) for efficient use of the program.
- Being generous with main storage.
- Using the most efficient sort/merge techniques.
- Planning for most efficient use of work storage devices.
- Specifying the input/output data set characteristics correctly.
- Using JCL to initiate DFSORT.
- Using options that may enhance performance.

These techniques are described in detail below.

Using System/370-XA Sorting Instructions

On MVS/XA systems, the System/370-XA sorting instructions can enhance DFSORT's performance when sorting FLR records. DFSORT will usually select the System/370-XA sorting instructions if the following requirements are met:

- The System/370-XA sorting instructions are activated.
- FLR records are being sorted.
- The Blockset sorting technique is being used.

Planning Applications

You should consider several factors when you design new applications. Some of these factors are discussed in the following sections.

Location of Control Fields: The following example illustrates the benefit of locating control fields at the beginning of a record.

Assume that your input record has the following layout:

A	1	B	2	C

where 1 = the more significant control field
2 = the less significant control field
A, B, and C are not control fields.

Internally, the program reorganizes the record fields prior to the actual sort or merge as follows:

1	2	A	B	C

When the sort or merge is completed, the record fields are restored to their original positions.

By designing your record format to conform to the second diagram, you can improve program performance.

Control Field Data Formats and Descriptions: Whenever possible,

- Use either EBCDIC character or binary control fields.
- Place binary control fields so as to start and end on byte boundaries.
- Avoid using the alternative collating sequence character translation, since this function not only increases processor time, but also increases the total length of the internal record.
- Specify fixed-point, packed decimal, and zoned decimal control fields (if you know they will always be positive) so that they can be sorted or merged as if they were binary control fields.
- Use packed decimal format rather than zoned decimal, because DFSORT packs the control fields and also increases the total length of the internal record.
- If several contiguous character or binary control fields in the 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 is normally improved if you block input and output records.

Variable-Length Records

You can help the program's optimization toward high performance if you:

- Keep the difference between the longest and the shortest variable-length record as small as possible. By splitting your long logical record into several shorter physical records, you can achieve a record length distribution that improves the program's performance.
- Give DFSORT the correct information about your variable-length record sorting application. This includes, among other things, average and minimum record lengths.

User-Written Exit Routines

User-written exit routines in a DFSORT application usually increase the time required to run a job, especially if dynamic link-edit of the routines is performed. Several control statements may provide the same function as a user routine, and you should use them whenever possible. Examples of these control statements are INCLUDE, OMIT, and SUM.

By carefully designing your application from the beginning with the above-mentioned considerations in mind, you will experience improved performance for your DFSORT applications.

Being Generous with Main Storage

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 should not exceed the amount of real storage generally available for one initiator; otherwise, excessive paging may occur.

DFSORT requires a minimum of 88K bytes, but you can improve performance by using larger regions, on the order of 1 megabyte. Improved performance will be most noticeable with large files (3 megabytes and larger).

Note: 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 300K bytes should be available to allow DFSORT to use storage effectively.

For more details of main storage size definitions, see *DFSORT Installation* and *DFSORT Application Programming: Guide.*

Using Efficient Sort/Merge Techniques

Depending on various conditions, DFSORT selects different techniques for sorting and merging. For a copying application, the Blockset technique is always selected. Message ICE143I will inform you which technique has been selected.

Sort Techniques

One condition that affects which sorting technique is selected is the type of device used for intermediate storage. The Blockset, Peerage, and Vale techniques can be used only with disk devices. If you use a tape device, the less efficient Conventional technique will be used. The Blockset, Peerage, and Vale techniques are discussed below. For more information on using tape devices for intermediate storage, see "Tape Work Storage Devices" on page 37.

Blockset Sorting Techniques

Fixed-Length Records: DFSORT's most efficient fixed-length record technique, FLR-Blockset, will be used for most sorting applications. If one or more of the conditions for the FLR-Blockset technique are not met (for example, if the control field is too long), the Peerage or Vale technique will be used.

Variable-Length Records: The high-performance VLR-Blockset technique will be used for sorting variable-length records in most cases. If one or more of the conditions for the VLR-Blockset technique are not met (for example, if the control field is too long), the Vale technique will be used.

Note: The Blockset techniques may require more intermediate work space than Peerage or Vale. For more information, see *DFSORT Application Programming: Guide*.

Peerage and Vale Sorting Techniques

If the conditions for use of the Blockset sorting techniques are not met, DFSORT will use Peerage or Vale.

Merge Techniques

For merging applications, DFSORT uses the Blockset and Conventional techniques.

Blockset Merging Techniques

Fixed-Length Records: DFSORT's most efficient merging technique, FLR-Blockset, will be used for merging fixed-length records in most cases.

Variable-Length Records: The high-performance VLR-Blockset technique will be used for merging variable-length records in most cases.

Conventional Merge Technique

If the conditions for use of the Blockset merging techniques are not met (for example, if the control field is too long), DFSORT will use the Conventional merge technique.

Using Work Storage Devices Efficiently

Performance is enhanced when multiple channels are available. Performance is also improved if the device is connected so that two channel paths exist between each device and the central processing unit that is running the program.

Direct-Access Work Storage Devices

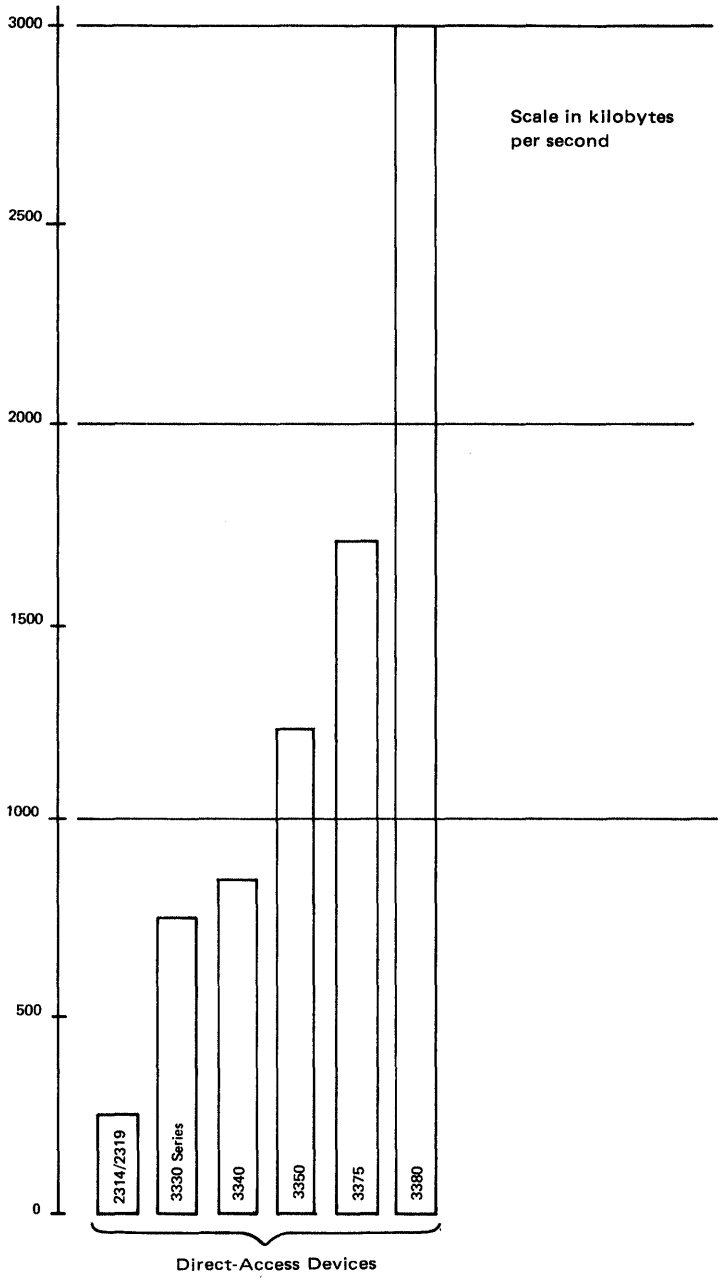
You can get the best performance using direct-access intermediate storage devices when:

- One work data set is assigned per actuator.
- The devices are of the same type.
- Two channel paths to devices exist.
- Input/output, and work data sets are on different channels and spindles.

For more information on direct-access work storage usage, see *DFSORT Application Programming: Guide.*

Device Data Transfer Rate

In general, the faster the data transfer rate of the storage device, the faster the sort. Therefore, the data in Figure 9 on page 36 should be taken into consideration when planning for your DFSORT applications.



Note: The data transfer rate of any processor is limited by the speed of the channel to which it is attached. For example, the 3880 Model 2 or 3 with the Speed Matching Buffer Feature permits attachment of the 3880 to systems with block multiplexor channels with data rates less than 3 megabytes per second.

Figure 9. Comparative Data Transfer Rates of Disk Work Storage Devices

Tape Work Storage Devices

The use of tape work storage devices prevents the use of the more efficient Blockset technique. The best performance using tape work storage is normally obtained when you use six or more tape drives.

Note: Frequency of tape direction changes, which occur during DFSORT workfile operations, will have more of an impact on the effective data rate of IBM 3480 Magnetic Tape Subsystems than on IBM 3420 Magnetic Tape Units. Because of this characteristic, performance comparisons between these tape units for intermediate storage cannot be reliably predicted and may vary widely.

For more detailed information on tape work storage usage, see *DFSORT Application Programming: Guide.*

Specifying Input/Output Data Set Characteristics Correctly

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, lead to inefficiency or program termination.

Data Set Size

When DFSORT has accurate information about the data set size, it can make the most efficient use of both main and intermediate work storage. Therefore, if the exact number of records to be sorted or merged is known, it should be specified as the value of the FILSZ parameter in the SORT, MERGE or OPTION control statement. When the exact number of records is not known, an estimated value, as accurate as possible, should be specified to obtain the best results for each application.

Variable-Length Records

When the input data set consists of variable-length records, the maximum, minimum, and average record lengths should be specified as accurately as possible in the RECORD statement.

Direct Access

For MVS, system performance is improved if storage is specified in cylinders rather than tracks. Storage on sortwork data sets will be reallocated in cylinders. The number of tracks per cylinder for direct-access devices is shown in Figure 10.

Device	Tracks per Cylinder
2314/2319	20
3330/3333	19
3340/3344	12
3350	30
3375	12
3380	15

Figure 10. Number of Tracks per Cylinder for Direct-Access Devices

If DFSORT has been installed with the option WRKSEC=YES and the data set is not virtual I/O, DFSORT will allocate secondary extents as required, even if not requested in the JCL.

For allocation of sort work data sets, it is normally adequate to allocate twice the space used by the input data set(s). Certain conditions may cause additional space requirements. These include:

- Long control words (more than 150 bytes)
- Using different device types or work data sets
- Use of 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 Figure 11 on page 39.

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 3 work storage tapes provided and file size not given.
Polyphase tape POLY	1 volume	Min=3 tape units	17 volumes	Used if 3 work storage tapes 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.

Figure 11. External Work Storage Requirements of the Various Tape Techniques

Key to Figure 11:

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.

Using JCL to Initiate DFSORT

You may enhance performance by initiating DFSORT by means of JCL instead of invoking it from a COBOL, assembler, or a PL/I program.

Using Options That May Enhance Performance

To obtain optimum performance, you should fine-tune the options specified at installation time and execution time.

Certain options may 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.

Other options may enhance performance, and should be used whenever possible. Several of these options are described below.

For more detailed information about options that affect performance, see *DFSORT Application Programming: Guide.*

COBEXIT

To take advantage of the COBOL II interface with DFSORT, and enhance performance thereby, specify COB2 in the COBEXIT parameter when running exits compiled with VS COBOL II.

FASTSRT

By specifying the VS COBOL II FASTSRT compile-time option, you can significantly reduce DFSORT processor time, EXCPs, and elapsed time. With FASTSRT, DFSORT input/output operations are more efficient because DFSORT rather than COBOL does the input and output (see Figure 12). For more details, see the VS COBOL II publications.

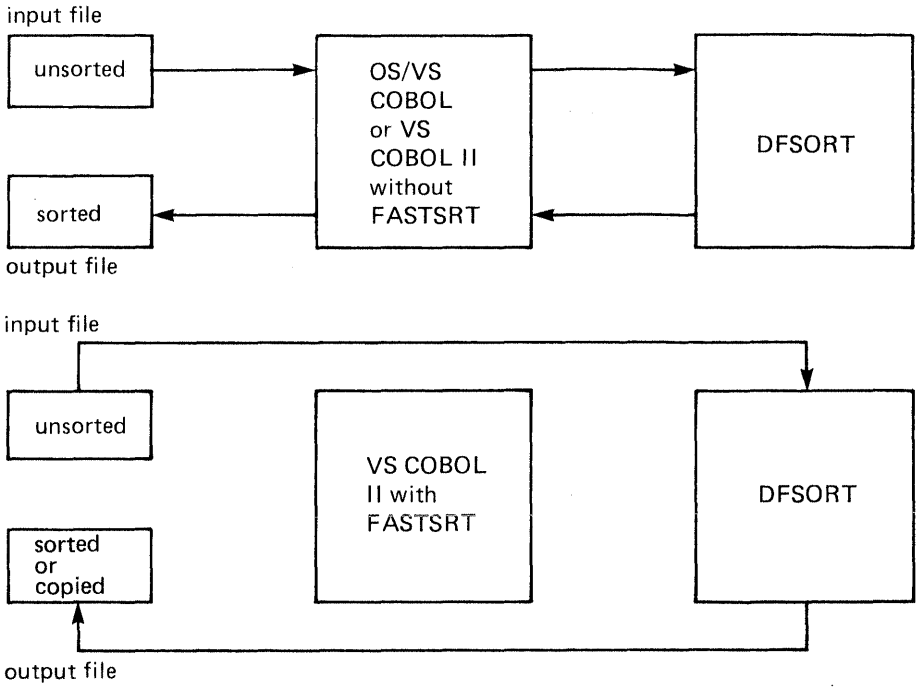


Figure 12. Performance is Improved with FASTSRT

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. Reducing the size of the input file may reduce processor and data transfer time.

- The INCLUDE and OMIT statements allow you to select records by comparing fields with constants and/or other fields.

- The STOPAFT option allows you to specify the maximum number of records that should be accepted for sorting or copying.
- The SKIPREC option allows you to skip records at the beginning of the input file for sorting or copying applications.

INREC and OUTREC

You can reduce the size of your records and thus make DFSORT more efficient by using the INREC statement to reformat the input records *before* processing.

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.

SUM

You can reduce processor and data transfer time by using SUM to add the contents of fields. The SUM statement works by adding the contents of fields defined in the statement whenever two records with equal control fields are found. The result is placed in one record while the other record is deleted, reducing in this way the number of records to be sorted or merged by DFSORT.

Note: For more details on program options available at execution time, see *DFSORT Application Programming: Guide*.

Index

A

ABEND parameter 12
access methods 17
ALTSEQ
  parameter 10, 27
  statement
    description 10
    example 13
    format 12
application development 32
ARESALL parameter
  ICEMAC macro 27
  OPTION statement 12
ARESINV parameter
  ICEMAC macro 27
  OPTION statement 12
ASCII 9, 17
assembler
  converting parameter list 20
  invoking from 18

B

balanced (BALN) 39
basic assembler language
  converting parameter list 20
  invoking from 18
block sizes 17
blocking 17, 33
Blockset techniques
  merge 34
  sort 34
BSAM 10
BSAM parameter 12
BUFFERS parameter 12

C

CHALT parameter
  ICEMAC macro 27
  OPTION statement 12
CHECK parameter
  ICEMAC macro 27
  OPTION statement 12
CKPT parameter 12, 39
COBEXIT parameter
  ICEMAC macro 27
  OPTION statement 12
  performace 40
COBOL
  exits 23
  invoking from 18
COBOL II
  exits 23
  invoking from 18
CODE parameter 12
collating sequences 9, 10
COND parameter 12
control fields
  collating sequences 9
  data format 32
  length 9
  location 32
  ordering 9
  performance 32
control statement parameters 11
control statements
  examples 13-14
  features 5-8
  list 10-11
conventional technique 34, 35
CTRx parameter 12
cylinders 38

D

data set characteristics
  input/output records 17
  size 37
DD statements 22
DEBUG statement
  description 10
  format 12
deleting records 23
designing applications 32
devices
  direct access 26, 35
  performance 35-37
  tape 26
  transferring data 35
direct access
  device types 26
  performance 36, 38
dynamic allocation
  DYNALLOC parameter 12
  DYNALOC parameter 28

E

EBCDIC
  collating sequence 9, 17
  modifying 10
  performance 32
efficiency
  using disk 35
  using tape 35
END statement
  description 10
  format 12
EQUALS parameter
  ICEMAC macro 28
  OPTION statement 12
ERET parameter 28
EXCPVR parameter 28
exits
  functions 23
  language requirements 23
  link-editing 23
extended parameter list
  converting 20
  overriding 20

F

FASTSRT 40
features 5-8
FIELDS parameter 12
FILES parameter 12
FILSZ parameter 12, 37
FLR-Blockset
  performance 31
  technique 34
FMTABEND parameter 12
FORMAT parameter 12

H

hardware requirements 25

I

ICEMAC macro 27
IGNCKPT parameter 28
INCLUDE statement
  description 10
  example 14
  format 12
  performance 40
initiating DFSORT 19
INPFIL statement 11
input/output
  characteristics for a merge 5, 17
  characteristics for a sort 5, 17
  data set identification 15
  error handling 23
  order of records 9
  performance 37
  record limitations 17
INREC statement
  description 10
  example 14
  format 12
  performance 41
inserting records 23
installation
  macro 27
  overriding options 22
  parameters 27-29
  SMP and SMP/E 27
intermediate storage
  dynamic allocation of 7
  performance 39
  requirements 26-27
INV parameter 28
invoking DFSORT 19
ISCII/ASCII 9, 17

J

JCL
  data set identification 15
  invoking from 22
  parameter 28
  performance 39

L

languages 23
LENGTH parameter 12
linkage conventions 23
LIST parameter
  ICEMAC macro 28
  OPTION statement 12

M

macro for installation options 27
main storage 33
MAINSIZE parameter 12
MAXLIM parameter 28
MERGE statement

description 10
format 12
merge techniques
Blockset 34
Conventional 35
messages
data set 12, 28
printing 12, 28
types 8
MINLIM parameter 28
MODS statement
description 10
format 12
MSGCON parameter 28
MSGDDN parameter
ICEMAC macro 28
OPTION statement 12
MSGPRT parameter
ICEMAC macro 28
OPTION statement 12
MVS/XA
operating system 25
region size 33
space benefits 6

N

NOABEND parameter 12
NOASSIST parameter 12
NOBLKSET parameter 12
NOCHALT parameter 12
NOCHECK parameter 12
NOEQUALS parameter 12
NOLIST parameter 12
NOOUTREL parameter 12
NOOUTSEC parameter 12
NOSTIMER parameter 12
NOVERIFY parameter 12
NOWRKREL parameter 12
NOWRKSEC parameter 12

O

OMIT statement
description 10
format 12
performance 40
operating systems 25
OPTION statement
description 10
example 13, 21
format 12
OS/VS 25
oscillating (OSL) 39
OUTFIL statement 11
OUTREC statement
description 10
examples 14
format 12
performance 41
OUTREL parameter 28
OUTSEC parameter 28
OVERRGN parameter 28
overriding
installaton options 22
parameter list 20-22

P

parameter list
converting to extended list 20
overriding 20-22
PARM field 19, 22
Peerage 34
performance 31-41
using disk 35
using tape 35
permanent storage requirements 25
PL/I 18, 19
polyphase(POLY) 39
program
control statements 11
description 23
distribution 27
exits 23
features 5-8
initiation 18
installation 27
performance 31-41
relationship to operating systems 25
programming languages 23

Q

QSAM 17

R

record
blocking 33
characteristics for a merge 5, 17
characteristics for a sort 5, 17
comparison 9
fixed-length 34
length 17, 33
limitations 17
SMF 18, 29
spanned 17
variable-length 34

RECORD statement
description 11
example 13
format 12
RESALL parameter
ICEMAC macro 28
OPTION statement 12
RESDNTx parameter 28
RESINV parameter
ICEMAC macro 28
OPTION statement 12

S

SIZE parameter
control statements 12
ICEMAC macro 28
SKIPREC parameter 12, 40
SMF parameter 18, 29
SMP 27
SMP/E 27
SORT statement
cataloged procedure 22
description 10
example 13, 21
format 13
sort techniques
Blockset 34
Conventional 34
Peerage 34
Vale 34
SORTCKPT DD statement 22
SORTCNTL DD statement
overriding installation defaults 22
overriding parameter list 20
passing control statements 5, 19
SORTDD parameter 12
SORTDIAG DD statement 22
SORTDKnn DD statement 22
SORTIN
OPTION statement 12
SORTIN DD statement 22
SORTINnn DD statement 22
SORTLIB DD statement 22
SORTOUT
OPTION statement 12
SORTOUT DD statement 22
SORTWKnn DD statement 22
statistical data collection 18
STIMER parameter 29
STOPAFT parameter 12, 40
storage
intermediate 26-27
main 33
performance 27
permanent 25
SUM statement
description 11
example 14
format 13
performance 41
SVC parameter 18, 29
System/370-XA sorting instructions
performance 6, 31

T

tape
device types 26
performance 37, 38-39
TMAXLIM parameter 29
tracks 38
transferring data 35
TYPE parameter 12

U

user-written routines
functions 23
language requirements 23
link-editing 23
performance 33
records 15

V

Vale 34
VERIFY parameter
ICEMAC macro 29
OPTION statement 12
VIO parameter 29
Virtual Storage Constraint Relief 6
VLR-Blockset
intermediate storage 17
technique 34
VLSHRT parameter 29
VS COBOL II
COBEXIT 40
exits 23
FASTSRT 40
invoking from 19
VSAM 17
VSCR 6

WRKSEC parameter 29, 38

W

work data sets
    dynamic allocation of 7
    performance 39
    requirements 26-27
WRKREL parameter 29

DFSORT
General Information
GC33-4033-12

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.

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

GC33-4033-12

old and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

d and tape | Please do not staple | Fold and tape

IBM®

DFSORT General Information (File No. S370-20) Printed in U.S.A. GC33-4033-12

DFSORT
General Information
GC33-4033-12

File No. S370-20

GC33-4033-12

Printed in U.S.A.