DFSORT

Application Programming Guide

Release 14

SC33-4035-20

DFSORT

Application Programming Guide

Release 14

SC33-4035-20

Note!

Before using this information and the product it supports, be sure to read the general information under “Appendix G. Notices” on page 601.

Twenty First Edition (July 2000)

This edition replaces and makes obsolete the previous edition, SC33-4035-19. Technical changes for this edition are summarized under “Summary of Changes” and are indicated by a vertical bar to the left of a change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

This edition applies to Release 14 of DFSORT, Program Number 5740-SM1, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

© Copyright International Business Machines Corporation 1973, 2000. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Preface . . . xi
About This Book. . . . xi
Required Publications . . . xii
DFSORT Publications . . . xii
DFSORT Library Softcopy Information . . . xiii
Related Publications . . . xiii
Referenced Publications . . . xiv
How to Send Your Comments . . . xv
Notational Conventions . . . xv

Summary of Changes . . . xix
Twenty-First Edition, July 2000 . . . xix
New Programming Support for Release 14 (PTFs — July, 2000) . . . xix
Twentieth Edition, March 1999 . . . xxi
New Programming Support for Release 14 (PTFs — March 1999) . . . xxi
Nineteenth Edition, September 1998 . . . xxi
New Programming Support for Release 14 . . . xxi

Chapter 1. Introducing DFSORT . . . 1
DFSORT Overview . . . 1
DFSORT on the Web . . . 3
DFSORT FTP Site . . . 3
Invoking DFSORT . . . 3
How DFSORT Works . . . 4
Operating Systems . . . 4
Control Fields and Collating Sequences . . . 4
Cultural Environment Considerations . . . 6
DFSORT Processing . . . 6
Input Data Sets—SORTIN and SORTINnn . . . 10
Output Data Sets—SORTOUT and OUTFIL . . . 10
Data Set Considerations . . . 11
Sorting or Copying Records . . . 11
Merging Records . . . 12
Data Set Notes and Limitations . . . 12
SmartBatch Pipe Considerations . . . 15
HFS File Considerations . . . 15
Installation Defaults . . . 16
Migrating to DFSORT from Other Sort Products . . . 21
DFSORT Messages and Return Codes . . . 22
Use Blockset Whenever Possible . . . 23

Chapter 2. Invoking DFSORT with Job Control Language . . . 25
Using the JCL . . . 25
Using the JOB Statement . . . 27
Using the EXEC Statement . . . 27
Specifying EXEC Statement Cataloged Procedures . . . 28
Specifying EXEC/DFSPARM PARM Options . . . 29
Aliases for PARM Options . . . 58
Using DD Statements . . . 59
Duplicate Ddnames . . . 61
Shared Tape Units . . . 61
System DD Statements . . . 62
Program DD Statements . . . 64

© Copyright IBM Corp. 1973, 2000

Chapter 3. Using DFSORT Program Control Statements 77
Using Program Control Statements 79
Control Statement Summary 80
Describing the Primary Task 80
Including or Omitting Records 80
Reformatting and Editing Records 80
Producing Multiple Output and Reports and Converting Records 81
Invoking Additional Functions and Options 81
Using Symbols 81
General Coding Rules 81
Continuation Lines 83
Inserting Comment Statements 84
Coding Restrictions 84
ALTSEQ Control Statement 85
Altering EBCDIC Collating Sequence—Examples 86
DEBUG Control Statement 87
Specifying Diagnostic Options—Examples 91
END Control Statement. 92
Discontinue Reading Control Statements—Examples 92
INCLUDE Control Statement 93
Relational Condition 95
Comparisons 95
Including Records in the Output Data Set—Comparison Examples 100
Substring Comparison Tests 102
Including Records in the Output Data Set—Substring Comparison Example 103
Bit Logic Tests 104
Method 1: Bit Operator Tests 104
Padding and Truncation 106
Including Records in the Output Data Set—Bit Operator Test Examples 106
Method 2: Bit Comparison Tests 107
Including Records in the Output Data Set—Bit Comparison Test Examples 108
Date Comparisons 109
Including Records in the Output Data Set—Date Comparisons 111
INCLUDE/OMIT Statement Notes. 112
INREC Control Statement 112
INREC Statement Notes 119
Reformatting Records Before Processing — Examples. 120
MERGE Control Statement 124
Specifying a MERGE or COPY—Examples 126
MODS Control Statement 127
Identifying User Exit Routines—Examples 130
OMIT Control Statement 130
Omitting Records from the Output Data Set—Example 133
OPTION Control Statement 134
Aliases for OPTION Statement Options 177
Specifying DFSORT Options or COPY—Examples 178
OUTFIL Control Statements. 182
OUTFIL Statements Notes 236
OUTFIL Features—Examples 239
OUTREC Control Statement 251
OUTREC Statement Notes 256
Reformatting the Output Record—Examples 257
RECORD Control Statement 260
Describing the Record Format and Length—Examples 263
SORT Control Statement. 264
SORT/MERGE Statement Notes 272

Specifying a SORT or COPY—Examples . . . 272
SUM Control Statement . . . 274
SUM Statement Notes . . . 275
Adding Summary Fields—Examples . . . 277

Chapter 4. Using Your Own User Exit Routines . . . 279
User Exit Routine Overview . . . 280
DFSORT Program Phases . . . 281
Functions of Routines at User Exits . . . 283
DFSORT Input/User Exit/Output Logic Examples . . . 283
Opening and Initializing Data Sets . . . 284
Modifying Control Fields . . . 284
Inserting, Deleting, and Altering Records . . . 285
Summing Records . . . 285
Handling Special I/O . . . 285
VSAM User Exit Functions . . . 286
Determining Action when Intermediate Storage Is Insufficient . . . 286
Closing Data Sets . . . 286
Terminating DFSORT . . . 286
Addressing and Residence Modes for User Exits . . . 286
How User Exit Routines Affect DFSORT Performance . . . 287
Summary of Rules for User Exit Routines . . . 287
Loading User Exit Routines . . . 288
User Exit Linkage Conventions . . . 288
Dynamically Link-Editing User Exit Routines . . . 289
Assembler User Exit Routines (Input Phase User Exits) . . . 290
E11 User Exit: Opening Data Sets/Initializing Routines . . . 290
E15 User Exit: Passing or Changing Records for Sort and Copy Applications 290
E16 User Exit: Handling Intermediate Storage Miscalculation . . . 293
E17 User Exit: Closing Data Sets . . . 294
E18 User Exit: Handling Input Data Sets . . . 294
E19 User Exit: Handling Output to Work Data Sets . . . 297
E61 User Exit: Modifying Control Fields . . . 298
Assembler User Exit Routines (Output Phase User Exits) . . . 299
E31 User Exit: Opening Data Sets/Initializing Routines . . . 300
E32 User Exit: Handling Input to a Merge Only . . . 300
E35 User Exit: Changing Records . . . 301
E37 User Exit: Closing Data Sets . . . 304
E38 User Exit: Handling Input Data Sets . . . 305
E39 User Exit: Handling Output Data Sets . . . 305
Sample Routines Written in Assembler . . . 306
E15 User Exit: Altering Record Length . . . 306
E16 User Exit: Sorting Current Records When NMAX Is Exceeded . . . 307
E35 User Exit: Altering Record Length . . . 308
E61 User Exit: Altering Control Fields . . . 308
COBOL User Exit Routines . . . 309
COBOL User Exit Requirements . . . 309
COBOL User Exit Routines (Input Phase User Exit) . . . 312
COBOL E15 User Exit: Passing or Changing Records for Sort . . . 312
COBOL User Exit Routines (Output Phase User Exit) . . . 318
COBOL E35 User Exit: Changing Records . . . 318
Sample Routines Written in COBOL . . . 324
COBOL E15 User Exit: Altering Records . . . 324
COBOL E35 User Exit: Inserting Records . . . 325
E15/E35 Return Codes and EXITCK . . . 327

Chapter 5. Invoking DFSORT from a Program . . . 331
Invoking DFSORT Dynamically . . . 331
What Are System Macro Instructions? . . . 331
Using System Macro Instructions . . . 331
Using JCL DD Statements . . . 332
Overriding DFSORT Control Statements from Programs . . . 332
Invoking DFSORT With the 24-Bit Parameter List . . . 333
Providing Program Control Statements . . . 333
Invoking DFSORT With The Extended Parameter List . . . 339
Providing Program Control Statements . . . 339
Writing the Macro Instruction . . . 343
Parameter List Examples . . . 343
Restrictions for Dynamic Invocation . . . 347
Merge Restriction . . . 347
Copy Restrictions . . . 347

Chapter 6. Using ICETOOL . . . 349
Overview . . . 350
ICETOOL/DFSORT Relationship . . . 351
ICETOOL JCL Summary . . . 351
ICETOOL Operator Summary . . . 352
Complete ICETOOL Examples . . . 353
Using Symbols . . . 353
Invoking ICETOOL . . . 354
Putting ICETOOL to Use . . . 354
Job Control Language for ICETOOL . . . 357
JCL Restrictions . . . 359
ICETOOL Statements . . . 359
General Coding Rules . . . 359
COPY Operator . . . 360
COPY Examples . . . 362
COUNT Operator . . . 364
COUNT Example . . . 365
DEFAULTS Operator . . . 365
DEFAULTS Example . . . 367
DISPLAY Operator . . . 369
Simple Report . . . 370
Tailored Report . . . 370
Sectioned Report . . . 371
DISPLAY Examples . . . 385
MODE Operator . . . 397
MODE Example . . . 398
OCCUR Operator . . . 398
Simple Report . . . 399
Tailored Report . . . 400
OCCUR Examples . . . 404
RANGE Operator . . . 406
RANGE Example . . . 408
SELECT Operator . . . 409
SELECT Examples . . . 412
SORT Operator . . . 414
SORT Examples . . . 416
STATS Operator . . . 418
STATS Example . . . 420
UNIQUE Operator . . . 420
UNIQUE Example . . . 421

VERIFY Operator 422
VERIFY Example 423
Calling ICETOOL from a Program 424
TOOLIN Interface 424
Parameter List Interface 424
ICETOOL Notes and Restrictions. 430
ICETOOL Return Codes 431

Chapter 7. Using Symbols for Fields and Constants 433
Field and Constant Symbols Overview. 433
DFSORT Example 434
SYMNAMES DD Statement. 436
SYMNOUT DD Statement 436
SYMNAMES Statements. 436
Comment and Blank Statements 436
Symbol Statements. 437
Keyword Statements 443
Using SYMNOUT to Check Your SYMNAMES Statements 445
Using Symbols in DFSORT Statements 446
SORT and MERGE. 447
SUM 447
INCLUDE and OMIT 447
INREC and OUTREC 448
OUTFIL 449
Using Symbols in ICETOOL Operators. 450
DISPLAY 451
OCCUR 451
RANGE 451
SELECT. 451
STATS, UNIQUE and VERIFY. 451
ICETOOL Example 451
Notes for Symbols 453

Chapter 8. Using Extended Function Support 455
Using EFS 456
Addressing and Residence Mode of the EFS Program 456
How EFS Works 457
DFSORT Program Phases 457
DFSORT Calls to Your EFS Program 458
What You Can Do with EFS. 463
Opening and Initializing Data Sets 464
Examining, Altering, or Ignoring Control Statements 464
Processing User-Defined Data Types with EFS Program User Exit Routines 466
Supplying Messages for Printing to the Message Data Set 466
Terminating DFSORT 466
Closing Data Sets and Housekeeping 466
Structure of the EFS Interface Parameter List 466
Action Codes 468
Control Statement Request List 469
Control Statement String Sent to the EFS program 469
Control Statement String Returned by the EFS Program 471
EFS Formats for SORT, MERGE, INCLUDE, and OMIT Control Statements 472
D1 Format on FIELDS Operand 473
D2 Format on COND Operand 473
Length of Original Control Statement 475
Length of the Altered Control Statement 475

EFS Program Context Area 475
Extract Buffer Offsets List 475
Record Lengths List 476
Information Flags 476
Message List 478
EFS Program Exit Routines 478
EFS01 and EFS02 Function Description 479
EFS01 User Exit Routine. 479
EFS02 User Exit Routine. 480
Addressing and Residence Mode of EFS Program User Exit Routines 482
EFS Program Return Codes You Must Supply 483
Record Processing Order 483
How to Request a SNAP Dump 486
EFS Program Example 486
DFSORT Initialization Phase: 486
DFSORT Termination Phase 488

Chapter 9. Improving Efficiency 489
Improving Performance 490
Design Your Applications to Maximize Performance 490
Directly Invoke DFSORT Processing 490
Plan Ahead When Designing New Applications. 491
Specify Efficient Sort/Merge Techniques 491
Specify Input/Output Data Set Characteristics Accurately 492
Use Sequential Striping 493
Use Compression 493
Use DFSMSrmm-Managed Tapes 493
Use SmartBatch Pipes 493
Use VIO in Expanded Storage. 493
Specify Devices that Improve Elapsed Time 494
Use Options that Enhance Performance 494
Use DFSORT's Fast, Efficient Productivity Features 496
Avoid Options that Degrade Performance. 497
Use Main Storage Efficiently 498
Tuning Main Storage 498
Releasing Main Storage 500
Allocate Temporary Work Space Efficiently 501
Direct Access Work Storage Devices 501
Virtual I/O for Work Data Sets 502
Tape Work Storage Devices 502
Use Hipersorting 503
Sort with Data Space 503
Use ICEGENER Instead of IEBGENER 504
ICEGENER Return Codes 506
Use DFSORT's Performance Booster for The SAS System 506
Use DFSORT's BLDINDEX Support. 507

Chapter 10. Examples of DFSORT Job Streams 509
Summary of Examples 509
Storage Administrator Examples 510
REXX Examples 510
Sort Examples. 511
Example 1. Sort with ALTSEQ 511
Example 2. Sort with OMIT, SUM, OUTREC, DYNALLOC and ZDPRINT 512
Example 3. Sort with ISCII/ASCII Tapes 513
Example 4. Sort with E15, E35, FILSZ, AVGRLEN and DYNALLOC 514

Example 5. Called sort with SORTCNTL, CHALT, DYNALLOC and FILSZ 516
Example 6. Sort with VSAM Input/Output, DFSPARM and Option Override 517
Example 7. Sort with COBOL E15, EXEC PARM, COBEXIT and MSGDDN 518
Example 8. Sort with Dynamic Link-Editing of Exits 520
Example 9. Sort with the Extended Parameter List Interface 521
Example 10. Sort with OUTFIL 524
Example 11. Sort with SmartBatch Pipes and OUTFIL SPLIT 526
Example 12. Sort with INCLUDE and LOCALE. 528
Example 13: Sort with HFS Files 529
Merge Examples 530
Example 1. Merge with EQUALS 530
Example 2. Merge with LOCALE and OUTFIL 531
Copy Examples 532
Example 1. Copy with EXEC PARMs, SKIPREC, MSGPRT and ABEND 532
Example 2. Copy with INCLUDE and VLSHRT. 533
ICEGENER Example 534
ICETOOL Example 535

Appendix A. Using Work Space 539
Introduction. 539
Hiperspace 539
Work Data Set Devices 540
DASD and Tape Devices 540
Number of Devices 540
Non-Synchronous Storage Subsystems 541
Allocation of Work Data Sets 541
Dynamic Allocation of Work Data Sets 542
Dynamic Over-Allocation of Work Space 544
JCL Allocation of Work Data Sets 545
DASD Capacity Considerations 546
Exceeding DASD Work Space Capacity 547
Tape Capacity Considerations 547
Exceeding Tape Work Space Capacity. 547

Appendix B. Specification/Override of DFSORT Options 549
Main Features of Sources of DFSORT Options 550
DFSPARM Data Set 550
EXEC Statement PARM Options 550
SORTCNTL Data Set 550
SYSIN Data Set 550
Parameter Lists 550
Override Tables 551
Directly Invoked DFSORT 551
Notes to Directly Invoked DFSORT Table. 560
Program Invoked DFSORT with the Extended Parameter List 560
Notes to Extended Parameter List Table 570
Program Invoked DFSORT with the 24-Bit Parameter List 570
Notes to 24-Bit List Table 580

Appendix C. Data Format Descriptions. 581

Appendix D. EBCDIC and ISCII/ASCII Collating Sequences 589
EBCDIC 589
ISCII/ASCII 591

Appendix E. DFSORT Abend Processing 595

Checkpoint/Restart 595
DFSORT Abend Categories. 596
Abend Recovery Processing for Unexpected Abends 596
Processing of Error Abends with A-Type Messages 597
CTRx Abend processing 597

Appendix F. Locales Supplied with C/370. 599

Appendix G. Notices 601
Programming Interface Information 602
Trademarks. 602

Summary of Changes 603
New Programming Support for Release 13 (PTFs after April, 1996) 603
Additional Year 2000 Features. 603
OS/390 Registration 603
New Programming Support for Release 13 (PTFs - April, 1996) 603
Year 2000 Features. 603
Performance Improvements for FLR and VLR Blockset Sorts 603
Floating Point for SUM 604
Security Improvements 604
EXCPVR Processing Removed 604
New Device Support for Release 13 (PTFs). 604
New Programming Support for Release 13 605
DFSORT's Performance Booster for The SAS** System 605
Dynamic Hipersorting 605
Performance 605
OUTFIL Processing. 605
National Language Support 606
ICETOOL Enhancements 606
INCLUDE/OMIT Substring Search 607
SMF Type-16 Record Enhancements 607
Other Enhancements 607
New Programming Support for Release 12 (PTFs) 608
New Device Support for Release 12 (PTFs). 609

Index 611

Preface

This book is intended to help you to sort, merge, and copy data sets using DFSORT. This book is not designed to teach you how to use DFSORT, but is for programmers who already have a basic understanding of DFSORT, and need a task-oriented guide and reference to its functions and options. If you are a new user, then you should read *Getting Started with DFSORT* first.*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.

About This Book

The various sections of this book present related information grouped according to tasks 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. The appendixes provide specific information about various topics.

- "Chapter 1. Introducing DFSORT" on page 1, presents an overview of DFSORT, explaining what you can do with DFSORT and how you invoke DFSORT processing. It describes how DFSORT works, discusses data set formats and limitations, and explains the defaults that might have been modified during installation at your site.
- "Chapter 2. Invoking DFSORT with Job Control Language" on page 25, explains how to use job control language (JCL) to run 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 77, 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, how to edit data by reformatting and summing records, and how to produce multiple output data sets and reports. It explains how to write statements that direct DFSORT to use your own routines during processing.
- "Chapter 4. Using Your Own User Exit Routines" on page 279, describes how to use DFSORT's program exits to call your own routines during program processing. You can write routines to delete, insert, alter, and summarize records, and you can incorporate your own error-recovery routines.
- "Chapter 5. Invoking DFSORT from a Program" on page 331, 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 6. Using ICETOOL" on page 349, describes how to use the multi-purpose DFSORT utility ICETOOL. It explains the JCL and operators you can use to perform a variety of tasks with ICETOOL.
- "Chapter 7. Using Symbols for Fields and Constants" on page 433 explains how to define symbols and use them in DFSORT control statements and ICETOOL operators.
- "Chapter 8. Using Extended Function Support" on page 455, 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 processing.

© Copyright IBM Corp. 1973, 2000

- “Chapter 9. Improving Efficiency” on page 489, recommends ways with which you can maximize DFSORT processing efficiency. This chapter covers a wide spectrum of improvements you can make, from designing individual applications for efficient processing at your site to using DFSORT features such as Hipersorting, dataspace sorting, and ICEGENER.
- “Chapter 10. Examples of DFSORT Job Streams” on page 509, contains annotated example job streams for sorting, merging, and copying records.
- “Appendix A. Using Work Space” on page 539, explains main storage considerations and how to estimate the amount of intermediate storage you might require when sorting data.
- “Appendix B. Specification/Override of DFSORT Options” on page 549, contains a series of tables you can use to find the order of override for similar options that are specified in different sources.
- “Appendix C. Data Format Descriptions” on page 581, gives examples of the assembled data formats used with IBM System/390.
- “Appendix D. EBCDIC and ISCII/ASCII Collating Sequences” on page 589, lists the collating sequences from low to high order for EBCDIC and ISCII/ASCII characters.
- “Appendix E. DFSORT Abend Processing” on page 595, describes the ESTAE recovery routine for processing abends, and the Checkpoint/Restart facility.
- “Appendix F. Locales Supplied with C/370” on page 599, lists the locales provided by the C/370 product.
- “Appendix G. Notices” on page 601, includes the notices, Programming Interface information, and the trademark list.

Required Publications

For up-to-date descriptions of all of the books that support OS/390, refer to the *OS/390 Information Roadmap*, GC28-1727.

You should be familiar with the information presented in the following publications:

Short Title	Publication	Order Number
JCL Reference	*MVS/ESA JCL Reference* (for MVS/ESA SP Version 5)	GC28-1479
	MVS/ESA JCL Reference (for MVS/ESA SP Version 4)	GC28-1654
JCL User’s Guide	*MVS/ESA JCL User’s Guide* (for MVS/ESA SP Version 5)	GC28-1473
	MVS/ESA JCL User’s Guide (for MVS/ESA SP Version 4)	GC28-1653

DFSORT Publications

The *DFSORT Application Programming Guide* is a part of a more extensive DFSORT library. The additional books in the library are listed below.

Task	Publication Title	Order Number
Planning For and Customizing DFSORT	*DFSORT Installation and Customization Release 14*	SC33-4034
Learning to Use DFSORT Panels	*DFSORT Panels Guide*	GC26-7037

Task	Publication Title	Order Number
Learning to Use DFSORT	*Getting Started with DFSORT Release 14*	SC26-4109
Quick Reference	*DFSORT Reference Summary Release 14*	SX33-8001
Diagnosing Failures and Interpreting Messages	*DFSORT Messages, Codes and Diagnosis Guide Release 14*	SC26-7050
Tuning DFSORT	*DFSORT Tuning Guide Release 14*	SC26-3111

You can order a complete set of DFSORT publications with the order number SBOF-1243, except for *DFSORT Licensed Program Specifications* GC33-4032, which must be ordered separately.

DFSORT Library Softcopy Information

A softcopy version of the DFSORT library is available on two CD-ROMs as shown in the table that follows. Each of the CD-ROMs contains all of the DFSORT books for Release 13 and Release 14 with the exception of the *DFSORT Reference Summary.*

Order Number	Title
SK2T-6700	*IBM Online Library: OS/390 Collection*

Related Publications

For up-to-date descriptions of all of the books that support OS/390, refer to the *OS/390 Information Roadmap*, GC28-1727.

In the course of programming a DFSORT application, you may need access to the additional publications listed in the table below.

Short Title	Publication	Order Number
Access Method Services for the Integrated Catalog Facility	*DFSMS/MVS Version 1 Release 4: Access Method Services for the Integrated Catalog Facility*	SC26-4906
	MVS/ESA Integrated Catalog Administration: Access Method Services Reference	SC26-4500
Checkpoint/Restart	*DFSMS/MVS Version 1 Release 1: Checkpoint/Restart*	SC26-4907
	MVS/ESA Checkpoint/Restart User's Guide Version 3	SC26-4503
Magnetic Tape Labels	*DFSMS/MVS Version 1 Release 2: Using Magnetic Tape Labels and File Structure*	SC26-4923
	MVS/ESA Magnetic Tape Labels and File Structure Administration	SC26-4511
Planning Guide	*DFSMS/MVS Version 1 Release 4: Planning for Installation*	SC26-4919

Related Publications

For more information on using DFSORT with COBOL, see the Programmer's Guide describing the compiler version available at your site.

Referenced Publications

Within the text of this document, references are made to the following publications:

Short Title	Publication Title	Order Number
Advanced Services for Data Sets	*DFSMS/MVS Version 1 Release 4: Using Advanced Services for Data Sets*	SC26-4921
	MVS/ESA System—Data Administration	SC26-4515
Application Development Guide	*MVS/ESA Programming: Assembler Services Guide* (for MVS/ESA SP Version 5)	GC28-1466
	MVS/ESA Application Development Guide: Assembler Language Programs (for MVS/ESA SP Version 4)	GC28-1644
Application Development Macro Reference	*MVS/ESA Programming: Assembler Services Reference* (for MVS/ESA SP Version 5)	GC28-1474
	MVS/ESA Application Development Reference: Services for Assembler Language Programs (for MVS/ESA SP Version 4)	GC28-1642
Assembler Reference	*High Level Assembler Language Reference*	SC26-4940
SmartBatch pipe	*SmartBatch Overview*	GC28-1627
	SmartBatch Users Guide and Reference	GC28-1640
Messages, Codes and Diagnosis	*DFSORT Messages, Codes and Diagnosis Guide*	SC26-7050
Getting Started	*Getting Started with DFSORT*	SC26-4109
Installation and Customization	*DFSORT Installation and Customization*	SC33-4034
Panels Guide	*DFSORT Panels Guide*	GC26-7037
JCL Reference	*MVS/ESA JCL Reference* (for MVS/ESA SP Version 5)	GC28-1479
	MVS/ESA JCL Reference (for MVS/ESA SP Version 4)	GC28-1654
JCL User's Guide	*MVS/ESA JCL User's Guide* (for MVS/ESA SP Version 5)	GC28-1473
	MVS/ESA JCL User's Guide (for MVS/ESA SP Version 4)	GC28-1653

Short Title	Publication Title	Order Number
Macro Instructions for Data Sets	*DFSMS/MVS Version 1 Release 4: Macro Instructions for Data Sets*	SC26-4913
	MVS/ESA Data Administration: Macro Instruction Reference	SC26-4506
	MVS/ESA VSAM Administration: Macro Instruction Reference	SC26-4517
Using Locales	*IBM C/C++ for MVS/ESA C/MVS Programming Guide V3 R1*	SC09-2062
	IBM C/C++ for MVS/ESA C++/MVS Programming Guide V3 R1	SC09-1994
	IBM SAA AD/Cycle C/370 V1.2 Programming Guide for LE/370 V1.3	SC09-1840
	Language Environment for MVS & VM Installation and Customization Guide	SC26-4817
Using Data Sets	*DFSMS/MVS Version 1 Release 4: Using Data Sets*	SC26-4922
	MVS/ESA Data Administration Guide	SC26-4505
	MVS/ESA VSAM Administration Guide	SC26-4518
Using DFSMSrmm	*OS/390 DFSMSrmm Reporting*	SC26–7335

How to Send Your Comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other DFSORT documentation:

- Send your comments by e-mail to:
  - IBMLink from US: starpubs@us.ibm.com
  - IBMLink from Canada: STARPUBS at TORIBM
  - IBM Mail Exchange: USIB3VVD at IBMMAIL
  - Internet: starpubs@us.ibm.com

  Be sure to include the name of the book, the part number of the book, version and product name, and if applicable, the specific location of the text you are commenting on (for example, a page number or a table number).
- Fill out one of the forms at the back of this book and return it by mail or by giving it to an IBM representative. If the form has been removed, address your comments to IBM Corporation, RCF Processing Department G26/050, 5600 Cottle Road, San Jose, California 95193-0001, U.S.A.

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.

```
►►──Required──┬──────────┬──────────────────────────────────────────►◄
              └─Optional─┘
```

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

```
►►──Operator──┬─Required Choice 1─┬──┬──────────────────────────┬──────►◄
              ├─Required Choice 2─┤  └──┬─Optional Choice 1─┬────┘
              └─Required Choice 3─┘     └─Optional Choice 2─┘
```

  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 first example below), to specify more than one choice at a time from the same stack (see the second example below), or to nest parentheses (see the third example below).

```
      ┌──,──┐                 ┌──,──────────┐                ┌──(──┐
►──────┴─a,b,c─────►   ►──┬────┴─Choice-1────┬──►   ►──────────┴─────►
                          ├─Choice-2─────────┤
                          └─Choice-3─────────┘
```

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.

```
──►◄
```

Summary of Changes

Twenty-First Edition, July 2000

New Programming Support for Release 14 (PTFs — July, 2000)

Larger Tape Block Sizes with OS/390 R10

DFSORT can now use tape data sets with block sizes greater than 32760 bytes for input and output, providing improved performance and tape utilization.

DFSORT can now select system-determined optimum block sizes greater than 32760 bytes for tape output data sets, if allowed by the BLKSZLIM value in effect. Installation and run-time options SDB=INPUT (the new IBM-supplied default), SDB=LARGE (new), SDB=YES (or its alias SDB=SMALL) and SDB=NO allow you to control DFSORT's use of system-determined block sizes, including block sizes greater than 32760 bytes for tape output data sets.

DFSORT's ICEGENER, like IEBGENER, will use SDB=value parameters you specify.

New Device Support

DFSORT now supports FICON channels.

DFSORT now recognizes the IBM 2105 control unit and provides optimizations for it appropriate for cached control units, such as cache fast write for work data sets.

DFSORT now treats work data sets on emulated 3390 model 9 devices as desirable choices with respect to performance. DFSORT continues to treat work data sets on real 3990 model 9 devices as undesirable choices with respect to performance.

DFSMSrmm-Managed Tapes

DFSORT can now automatically obtain accurate input file size information for DFSMSrmm-managed tapes. This can result in improved sort performance and more accurate dynamic workspace allocation.

DFSORT can now automatically obtain input and output attributes such as RECFM, LRECL and BLKSIZE for DFSMSrmm-managed tapes.

VSAM Processing

DFSORT now supports the VSAM extended addressability function for extended format VSAM data sets, which provides the capability of VSAM data sets larger than four gigabytes.

DFSORT now allows an empty VSAM input data set to be accepted and processed as having zero records. New installation option VSAMEMT=YES/NO and run-time options VSAMEMT and NVSAMEMT allow you to control the processing of empty VSAM input data sets.

DFSORT now allows a VSAM output data set defined with REUSE to be processed as a NEW or MOD data set. New installation option RESET=YES/NO and run-time options RESET and NORESET allow you to control the processing of VSAM output data sets.

© Copyright IBM Corp. 1973, 2000

DFSORT now allows a VSAM data set defined with REUSE to be sorted in-place. New installation option VSAMIO=YES/NO and run-time options VSAMIO and NOVSAMIO allow you to control the use of the same VSAM data set for input and output.

HFS Files

DFSORT now supports Hierarchical File System (HFS) files for input and output.

Productivity and Easier Migration from other Sort Products

By default, DFSORT now uses the L3, OUTREC or INREC record length as the SORTOUT LRECL when the SORTOUT LRECL is unavailable. New installation option SOLRF=YES/NO and run-time options SOLRF and NOSOLRF allow you to control the use of the reformatted record length for the SORTOUT LRECL.

DFSORT now provides enhanced capabilities for OUTFIL, including:

- Padding short records with blanks, by default, when converting variable-length records to fixed-length records.
- Creation of edited counts and subcounts in trailers.

DFSORT now provides enhanced capabilities for INREC and OUTREC that were previously only available with OUTFIL OUTREC, including:

- Hexadecimal display.
- Transformation of dates.
- Editing of numeric fields.
- Lookup and change.

DFSORT now provides enhanced capabilities for INREC, OUTREC and OUTFIL OUTREC, including:

- Conversion of numeric fields to other numeric formats.
- Editing of transformed dates.
- Conversion of transformed dates to numeric formats.
- Creation of sequence numbers.

DFSORT now allows INCLUDE/OMIT fields that extend beyond variable-length records (that is, short fields) to be compared as if they were padded with binary zeros. This allows both the short and non-short fields to be compared. New installation option VLSCMP=YES/NO and run-time options VLSCMP and NOVLSCMP allow you to control the use of binary zero padding for short control fields.

DFSORT now allows -0 and +0 to be treated as unsigned (that is, the same) for processing by DFSORT statements INCLUDE, INREC, MERGE, OMIT, OUTFIL, OUTREC and SORT. New installation option SZERO=YES/NO and run-time options SZERO and NOSZERO allow you to select signed or unsigned zero processing for these statements.

DFSORT now allows -0 and +0 to be treated as unsigned (that is, the same) for processing by ICETOOL operators DISPLAY, OCCUR, SELECT and UNIQUE. New option UZERO allows you to select unsigned zero processing for these operators.

DFSORT now always determines or assigns the record type (F or V) when the input is VSAM, or when an E15 or E32 exit supplies all of the input records.

DFSORT now allows an LRECL greater than 80 for SYSIN, SORTCNTL and DFSPARM; only the first 80 bytes of each control statement are used.

DFSORT now allows variable-length output records that are longer than the output data set LRECL to be truncated to the LRECL. New installation option VLLONG=YES/NO and run-time options VLLONG and NOVLLONG allow you to control the truncation of long variable-length output records.

DFSORT now accepts and processes the following as EXEC/DFSPARM PARM options: COBEXIT=COB1/COB2, NOVERIFY, NOVLSHRT, NZDPRINT, OVFLO=RC0/RC4/RC16, PAD=RC0/RC4/RC16, SPANINC=RC0/RC4/RC16, TRUNC=RC0/RC4/RC16, VERIFY, VLSHRT and ZDPRINT.

DFSORT now accepts and processes the following as OPTION statement options: EXITCK=STRONG/WEAK, WRKREL and WRKSEC.

DFSORT now accepts and processes Y2PAST=s/f as a SORT and MERGE statement option.

DFSORT now accepts, and processes or ignores as appropriate, many more aliases for DFSORT options.

DFSORT now accepts and processes the following as 24-Bit Parmlist control statement entry codes: X’10’, X’16’, X’18’ and X’20’ through X’29’.

Twentieth Edition, March 1999

New Programming Support for Release 14 (PTFs — March 1999)

New Generation of Year 2000 Features

New Y2T, Y2U, Y2V, Y2W, Y2X, and Y2Y formats handle CH, ZD, and PD full dates like yymmdd, yyddd, yymm, yyq, mmddyy, dddyy, mmyy, and qyy, and their special indicators like zeros and nines, according to the specified fixed or sliding century window.

You can use these new formats in the SORT, MERGE, INCLUDE, OMIT, and OUTFIL statements to sort, merge, compare, and transform full dates.

Nineteenth Edition, September 1998

New Programming Support for Release 14

Symbols for Fields and Constants

DFSORT now provides a simple and flexible method for using symbols in DFSORT and ICETOOL statements. You can define and use a symbol for any field or constant that is recognized in a DFSORT control statement or ICETOOL operator. This makes it easy to create and reuse collections of symbols (that is, mappings) for your frequently used data.

In addition, you can obtain and use collections of DFSORT symbols created specifically for data associated with other products (for example, RACF, DFSMSrmm and DCOLLECT) or by your site.

DFSORT symbols can increase your productivity by automatically providing the positions, lengths and formats of the fields and the literals, numbers, and bit flags of the constants, associated with the particular records you are processing with DFSORT or ICETOOL.

Improvements in Performance, Capacity and Storage Usage

Blockset copy and merge applications can now use storage above 16MB virtual, providing improved performance and virtual storage constraint relief.

Blockset copy and merge modules will now reside above 16MB virtual, providing virtual storage constraint relief.

DFSORT can now handle a significantly larger number of INCLUDE and OMIT conditions.

DFSORT can now handle a significantly larger number of SUM fields.

The upper limit for the number of JCL and dynamically allocated work data sets that can be specified and used by DFSORT's Blockset technique has been raised from 100 to 255. The use of more work data sets increases the maximum amount of data DFSORT can process in a single sort application. Any valid ddname of the form SORTWKdd or SORTWKd can now be used for DASD work data sets (for example, SORTWK01, SORTWKC3, SORTWK2, SORTWK#5, SORTWKA, SORTWKXY and so on).

The upper limit for the number of input data sets that can be specified and used for a Blockset merge application has been raised from 16 to 100. The use of more merge input data sets increases the maximum amount of data DFSORT can process in a single merge application.

Time-of-Day Option Controls

New time-of-day installation modules (ICETD1-4) allow different sets of installation defaults to be used, based on the day and time DFSORT applications run. Each environment installation module (ICEAM1-4) can enable one or more time-of-day installation modules. This capability allows new levels of control for installation defaults. For example, larger storage, hiperspace and data space limits could be used only for batch program-invoked DFSORT applications that run off-shift during the week, and all weekend.

Repackaging

The product has been repackaged to simplify installation and customization:

- IBM's DFSORT, DFSMSdfp, and MVS/DFP teams have simplified the process of replacing IEBGENER with ICEGENER. You now only need to apply a DFSMS or DFP PTF that supplies an alias of ″IEBGENR″ for IEBGENER and place ICEGENER with an alias of ″IEBGENER″ ahead of IEBGENER in the system's search order for programs.
- The number of FMIDs has been reduced from 10 to 3.
- The number of libraries required to install DFSORT has been reduced from 40 to 26.
- DFSORT R14 now supports a single installation of the product for both resident and nonresident features. This allows you to decide how to use DFSORT independent of the installation method, thus reducing the number of decisions you have to make at installation time.
- All FMIDs in DFSORT R14 can be installed together, including the FMIDs for both English and Japanese messages and panels.

OUTFIL Processing Enhancements

OUTFIL now supports creation of multiple output records using the fields of the input record. This allows you to split each record into pieces, include a field in more than one record, include different fields in different records, and more.

OUTFIL now supports processing of variable-length input records which are too short to contain all specified OUTFIL OUTREC fields. OUTFIL's new VLFILL=byte operand can be used to replace missing bytes in OUTFIL OUTREC fields with the specified fill byte so the filled fields can be processed.

ICETOOL Enhancement

A new DISCARD(savedd) operand of ICETOOL's SELECT operator allows you to save the records that are not selected, in the savedd data set. Thus, in one pass, you can create an outdd data set with the records that meet your specified criteria, and a savedd data set with the records that do not meet your specified criteria. DISCARD(savedd) can be used to save the records discarded by ALLDUPS, NODUPS, HIGHER(x), LOWER(y), EQUAL(v), FIRST or LAST.

Installation and Run-Time Option Enhancements

A new p% value for the EXPRES, EXPOLD, and EXPMAX installation options and the HIPRMAX installation and run-time options is now available. p% can be used to vary the limit DFSORT calculates for the corresponding option as a percentage of the configured expanded storage on the system at run time. If the configured expanded storage on a system changes, p% will cause a corresponding change in the run-time limit calculated for the corresponding option. When sharing DFSORT installation options between systems, such as in a sysplex, p% can be used to tailor the limit DFSORT calculates for the corresponding option to the system on which the application runs.

A new SPANINC installation and run-time option allows you to specify what you want DFSORT to do if it detects incomplete spanned records. This gives you control over the action (continue by eliminating incomplete spanned records and recovering valid records, or terminate), type of message (informational or error) and return code (0, 4 or 16) for incomplete spanned records.

A new OVFLO installation and run-time option allows you to specify what you want DFSORT to do when BI, FI, PD or ZD summary fields overflow. This gives you control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for summary overflow.

A new PAD installation and run-time option allows you to specify what you want DFSORT to do when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL. This gives you control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for LRECL padding.

A new TRUNC installation and run-time option allows you to specify what you want DFSORT to do when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL. This gives you control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for LRECL truncation.

The IBM-supplied default for ICEMAC option DSA has been changed from 16MB to 32MB.

The IBM-supplied default for ICEMAC option GENER has been changed from IEBGENER to IEBGENR.

The maximum value for ICEMAC option OVERRGN has been changed from 64KB to 16128KB.

Other Enhancements

New messages ICE178I and ICE179A provide information about reallocation of VIO work data sets.

The option-in-effect messages (ICE127I-ICE133I) are now printed for Blockset copy and merge applications.

The user exit address constant can now be passed to E32 user exits for Blockset merge applications.

Null segments in variable spanned input records are now processed by DFSORT and no longer result in termination. A null segment means that there are no more segments in the block.

OS/390 and MVS/ESA Only

DFSORT Release 14 only supports the OS/390 and MVS/ESA environments. MVS/XA and VIRTDSP processing for MVS/XA are no longer supported.

Chapter 1. Introducing DFSORT

DFSORT Overview 1
DFSORT on the Web 3
DFSORT FTP Site 3
Invoking DFSORT 3
How DFSORT Works 4
Operating Systems 4
Control Fields and Collating Sequences 4
Cultural Environment Considerations 6
DFSORT Processing 6
Input Data Sets—SORTIN and SORTINnn 10
Output Data Sets—SORTOUT and OUTFIL 10
Data Set Considerations 11
Sorting or Copying Records 11
Merging Records 12
Data Set Notes and Limitations 12
General Considerations 12
Padding and Truncation 13
QSAM Considerations 14
VSAM Considerations 14
SmartBatch Pipe Considerations 15
HFS File Considerations 15
Installation Defaults 16
Migrating to DFSORT from Other Sort Products 21
DFSORT Messages and Return Codes 22
Use Blockset Whenever Possible 23

DFSORT Overview

This chapter introduces IBM DFSORT Licensed Program 5740-SM1.

DFSORT is intended to run in problem state and in a user key (that is, key 8 or higher).

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 are identically sorted for the information you will use to merge them and that they are in the same order required for output. You can merge up to 100 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 application. 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

© Copyright IBM Corp. 1973, 2000

work only with 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. INREC and OUTREC statements support a wide variety of tasks including:

- Insertion of blanks, zeros, strings and sequence numbers before, between, and after the input fields in the reformatted records.
- Sophisticated editing and conversion capabilities, such as hexadecimal display, conversion of numeric values from one format to another, and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.
- Transformation of various forms of two-digit year dates to various forms of four-digit year dates using a specified fixed or sliding century window.
- Selection of a character or hexadecimal string from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).

You can sum numeric information from many records into one record 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 create one or more output data sets for a sort, copy, or merge application from a single pass over one or more input data sets by using OUTFIL control statements. You can use multiple OUTFIL statements, with each statement specifying the OUTFIL processing to be performed for one or more output data sets. OUTFIL processing begins after all other processing ends (that is, after processing for exits, options, and other control statements). OUTFIL statements support a wide variety of output data set tasks, including:

- Creation of multiple output data sets containing unedited or edited records from a single pass over one or more input data sets.
- Creation of multiple output data sets containing different ranges or subsets of records from a single pass over one or more input data sets. In addition, records that are not selected for any subset can be saved in a separate output data set.
- Conversion of variable-length record data sets to fixed-length record data sets.
- Insertion of blanks, zeros, strings and sequence numbers before, between, and after the input fields in the reformatted records.
- Sophisticated editing and conversion capabilities, such as hexadecimal display, conversion of numeric values from one format to another, and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.
- Transformation of two-digit year dates to various forms of four-digit year dates using a specified fixed or sliding century window.

- Selection of a character or hexadecimal string for output from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).
- Highly detailed three-level (report, page, and section) reports containing a variety of report elements you can specify (for example, current date, current time,page number, character strings, and blank lines) or derive from the input records (for example, character fields; edited numeric input fields; record counts; and edited totals, maximums, minimums, and averages for numeric input fields).
- Creation of multiple output records from each input record, with or without intervening blank output records.

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 run time 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. However, keep in mind that the extensive editing capabilities provided by the INCLUDE, OMIT, INREC, OUTREC, SUM, and OUTFIL statements can eliminate the need to write user exit routines. You can write your own routines to correct I/O errors that DFSORT does not 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 or to provide alternate sequence support for user-defined data.

You can define and use a symbol for any field or constant that is recognized in a DFSORT control statement or ICETOOL operator. This makes it easy to create and reuse collections of symbols (that is, mappings) representing information associated with various record layouts. See “Chapter 7. Using Symbols for Fields and Constants” on page 433.

DFSORT on the Web

For articles, online books, news, tips, techniques, examples, and more, visit the DFSORT/MVS home page at:

`http://www.ibm.com/storage/dfsort/`

DFSORT FTP Site

You can obtain DFSORT articles and examples via anonymous FTP to:

`ftp.software.ibm.com/storage/dfsort/mvs/`

Invoking DFSORT

You can invoke DFSORT processing in the following ways:

- 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 5. Invoking DFSORT from a Program” on page 331. (Foreground TSO users can also use the name of the program to invoke DFSORT.)
- With a program written in basic assembler language using a system macro instruction. See “Chapter 5. Invoking DFSORT from a Program” on page 331.

- 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 the ICETOOL utility. See "Chapter 6. Using ICETOOL" on page 349.
- With interactive panels supported under ISPF and ISMF. See *Panels Guide* for complete information.

Note: DFSORT Panels supports interactive panels for a subset of the functions available with DFSORT. Although interactive panels for functions such as DFSORT's Year 2000 Features and the OUTFIL Statement are **not** provided, you can use Free Form Entry panels to specify complete DFSORT Control Statements for these functions.

In this book, the term *directly invoked* means that DFSORT is not initiated from another program. The term *program invoked* means that DFSORT is initiated from another program.

How DFSORT Works

This section contains a list of the operating systems supported by DFSORT and an explanation of how DFSORT uses control fields and collating sequences to sort, merge, and copy the records of a data set.

The Blockset technique is DFSORT's most efficient technique for sorting, merging and copying data sets. DFSORT uses the Blockset technique whenever possible to take advantage of its highly optimized internal algorithms and efficient utilization of IBM hardware. If Blockset cannot be used, DFSORT uses another of its techniques — Peerage/Vale or Conventional.

Operating Systems

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

- OS/390
- MVS/ESA

Additionally, DFSORT operates on these systems as a guest under VM/ESA 2.1.

DFSORT is compatible with all of the IBM processors supported by OS/390 and MVS/ESA. In addition to any device supported by these operating systems for program residence, DFSORT also operates with any device QSAM or VSAM uses for input or output.

Control Fields and Collating Sequences

You define *control fields* to identify the information you want DFSORT to sort or merge. When thinking of the contents of your data sets, you probably think of names, dates, account numbers, or similar pieces of useful information. For example, when sorting your data sets, you might choose to arrange 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.

To arrange your records in a specific order, identify 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, 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 (see Figure 1).

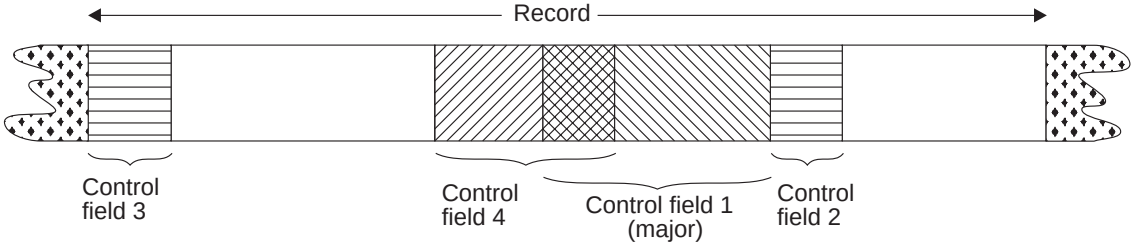


Figure 1. Control Fields. Control fields may overlap, or be contained within other control fields.

DFSORT offers several standard *collating sequences*. You can choose to arrange your records according to these standard collating sequences or according to a collating sequence defined in the active locale. 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 when sorting or merging records. If locale processing is in effect, DFSORT will use the collating sequence defined in the active locale.

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 collating of national characters. An alternate collating

sequence can be defined during installation with the ICEMAC ALTSEQ option, or you can define it yourself at run-time with the ALTSEQ program control statement. You can also specify a modified collating sequence with an E61 user exit or with an EFS program.

You can specify the LOCALE installation or run-time option to use an active locale's collating rules.

Cultural Environment Considerations

DFSORT's collating behavior can be modified according to your cultural environment. Your cultural environment is defined to DFSORT using the X/Open** locale model. A locale is a collection of data grouped into categories that describes the information about your cultural environment.

The collate category of a locale is a collection of sequence declarations that defines the relative order between collating elements (single character and multi-character collating elements). The sequence declarations define the collating rules.

The cultural environment is established by selecting the active locale. The active locale affects the behavior of locale-sensitive functions. In particular, the active locale's collating rules affect DFSORT's SORT, MERGE, INCLUDE, and OMIT processing as follows:

- Sort and Merge

  DFSORT produces sorted or merged records for output according to the collating rules defined in the active locale. This provides sorting and merging for single- or multi-byte character data based on defined collating rules that retain the cultural and local characteristics of a language.
- Include and Omit

  DFSORT includes or omits records for output according to the collating rules defined in the active locale. This provides inclusion or omission for single- or multi-byte character data based on defined collating rules that retain the cultural and local characteristics of a language.

The DFSORT option LOCALE specifies whether locale processing is to be used and, if so, designates the active locale. Only one locale can be active at a time for any DFSORT application.

DFSORT Processing

Unless you use DFSORT Panels to prepare and submit your job (see *Panels Guide*), you must prepare job control language (JCL) statements and DFSORT program control statements to invoke DFSORT processing. JCL statements (see "Chapter 5. Invoking DFSORT from a Program" on page 331) are processed by your operating system. They describe your data sets to the operating system and initiate DFSORT processing. DFSORT program control statements (see "Chapter 3. Using DFSORT Program Control Statements" on page 77) are processed by the DFSORT program. They describe the functions you want to perform and invoke the processing you request.

A sort application usually requires intermediate storage as working space during the program run. This storage can be one of the following:

1. Hiperspace, using DFSORT's Hipersorting feature.
2. Work data sets—either allocated dynamically by DFSORT's DYNALLOC facility or specified by the user, using JCL DD statements. If specified by the user, the

intermediate storage devices and the amount of work space must be indicated. Methods for determining the amount of work space to allocate are explained in “Appendix A. Using Work Space” on page 539.

3. A combination of Hiperspace and work data sets.

Merge and copy applications do not require intermediate storage.

Figure 2 on page 8 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 run your job.

DFSORT Processing

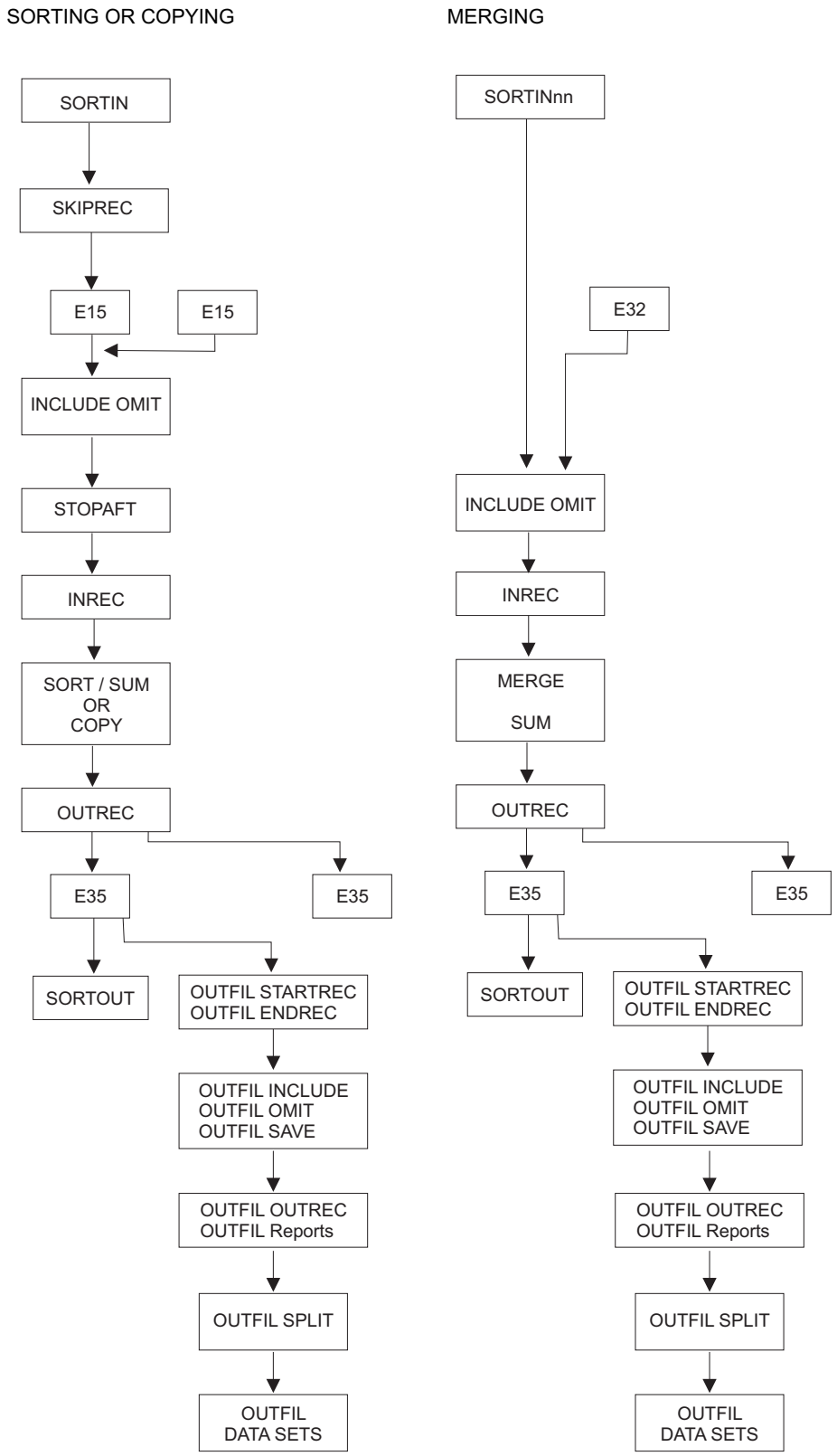


Figure 2. Record Processing Order

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

1. DFSORT first checks whether you supplied a SORTIN data set for SORT and COPY jobs or 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 user exit to insert all the records. (This is also true if you invoke DFSORT from a program with the address of an E15 user exit in the parameter list, because SORTIN will be ignored.) DFSORT can use a COBOL E15 routine if you specified the E15 user exit in the MODS statement.
   - If no SORTINnn data sets are present for a MERGE job, you must use an E32 user exit to insert all the records.
2. If input records for SORT or COPY jobs are 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 a SORT or COPY gives better performance.
3. If the input records for a SORT or COPY job are read from a SORTIN data set, DFSORT checks whether you specified an E15 user exit. If so, DFSORT transfers control to the user exit routine. You can use a COBOL E15 routine if the E15 user 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, 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 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 an INCLUDE or OMIT statement.
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 one.
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. 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 one.
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 one.

9. If an E35 user exit is present, DFSORT transfers control to your user exit routine after all statement processing is completed. If you changed record format, the E35 user exit receives the records in the current format rather than in the original one. You can use a COBOL E35 routine if you specify the E35 user exit in the MODS statement. You can use the E35 exit routine to add, delete, or reformat records.

   If SORTOUT and OUTFIL data sets are not present, the E35 exit must dispose of all the records because DFSORT treats these records as deleted. (This is also true if you do not specify OUTFIL data sets and DFSORT is invoked with the address of an E35 user exit in the parameter list, because SORTOUT will be ignored.)
10. DFSORT writes your records to the SORTOUT data set, if present.
11. DFSORT performs processing specified in one or more OUTFIL statements, if present:
    - DFSORT performs processing specified with the STARTREC and/or ENDREC options. Record input for the OUTFIL data sets starts with the record indicated by STARTREC and ends with the record indicated by ENDREC.
    - DFSORT performs processing specified with the INCLUDE, OMIT, or SAVE option. Records are included or omitted from the OUTFIL data sets according to the criteria specified.
    - DFSORT performs processing specified with the OUTREC, LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, and NODETAIL options. Data records are reformatted and report records are generated for the OUTFIL data sets.
    - DFSORT performs SPLIT processing. Records are distributed among the OUTFIL data sets as evenly as possible.
    - DFSORT writes your OUTFIL records to the appropriate OUTFIL data sets.

Input Data Sets—SORTIN and SORTINnn

DFSORT processes two types of input data sets, referred to as the SORTIN data set (or just SORTIN) and the SORTINnn data sets (or just SORTINnn).

The SORTIN DD statement specifies the input data set (or concatenated input data sets) for a sort or copy application. If a SORTIN DD statement is present, it will be used by default for a sort or copy application unless you invoke DFSORT from a program with the address of an E15 user exit in the parameter list.

The SORTINnn DD statements (where nn can be 00 to 99) specify the data sets for a merge application. If a SORTINnn DD statement is present, it will be used by default for a merge application unless you invoke DFSORT from a program with the address of an E32 user exit in the parameter list.

“Data Set Considerations” on page 11 contains general information about input data sets. For specific information about the SORTIN data set, see “SORTIN DD Statement” on page 66. For specific information about the SORTINnn data sets, see “SORTINnn DD Statement” on page 67.

Output Data Sets—SORTOUT and OUTFIL

DFSORT processes two types of output data sets, referred to as the SORTOUT data set (or just SORTOUT) and the OUTFIL data sets.

Output Data Sets—SORTOUT and OUTFIL

The SORTOUT DD statement specifies the single non-OUTFIL output data set for a sort, copy, or merge application. OUTFIL processing does not apply to SORTOUT. If a SORTOUT DD statement is present, it will be used by default for a sort, copy, or merge application unless you invoke DFSORT from a program with the address of an E35 user exit in the parameter list.

The FNAMES and/or FILES parameters of one or more OUTFIL statements specify the ddnames of the OUTFIL data sets for a sort, copy, or merge application. The parameters specified for each OUTFIL statement define the OUTFIL processing to be performed for the OUTFIL data sets associated with that statement. Each ddname specified must have a corresponding DD statement.

Although the ddname SORTOUT can actually be used for an OUTFIL data set, the term “SORTOUT” will be used to denote the single non-OUTFIL output data set.

“Data Set Considerations” contains general information about output data sets. For specific information about the SORTOUT data set, see “SORTOUT and OUTFIL DD Statements” on page 71. For specific information about the OUTFIL data sets, see “SORTOUT and OUTFIL DD Statements” on page 71 and “OUTFIL Control Statements” on page 182.

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 processing. See *Application Development Guide* 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 DD statements. You must place the DD statements in the operating system input stream with the job step that allocates DFSORT processing.

DFSORT Panels operates in two modes: foreground and background. Foreground mode uses CLIST processing instead of JCL, so if you choose this technique you do not need JCL at all. 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. See *Panels Guide* for more information.

Sorting or Copying Records

Input to a sort or copy application can be a blocked or unblocked QSAM or VSAM data set containing fixed- or variable-length records. QSAM input data sets can be concatenated even if they are on dissimilar devices. See “SORTIN DD Statement” on page 66 for the restrictions that apply.

Output from a sort or copy application can be blocked or unblocked QSAM or VSAM data sets, regardless of whether the input is QSAM or VSAM. Unless OUTFIL is used to convert variable input to fixed output, an output data set must be the same type (fixed or variable) as the input data set.

Hierarchical File System (HFS) files are supported as input and output for sort and copy applications.

SmartBatch pipes are supported as input and output for sort and copy applications.

Merging Records

Input to a merge application can be up to 100 blocked or unblocked QSAM or VSAM data sets containing fixed- or variable-length records. The 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 application can be blocked or unblocked QSAM or VSAM data sets, regardless of whether the input is QSAM or VSAM. Unless OUTFIL is used to convert variable input to fixed output, an output data set must be the same type (fixed or variable) as the input data sets.

Hierarchical File System (HFS) files are supported as input and output for merge applications.

SmartBatch pipes are supported as input and output for merge applications.

Data Set Notes and Limitations

There are some considerations and limitations that you need to be aware of. These are described in the following sections.

For more information about specific DFSORT data sets, see “Using DD Statements” on page 59.

General Considerations

Variable-length records are processed with a record descriptor word (RDW) in positions 1-4, so the data starts in position 5. Fixed-length records are processed without an RDW, so the data starts in position 1. Control statement positions should be specified accordingly.

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, or you can use locale processing with the appropriate locale.

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

Standard system data management rules apply to all data set processing. In particular, be aware that when using fixed standard record format for input data sets, the first short block is treated like an End of Volume. See *Using Data Sets* for more details.

The maximum record length DFSORT can handle is subject to the following limitations:

- Record length can never exceed the maximum record length you specify.
- Variable-length records are limited to 32756 bytes.
- VSAM variable-length records are limited to 32752 bytes.
- Fixed-length records are limited to 32760 bytes.
- Variable block-spanned records are limited to 32767 bytes.
- For a tape work data set sort, the maximum record length is limited to 32752 bytes with NOEQUALS in effect and to 32748 bytes with EQUALS in effect.

Note: If AQ format is specified, or CH format is specified and the CHALT option is in effect, the maximum record length for variable-length records is 32767 bytes, less the length of the control fields.

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 routine can limit the record length that can be used when processing variable-length records.

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

Padding and Truncation

You can control the action that DFSORT takes when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL with the TRUNC option as described in “OPTION Control Statement” on page 134.

DFSORT truncates fixed-length records on the right when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL provided that:
- The application is not a conventional merge or tape work data set sort.
- TRUNC=RC16 is not in effect.

You can control the action that DFSORT takes when a variable-length output record is longer than the LRECL of the SORTOUT or OUTFIL data set to which it is to be written by using the VLLONG or NOVLLONG option as described in “OPTION Control Statement” on page 134.

You can control the action that DFSORT takes when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL with the PAD option as described in “OPTION Control Statement” on page 134.

DFSORT pads fixed-length records with binary zeros on the right when the SORTOUT LRECL is larger than the SORTIN LRECL provided that:
- The Blockset technique is selected.
- The application is a sort or copy.
- PAD=RC16 is not in effect.

DFSORT does not pad or truncate records returned from an E15 or E35 user exit since it expects the exit to pad or truncate each record appropriately.

You can use INREC, OUTREC, and OUTFIL to pad, truncate, and reformat records. See “INREC Control Statement” on page 112 and “OUTREC Control Statement” on page 251 for details.

See “Use ICEGENER Instead of IEBGENER” on page 504 for information about padding and truncating with ICEGENER.

For more information about Blockset and other DFSORT techniques, see “Specify Efficient Sort/Merge Techniques” on page 491.

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.
- Empty input data sets can be used.
- If any of the input data sets are on tape without standard labels, DCB parameters must be specified on their DD statements.
- ISO/ANSI Version 1 tape files can only be used as input—never as output.
- DFSORT sets appropriate BUFNO values for the input and output data sets; specifying BUFNO in the DD statements for these data sets has no effect.

See “SORTIN DD Statement” on page 66 for additional considerations.

VSAM Considerations

- You can have DFSORT process VSAM records as fixed-length (F) or variable-length (V). When you use VSAM input, DFSORT selects fixed-length processing if you specify RECORD TYPE=F or variable-length processing if you specify RECORD TYPE=V. If you do not specify RECORD TYPE=x, DFSORT selects the record type to use according to the ″rules″ described in the discussion of the TYPE operand in “RECORD Control Statement” on page 260. The record type selected affects how the records are treated, and how control statement positions should be specified, as follows:
  - **Variable-length processing:** An RRDS, KSDS, ESDS or VRRDS can always be processed as variable-length. For VSAM input, DFSORT reads each record and prepends a record descriptor word (RDW) to it. For VSAM output, DFSORT removes the RDW before writing each record. Since DFSORT uses an RDW in positions 1-4 to process variable-length records, the data starts in position 5. Control statement positions should be specified accordingly.
  - **Fixed-length processing:** An RRDS can always be processed as fixed-length. A KSDS, ESDS or VRRDS used for input should only be processed as fixed-length if all of its records have a length equal to the maximum record size defined for the cluster. Otherwise, input records which are shorter than the maximum record size are padded with bytes that may or may not be zeros (that is, ″garbage″ bytes). DFSORT does not use an RDW to process fixed-length records, so the data starts in position 1. Control statement positions should be specified accordingly.
- If a data set is password protected, passwords can be entered at the console or (with some restrictions) through routines at user exits E18, E38, and E39.

  Note: Passwords cannot be handled in this way for OUTFIL data sets.
- If VSAMIO and RESET are in effect, a data set defined with REUSE can be used for both input and output for a sort; that is, the data set can be sorted in-place.
- A data set used for input or output must have been previously defined.
- If VSAMEMT is in effect, an empty input data set is processed as having zero records.
- VSAM data sets must not be concatenated (system restriction).
- VSAM and non-VSAM input data sets must not be specified together for a sort, merge or copy application.
- If output is a VSAM key-sequenced 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 can be used except EODAD. See the description of E18 use with VSAM in “Chapter 4. Using Your Own User Exit Routines” on page 279.

- You must build the VSAM exit list with the VSAM EXLST macro instruction giving the addresses of your routines that handle VSAM exit functions.
- When processing variable-length records with VSAM input and non-VSAM output, the output LRECL must be at least 4 bytes greater than the maximum record size defined for the cluster. Non-VSAM 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 for the VSAM cluster is therefore 4 bytes less than the non-VSAM LRECL.
- An output data set defined without REUSE is processed as MOD.
- If RESET is in effect, an output data set defined with REUSE is processed as NEW. If NORESET is in effect, an output data set defined with REUSE is processed as MOD.
- DFSORT cannot access VSAM data sets in RLS mode, that is, RLS=CR and RLS=NRI are not supported for VSAM input and output data sets.

SmartBatch Pipe Considerations

SmartBatch pipe data sets can be used for input and output, but are only supported by the Blockset technique. If Blockset is not selected for a DFSORT application that uses SmartBatch pipe data sets, DFSORT will issue an error message and terminate.

If DFSORT determines that a SmartBatch pipe data set is being used for input or output:

- it automatically forces the ABEND option on, to ensure that an ABEND will be generated if an error is detected, and
- it terminates with user ABEND zero instead of return code 16 if an E15, E32, or E35 user exit requests termination.

If ICETOOL determines that a SmartBatch pipe data set is being used for input or output, it automatically terminates with user ABEND 2222 instead of return code 12.

Generation of an ABEND in these situations allows for appropriate error propagation by the system to other applications that may be accessing the same SmartBatch pipe data set as DFSORT or ICETOOL.

If DFSORT or ICETOOL detects an error before determining that a SmartBatch pipe data set is being used or before opening the SmartBatch pipe data set, appropriate error propagation may not occur. This can cause another application to go into a permanent wait for a SmartBatch pipe data set.

HFS File Considerations

Hierarchical File System (HFS) files can be used for input and output, but are only supported by the Blockset technique. If Blockset is not selected for a DFSORT application that uses HFS files, DFSORT will issue an error message and terminate.

You should be familiar with the information found in *OS/390 UNIX System Services User's Guide* regarding HFS files if you use them. DFSORT uses BSAM to access HFS files and is thus subject to all of the capabilities and restrictions that entails, as described in *OS/390 DFSMS: Using Data Sets*.

Installation Defaults

When your system programmers installed DFSORT, they selected separate sets of installation (ICEMAC) parameters to be used by default for the following eight installation modules:

ICEAM1 (JCL)
is the batch direct invocation environment installation module. This set of defaults is used at run time when DFSORT is invoked directly (that is, not through programs) by batch jobs, provided that an enabled time-of-day installation module (ICETDx) is not activated.

ICEAM2 (INV)
is the batch program invocation environment installation module. This set of defaults is used at run time when DFSORT is invoked through batch programs, provided that an enabled time-of-day installation module (ICETDx) is not activated.

ICEAM3 (TSO)
is the TSO direct invocation environment installation module. This set of defaults is used at run time when DFSORT is invoked directly (that is, not through programs) by foreground TSO users, provided that an enabled time-of-day installation module (ICETDx) is not activated.

ICEAM4 (TSOINV)
is the TSO program invocation environment installation module. This set of defaults is used at run time when DFSORT is invoked through programs by foreground TSO users, provided that an enabled time-of-day installation module (ICETDx) is not activated.

ICETD1 (TD1)
is the first time-of-day installation module. This set of defaults is used at run time when it is activated for the time-of-day of the run, provided it is enabled by the environment installation module (ICEAMx) in effect.

ICETD2 (TD2)
is the second time-of-day installation module. This set of defaults is used at run time when it is activated for the time-of-day of the run, provided it is enabled by the environment installation module (ICEAMx) in effect.

ICETD3 (TD3)
is the third time-of-day installation module. This set of defaults is used at run time when it is activated for the time-of-day of the run, provided it is enabled by the environment installation module (ICEAMx) in effect.

ICETD4 (TD4)
is the fourth time-of-day installation module. This set of defaults is used at run time when it is activated for the time-of-day of the run, provided it is enabled by the environment installation module (ICEAMx) in effect.

The selected defaults can affect the way your applications run, and in many cases can be overridden by specifying the appropriate run-time parameters (see “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details). This book assumes that DFSORT was installed at your site with the defaults that it was delivered with.

You can use an ICETOOL job similar to the following one to list the installation defaults actually in use at your site for the eight installation modules and the IBM-supplied defaults they override, where appropriate.

Table 1. Using ICETOOL to List Installation Defaults

```
//DFRUN JOB A402,PROGRAMMER
//LISTDEF EXEC PGM=ICETOOL,REGION=1024K
//TOOLMSG DD SYSOUT=A
//DFSMSG DD SYSOUT=A
//SHOWDEF DD SYSOUT=A
//TOOLIN DD *
DEFAULTS LIST(SHOWDEF)
/*
```

See “Chapter 6. Using ICETOOL” on page 349 and “DEFAULTS Operator” on page 365 for more information on using ICETOOL and the DEFAULTS operator.

The functions of the available ICEMAC parameters are summarized below. *Installation and Customization* contains complete descriptions of the available ICEMAC parameters, as well as planning considerations and general information about installing DFSORT. Step-by-step installation procedures are listed in the *DFSORT Program Directory*.

Parameter	Function
INV|JCL|TSO|TSOINV|TD1|TD2|TD3|TD4	Specifies the environment installation module (ICEAMx) or time-of-day installation module (ICETDx) for which this set of ICEMAC defaults is to be used.
ENABLE	Specifies whether ICETDx installation modules are to be used if activated for this ICEAMx environment module.
day	Specifies the time ranges for each day of the week when this ICETDx installation module is to be activated.
ABCODE	Specifies the ABEND code used when DFSORT abends for a critical error.
ALTSEQ	Alters the normal EBCDIC collating sequence.
ARESALL	Specifies the number of bytes reserved above 16MB virtual for system use.
ARESINV	Specifies the number of bytes reserved above 16MB virtual for the invoking program when DFSORT is program invoked.
CFW	Specifies whether DFSORT can use cache fast write when processing work data sets.
CHALT	Translates format CH as well as format AQ, or translates format AQ only.
CHECK	Specifies whether record count checking is suppressed for applications that use an E35 user exit routine without an output data set.
CINV	Specifies whether DFSORT can use control interval access for VSAM data sets.
COBEXIT	Specifies the library for COBOL E15 and E35 routines.

Installation Defaults

DIAGSIM	Specifies whether a SORTDIAG DD statement is to be simulated for DFSORT applications.
DSA	Specifies the maximum amount of storage available to DFSORT for dynamic storage adjustment of Blockset sort applications.
DSPSIZE	Specifies the maximum amount of data space to use for dataspace sorting.
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 and run-time option DYNALLOC.
DYNAUTO	Specifies whether work data sets are dynamically allocated automatically.
DYNSPC	Specifies the default primary space allocation for dynamically allocated work data sets.
EFS	Specifies the name of a user-written Extended Function Support program to be called by DFSORT.
EQUALS	Specifies whether the order of records that collate identically is preserved from input to output.
ERET	Specifies the action taken if DFSORT encounters a critical error.
ESTAE	Specifies whether DFSORT deletes its ESTAE recovery routine early or uses it for the entire run.
EXITCK	Specifies whether DFSORT terminates or continues when it receives certain invalid return codes from E15 or E35 user exit routines.
EXPMAX	Specifies the maximum total amount of available expanded storage to be used at any one time by all Hipersorting applications.
EXPOLD	Specifies the maximum total amount of old expanded storage to be used at any one time by all Hipersorting applications.
EXPRES	Specifies the minimum amount of available expanded storage to be reserved for use by non-Hipersorting applications.
FSZEST	Specifies whether DFSORT treats run-time options FILSZ=n and SIZE=n as exact or estimated file sizes.
GENER	Specifies the name that ICEGENER is to use to transfer control to the IEBGENER system utility. (ICEGENER is DFSORT's facility for IEBGENER jobs.)
GNPAD	Specifies the action to be taken by ICEGENER for LRECL padding.
GNTRUNC	Specifies the action to be taken by ICEGENER for LRECL truncation.

HIPRMAX	Specifies the maximum amount of Hiperspace to use for Hipersorting.
IDRCPCT	Specifies a percentage which represents the approximate amount of data compaction achieved by using the Improved Data Recording Capability feature of IBM tape devices that support compaction.
IEXIT	Specifies whether DFSORT passes control to your site's ICEIEXIT routine.
IGNCKPT	Specifies whether the checkpoint/restart facility is ignored if it is requested at run-time and the Blockset technique (which does not support the checkpoint/restart facility) can be used.
IOMAXBF	Specifies an upper limit to the amount of buffer space to be used for SORTIN, SORTINnn and SORTOUT data sets.
LIST	Specifies whether DFSORT prints control statements.
LISTX	Specifies whether DFSORT prints control statements returned by an Extended Function Support program.
LOCALE	Specifies whether locale processing is to be used and, if so, designates the active locale.
MAXLIM	Specifies an upper limit to the amount of main storage available to DFSORT below 16MB virtual.
MINLIM	Specifies a lower limit to the amount of main storage available to DFSORT.
MSGCON	Specifies the class of program messages DFSORT writes to the master console.
MSGDDN	Specifies an alternate name for the message data set.
MSGPRT	Specifies the class of program messages DFSORT writes to the message data set.
NOMSGDD	Specifies whether DFSORT terminates or continues when the message data set is required but is not available.
ODMAXBF	Specifies an upper limit to the amount of buffer space to be used for each OUTFIL data set.
OUTREL	Specifies whether unused temporary output data set space is released.
OUTSEC	Specifies whether DFSORT uses automatic secondary allocation for output data sets that are temporary or new.
OVERRGN	Specifies the amount of main storage above the REGION value available to Blockset.
OVFLO	Specifies the action to be taken by DFSORT when BI, FI, PD or ZD summary fields overflow.

PAD	Specifies the action to be taken by DFSORT for LRECL padding.
PARMDDN	Specifies an alternate ddname for the DFSORT DFSPARM data set.
RESALL	Reserves storage for system and application use when SIZE/MAINSIZE=MAX is in effect.
RESET	Specifies whether DFSORT processes a VSAM output data set defined with REUSE as a NEW or MOD data set.
RESINV	Reserves storage for programs invoking DFSORT when SIZE/MAINSIZE=MAX is in effect.
SDB	Specifies whether DFSORT should use the system-determined optimum block size for output data sets when the block size is zero.
SDBMSG	Specifies whether DFSORT and ICETOOL should use the system-determined optimum block size for message and list data sets when the block size is zero.
SIZE	Specifies the maximum amount of main storage available to DFSORT.
SMF	Specifies whether DFSORT produces SMF type-16 records.
SOLRF	Specifies whether DFSORT uses the reformatted record length for the SORTOUT LRECL.
SORTLIB	Specifies whether DFSORT searches a system or private library for the modules used with a tape work data set sort or Conventional merge.
SPANINC	Specifies the action to be taken by DFSORT when incomplete spanned records are detected.
STIMER	Specifies whether DFSORT uses the STIMER macro. If DFSORT does not use the STIMER macro, processor timing data does 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.
SZERO	Specifies whether DFSORT treats numeric −0 and +0 values as signed (that is, different) or unsigned (that is, the same).
TEXIT	Specifies whether DFSORT passes control to your site's ICETEXIT routine.
TMAXLIM	Specifies an upper limit to the total amount of main storage above and below 16MB virtual available to DFSORT when SIZE/MAINSIZE=MAX is in effect.
TRUNC	Specifies the action to be taken by DFSORT for LRECL truncation.

VERIFY Specifies whether the sequence of output records is verified.

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

VLLONG Specifies whether DFSORT truncates long variable-length output fields.

VLSCMP Specifies whether DFSORT pads short variable-length compare fields.

VLSHRT Specifies whether DFSORT continues processing if a short variable-length control field or compare field is found.

VSAMBSP Specifies the number of VSAM buffers DFSORT can use.

VSAMEMT Specifies whether DFSORT accepts an empty VSAM input data set.

VSAMIO Specifies whether DFSORT allows a VSAM data set defined with REUSE to be sorted in-place.

WRKREL Specifies whether unused temporary work data set space is released.

WRKSEC Specifies whether DFSORT uses automatic secondary allocation for temporary work data sets.

Y2PAST Specifies the sliding or fixed century window.

ZDPRINT Specifies whether DFSORT produces printable numbers from positive ZD fields that result from summarization.

Tables showing all the possible sources of specification and order of override for each option are shown in “Appendix B. Specification/Override of DFSORT Options” on page 549.

Migrating to DFSORT from Other Sort Products

If you are migrating to DFSORT, you should review the IBM-supplied ICEMAC defaults and change them as appropriate to correspond to equivalent settings for your previous sort product. In particular, the options shown in the table that follows can make DFSORT operate more like other sort products, thus making migration easier. The ICEMAC options, described in *DFSORT Installation and Customization*, can be used to change the way DFSORT works globally by default. The run-time options, described in “Chapter 2. Invoking DFSORT with Job Control Language” on page 25 and “Chapter 3. Using DFSORT Program Control Statements” on page 77, can be used to override the ICEMAC options for specific jobs.

Table 2. Options That Can Ease Migration

ICEMAC	Run-Time
ABCODE=MSG|n	
DYNALOC=(d,n)	DYNALLOC=(d,n)
DYNAUTO=YES|IGNWKDD|NO	DYNALLOC=(d,n)
EQUALS=YES/NO/VBLKSET	EQUALS|NOEQUALS

Table 2. Options That Can Ease Migration (continued)

ICEMAC	Run-Time
EXITCK=STRONG|WEAK	EXITCK=STRONG|WEAK
FSZEST=YES|NO	FILSZ=n|En|Un
NOMSGDD=QUIT|ALL|CRITICAL|NONE	
PARMDDN=ddname	
RESET=YES|NO	RESET|NORESET
SORTLIB=SYSTEM|PRIVATE	
SZERO=YES|NO	SZERO|NOSZERO
VLLONG=YES|NO	VLLONG|NOVLLONG
VLSCMP=YES|NO	VLSCMP|NOVLSCMP
VSAMEMT=YES|NO	VSAMEMT|NVSAMEMT
VSAMIO=YES|NO	VSAMIO|NOVSAMIO
ZDPRINT=YES/NO	ZDPRINT|NZDPRINT

DFSORT Messages and Return Codes

You can determine, during installation or run-time, whether DFSORT writes messages to the message data set, to the master console, or to 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 run-time.

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

See *Messages, Codes and Diagnosis* for complete information about DFSORT messages.

For successful completion, DFSORT passes back a return code of 0 or 4 to the operating system or the invoking program.

For unsuccessful completion due to an unsupported operating system, DFSORT passes back a return code of 24 to the operating system or the invoking program.

For unsuccessful completion with NOABEND in effect, DFSORT passes back a return code of 16 or 20 to the operating system or the invoking program.

For unsuccessful completion with ABEND in effect, DFSORT issues a user abend with the appropriate code as specified by ICEMAC option ABCODE (either the error message number or a number between 1 and 99).

The meanings of the return codes that DFSORT passes back (in register 15) are:

0 **Successful completion**. DFSORT completed successfully.

4 **Successful completion**. DFSORT completed successfully, and:

- OVFLO=RC4 was in effect and summary fields overflowed, or

- PAD=RC4 was in effect and the SORTOUT LRECL was larger than the SORTIN/SORTINnn LRECL (LRECL padding), or
- TRUNC=RC4 was in effect and the SORTOUT LRECL was smaller than the SORTIN/SORTINnn LRECL (LRECL truncation), or
- SPANINC=RC4 was in effect and one or more incomplete spanned records was detected.

16 **Unsuccessful completion**. DFSORT detected an error that prevented it from completing successfully.

20 **Message data set missing**. ICEMAC option NOMSGDD=QUIT was in effect and neither a message data set DD statement nor a SYSOUT DD statement was provided.

24 **Unsupported operating system**. This operating system is not supported by this release of DFSORT. Only current or subsequent releases of the following systems are supported:

- OS/390
- MVS/ESA

Use Blockset Whenever Possible

Blockset is DFSORT's most efficient technique. It supports many features not supported by DFSORT's less efficient Peerage/Vale and Conventional techniques (see ICE189A in *Messages, Codes and Diagnosis* for a list of these features). DFSORT always selects Blockset for a copy application. DFSORT selects Blockset for a sort or merge application unless something prevents it from doing so (see ICE800I in *Messages, Codes and Diagnosis* for a list of reasons Blockset cannot be used).

Note: Blockset cannot be used to process BDAM data sets.

Message ICE143I indicates whether Blockset or a less efficient technique was selected for a particular run. If Blockset was not selected for a sort or merge application, check the reason code in message ICE800I, which indicates the reason Blockset could not be used. If you did not get message ICE800I, add the following DD statements to your application and rerun it:

```
//SORTDIAG DD DUMMY
//SYSOUT DD SYSOUT=*
```

If possible and appropriate, remove the obstacle that is causing Blockset not to be selected.

Chapter 2. Invoking DFSORT with Job Control Language

Using the JCL 25
Using the JOB Statement 27
Using the EXEC Statement 27
Specifying EXEC Statement Cataloged Procedures 28
SORT Cataloged Procedure 28
SORTD Cataloged Procedure 29
Specifying EXEC/DFSPARM PARM Options 29
Aliases for PARM Options 58
Using DD Statements 59
Duplicate Ddnames 61
Shared Tape Units 61
System DD Statements 62
Program DD Statements 64
SORTLIB DD Statement 65
SYMNAMES DD and SYMNOUT DD Statements 66
SORTIN DD Statement 66
SORTINnn DD Statement 67
SORTWKdd DD Statement 69
SORTOUT and OUTFIL DD Statements. 71
SORTCKPT DD Statement 73
SORTCNTL DD Statement 73
DFSPARM DD Statement 74
SORTDKdd DD Statement 76
SORTDIAG DD Statement 76
SORTSNAP DD Statement 76

Using the JCL

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 run your job
- Run your job
- Return information to you about the results
- Terminate your job.

Unless you create your jobs with the interactive DFSORT Panels facility (see *Panels Guide*), you must supply JCL 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:

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

© Copyright IBM Corp. 1973, 2000

Using the JCL

DFSORT Panels offers an alternative to coding JCL directly. When you use panels to prepare a job to be run 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 *Panels Guide* 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
//SYMNAMES DD	Defines the SYMNAMES data set containing statements to be used for symbol processing
//SYMNOUT DD	Defines the data set in which SYMNAMES statements and the symbol table are to be listed
//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 SORTOUT output data set for a sort, merge, or copy
//outfil DD	Defines an OUTFIL output data set for a sort, merge, or copy
//SORTWKdd 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
//SYSMDUMP DD	Same as //SYSUDUMP DD
//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 user exit routines from SYSIN.

[1] These are the default ddnames with which DFSORT was delivered. SYSOUT and DFSPARM may have been changed during DFSORT installation. You can change all of the indicated ddnames at run time. For override information, see “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 may have made information such as account number and the name of the programmer mandatory:

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

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──┘ │ └─,PARM='─options─┴─────────────────┴─'─┘
                    └─┬─SORT───┬─────┘
                      └─SORTD──┘

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

If you use a cataloged procedure (discussed in detail below), specify PROC=SORT or PROC=SORTD. You can omit PROC= and simply specify 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 one of its aliases: SORT, IERRCO00, or IGHRCO00. 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 DFSORT program modules. If DFSORT was installed as part of the normal system link libraries, the STEPLIB DD statement is unnecessary. It is needed only if DFSORT resides in a separate link library which is not part of the “link list.” (Your installation’s system programmers can give you this information.) The STEPLIB DD statement shown assumes that the data set name represented by yyy is cataloged.
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 installation time; it can be SYS1.SORTLIB. If the modules were installed in a system library and ICEMAC SORTLIB=SYSTEM is used, the SORTLIB DD statement is unnecessary and is ignored unless dynamic link of user exits is used.
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

primary 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 when you include user routines that do not require link-editing.

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

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

Line **Explanation**

00 The stepname of the SORTD procedure is SORT

10 The STEPLIB DD statement defines the data set containing the DFSORT program modules. If DFSORT was installed as part of the normal system link libraries, the STEPLIB DD statement is unnecessary. It is needed only if DFSORT resides in a separate link library which is not part of the “link list.” (Your installation’s system programmers can give you this information.) The STEPLIB DD statement shown assumes that the data set name represented by yyy is cataloged.

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 installation 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 unless dynamic link edit of user exits is used.

30 Directs messages to system output class A

Specifying EXEC/DFSPARM PARM Options

When you invoke DFSORT with JCL, you can specify some DFSORT options on the PARM parameter of the EXEC statement as illustrated on the following page. 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 B. Specification/Override of DFSORT Options” on page 549.

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 74.

Details of aliases for PARM options are given under the description of individual options. “Aliases for PARM Options” on page 58 summarizes the available aliases.

DFSORT accepts but does not process the following EXEC/DFSPARM PARM options: BALANCE, BALN, BIAS=value, BMSG, CASCADE, CMP=value, CPU, CRCX, DEBUG, DIAG, ELAP, EXCPVR=value, IO, INCOR=value, INCORE=value, LRGSORT, L6=value, L7=value, NOCOMMAREA, NOINC, NOIOERR, OPT=value, OSCL, PEER, POLY, and PRINT121.

```
                  ┌─,──────────────────────────────────────┐
►►─,PARM='─────────┴─┬─ABEND──┬──────────────────────────┴──────────────────────────►◄
                     │ NOABEND │
                     ├─ARESALL=──┬─n──┬──────────────────────────────────────────────
                     │           ├─nK─┤
                     │           └─nM─┘
                     ├─AVGRLEN=n──────────────────────────────────────────────────────
                     ├─BSAM───────────────────────────────────────────────────────────
                     ├─┬─CINV───┬─────────────────────────────────────────────────────
                     │ └─NOCINV─┘
                     ├─COBEXIT=──┬─COB1─┬──────────────────────────────────────────────
                     │           └─COB2─┘
                     ├─DSPSIZE=──┬─MAX──┬──────────────────────────────────────────────
                     │           └─n────┘
                     ├─DYNALLOC──┬──────────────────────────────────────────────────────
                     │           └─=──┬─d──────────┬──────────────────────────────────
                     │                ├─(d)────────┤
                     │                ├─(,n)───────┤
                     │                ├─(d,n)──────┤
                     │                ├─OFF────────┤
                     │                └─(OFF)──────┘
                     ├─EFS=──┬─name─┬───────────────────────────────────────────────
                     │       └─NONE─┘
                     ├─┬─EQUALS───┬─────────────────────────────────────────────────
                     │ └─NOEQUALS─┘
                     ├─E15=COB────────────────────────────────────────────────────────
                     ├─E35=COB────────────────────────────────────────────────────────
                     ├─FILSZ=──┬─x──┬─────────────────────────────────────────────────
                     │         ├─Ex─┤
                     │         └─Ux─┘
                     ├─HIPRMAX=──┬─OPTIMAL─┬──────────────────────────────────────────
                     │           ├─n───────┤
                     │           └─p%──────┘
                     ├─┬─LIST───┬─────────────────────────────────────────────────────
                     │ └─NOLIST─┘
                     ├─┬─LISTX───┬────────────────────────────────────────────────────
                     │ └─NOLISTX─┘
                     ├─LOCALE=──┬─name────┬──────────────────────────────────────────
                     │          ├─CURRENT─┤
                     │          └─NONE────┘
                     ├─MSGDDN=ddname──────────────────────────────────────────────────
                     ├─MSGPRT=──┬─ALL────┬────────────────────────────────────────────
                     │          ├─CRITICAL─┤
                     │          └─NONE─────┘
                     ├─ODMAXBF=──┬─n──┬──────────────────────────────────────────────
                     │           ├─nK─┤
                     │           └─nM─┘
                     ├─┬─OUTREL───┬─────────────────────────────────────────────────
                     │ └─NOOUTREL─┘
                     └─OVFLO=──┬─RC0──┬───────────────────────────────────────────────
                               ├─RC4──┤
                               └─RC16─┘
```

Figure 3. Syntax Diagram for EXEC PARM (Part 1 of 3)

```
├──┬─PAD=──┬─RC0──┬──────────────────────────────────────────────────────┤
   │       ├─RC4──┤                                                      │
   │       └─RC16─┘                                                      │
   ├─RESALL=──┬─n──┬──────────────────────────────────────────────────────│
   │          ├─nK─┤                                                      │
   │          └─nM─┘                                                      │
   ├──┬─RESET────┬──────────────────────────────────────────────────────│
   │  └─NORESET──┘                                                      │
   ├─SDB=──┬─LARGE──┬──────────────────────────────────────────────────│
   │       ├─YES────┤                                                    │
   │       ├─INPUT──┤                                                    │
   │       └─NO─────┘                                                    │
   ├─SIZE=──┬─n──────┬──────────────────────────────────────────────────│
   │        ├─nK─────┤                                                    │
   │        ├─nM─────┤                                                    │
   │        ├─MAX────┤                                                    │
   │        ├─MAX-m──┤                                                    │
   │        ├─MAX-mK─┤                                                    │
   │        └─MAX-mM─┘                                                    │
   ├─SKIPREC=z──────────────────────────────────────────────────────────│
   ├──┬─SOLRF────┬──────────────────────────────────────────────────────│
   │  └─NOSOLRF──┘                                                      │
   ├─SPANINC=──┬─RC0──┬──────────────────────────────────────────────────│
   │           ├─RC4──┤                                                    │
   │           └─RC16─┘                                                    │
   ├──┬─STIMER───┬──────────────────────────────────────────────────────│
   │  └─NOSTIMER─┘                                                      │
   ├─STOPAFT=n──────────────────────────────────────────────────────────│
   ├──┬─SZERO────┬──────────────────────────────────────────────────────│
   │  └─NOSZERO──┘                                                      │
   ├─TRUNC=──┬─RC0──┬──────────────────────────────────────────────────│
   │         ├─RC4──┤                                                    │
   │         └─RC16─┘                                                    │
   ├──┬─VERIFY───┬──────────────────────────────────────────────────────│
   │  └─NOVERIFY─┘                                                      │
   ├──┬─VLLONG───┬──────────────────────────────────────────────────────│
   │  └─NOVLLONG─┘                                                      │
   ├──┬─VLSCMP───┬──────────────────────────────────────────────────────│
   │  └─NOVLSCMP─┘                                                      │
   ├──┬─VLSHRT───┬──────────────────────────────────────────────────────│
   │  └─NOVLSHRT─┘                                                      │
   ├──┬─VSAMEMT──┬──────────────────────────────────────────────────────│
   │  └─NOVSAMENT┘                                                      │
   ├──┬─VSAMIO───┬──────────────────────────────────────────────────────│
   │  └─NOVSAMIO─┘                                                      │
```

Figure 3. Syntax Diagram for EXEC PARM (Part 2 of 3)

```
├──┬─┬─WRKREL──┬──┬──────────────────────────────────────────────────────┤
   │ │ NOWRKREL┘  │
   │ ├─WRKSEC────┤
   │ │ NOWRKSEC┘  │
   │ ├─Y2PAST=─┬─s─┤
   │ │         │ f │
   │ │         └───┘
   │ └─ZDPRINT───┤
   │   NZDPRINT┘
```

Figure 3. Syntax Diagram for EXEC PARM (Part 3 of 3)

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, DFSORT 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 may 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.

Notes:

1. RC16=ABE and NORC16 can be used instead of ABEND and NOABEND, respectively.
2. If DFSORT determines that a SmartBatch pipe data set is being used, it automatically forces the ABEND option on, to ensure that an abend will be generated if an error is detected. This allows for appropriate error propagation by the system to other applications that may be accessing the same SmartBatch pipe data set.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See Appendix B. Specification/Override of DFSORT Options

ARESALL

```
►►──ARESALL=──┬─n──┬──────────────────────────────────────────────────►◄
              ├─nK─┤
              └─nM─┘
```

Temporarily overrides the ARESALL installation option, which specifies the number of bytes to be reserved above 16MB virtual for system use. For more information, see the discussion of the ARESALL option in “OPTION Control Statement” on page 134.

n specifies that n bytes of storage are to be reserved.

Limit: 8 digits.

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

Limit: 5 digits.

nM
specifies that n times 1048576 bytes of storage are to be reserved.

Limit: 2 digits.

Note: RESERVEX=value can be used instead of ARESALL=value.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

AVGRLEN

```
►►──AVGRLEN=n──────────────────────────────────────────────────────────►◄
```

Specifies the average input record length in bytes for variable-length record sort applications. For more information, see the discussion of the AVGRLEN option in “OPTION Control Statement” on page 134.

n specifies the average input record length. The value for n must be between 4 and 32767 and must include the 4 byte record descriptor word (RDW).

Note: L5=n can be used instead of AVGRLEN=n.

Default: If AVGRLEN=n is not specified, DFSORT will use one-half of the maximum record length as the average record length. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

BSAM

```
►►──BSAM───────────────────────────────────────────────────────────────►◄
```

Temporarily bypasses the EXCP access method normally used for input and output data sets. BSAM is ignored for VSAM input and output data sets. Note that if Blockset is not selected and BSAM processing is used with concatenated SORTIN input and both null and non-null data sets are specified, all null data sets must precede all non-null data sets; otherwise, the results are unpredictable.

Note: This option can degrade performance.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 option in “OPTION Control Statement” on page 134.

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 B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

COBEXIT

```
►►──COBEXIT=──┬─COB1──┬──────────────────────────────────────────────►◄
              └─COB2──┘
```

Temporarily overrides the COBEXIT installation option, which specifies the library for COBOL E15 and E35 routines.

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

COB2
specifies that COBOL E15 and E35 routines are run with either the VS COBOL II run-time library or the Language Environment run-time library.

Note: See “COBOL User Exit Requirements” on page 309 for additional information on the use of COBEXIT=COB2.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

DSPSIZE

```
►►──DSPSIZE=──┬─MAX─┬──────────────────────────────────────────────────►◄
              └─n───┘
```

Temporarily overrides the DSPSIZE installation option, which specifies the maximum amount of data space to be used for dataspace sorting. For more information, see the discussion of the DSPSIZE option in “OPTION Control Statement” on page 134.

MAX
specifies that DFSORT dynamically determines the maximum amount of data space that will be used for dataspace sorting. In this case, DFSORT bases its data space usage on the size of the file being sorted and the paging activity of the system.

n specifies the maximum amount, in megabytes, of data space to be used for dataspace sorting. n must be a value between 0 and 9999. The actual amount of data space used does not exceed n, but may be less depending on the size of the file being sorted and the paging activity of the system.

If n is zero, dataspace sorting is not used.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

DYNALLOC

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

Specifies that DFSORT dynamically allocates needed work space. You do not need to calculate and use JCL to specify the amount of work space needed by the program.

For more information, see the discussion of the DYNALLOC option in “OPTION Control Statement” on page 134 and “Appendix A. Using Work Space” on page 539

d specifies the device name. You can specify any IBM direct access storage device or tape device supported by your operating system in the same way you would specify it in the JCL UNIT parameter. You can also specify a group name, such as DISK or SYSDA.

For best performance, specify an emulated 3390-9 device (such as a RAMAC) or another high-speed IBM DASD device, and avoid specifying a tape, virtual (VIO) or real 3390-9 device.

n specifies the maximum number of requested work data sets. If you specify more than 255, a maximum of 255 data sets is used. If you specify 1 and the Blockset technique is selected, a maximum of 2 data sets is used. If you specify more than 32 and the Blockset technique is not selected, a maximum of 32 data sets is used.

Note: For optimum allocation of resources such as virtual storage, avoid specifying a large number of work data sets unnecessarily.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

DYNALLOC=OFF

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

Directs DFSORT *not* to allocate intermediate workspace dynamically. It overrides the ICEMAC installation option DYNAUTO=YES or the DYNALLOC parameter (without OFF) specified at run-time. For more information, see the discussion of the DYNALLOC option in “OPTION Control Statement” on page 134.

OFF
directs DFSORT not to allocate intermediate workspace dynamically.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

EFS

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

Temporarily overrides the EFS installation option, which specifies whether DFSORT passes control to an EFS program. See “Chapter 8. Using Extended Function Support” on page 455 for more information on EFS.

name
specifies 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.

Note: If you use locale processing for SORT, MERGE, INCLUDE, or OMITfields, you must not use an EFS program. DFSORT’s locale processing may eliminate the need for an EFS program. See “OPTION Control Statement” on page 134 for information related to locale processing.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

EQUALS or NOEQUALS

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

Temporarily overrides the EQUALS installation option, which specifies whether the original sequence of records that collate identically for a sort or a merge should be preserved from input to output. For more information, see the discussion of the EQUALS and NOEQUALS options in “OPTION Control Statement” on page 134.

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 B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

FILSZ

```
►►──FILSZ=──┬─x──┬──────────────────────────────────────────────────►◄
            ├─Ex─┤
            └─Ux─┘
```

Specifies either the exact number of records to be sorted or merged, or an estimate of the number of records to be sorted. This record count is used by DFSORT for two purposes:

1. To check that the actual number of records sorted or merged is equal to the exact number of records expected. FILSZ=x causes this check to be performed and results in termination with message ICE047A if the check fails.
2. To determine the input file size for a sort application. DFSORT performs calculations based on the user supplied record count and other parameters (such as AVGRLEN) to estimate the total number of bytes to be sorted. This value is important for sort applications, since it is used for several internal optimizations as well as for dynamic work data set allocation (see OPTION DYNALLOC). If no input record count (or only an estimate) is supplied for the sort application, DFSORT attempts to automatically compute the file size to be used for the optimizations and allocations.

The type of FILSZ value specified (x, Ex, Ux, or none) controls the way DFSORT performs the above two functions, and can have a significant effect on performance and work data set allocation. See “Specify Input/Output Data Set Characteristics Accurately” on page 492 and “Allocation of Work Data Sets” on page 541 for more information on file size considerations.

x specifies the exact number of records to be sorted or merged. This value is always used for both the record check and file size calculations. FILSZ=x can be used to force DFSORT to perform file size calculations based on x, and to cause DFSORT to terminate the sort or merge application if x is not exact.

If the FSZEST=NO installation option is in effect and FILSZ=x is specified, DFSORT terminates if the actual number of records is different from the specified value (x), the actual number of records placed in the IN field of message ICE047A (or message ICE054I) before termination. However, if the FSZEST=YES installation option is in effect, DFSORT treats FILSZ=x like FILSZ=Ex; it does not terminate when the actual number of records does not equal x.

The specified value (x) must take into account the number of records in the input data sets, records to be inserted or deleted by exit E15 or E32, and records to be deleted by the INCLUDE/OMIT statement, SKIPREC, and STOPAFT. x must be changed whenever the number of records to be sorted or merged changes in any way.

FILSZ=0 causes Hipersorting, dataspace sorting, and dynamic allocation of work space not to be used, and results in termination with the message ICE047A unless the number of records sorted or merged is 0.

Limit: 28 digits (15 significant digits)

Ex

specifies an estimated number of records to be sorted. This value is not used for the record check. It is used for file size calculations, but only if DFSORT could not automatically compute the file size. In all other cases,

this value is ignored by DFSORT. See “Dynamic Allocation of Work Data Sets” on page 542 for details on exactly when FILSZ=Ex is used or ignored by DFSORT.

The specified value (x) should take into account the number of records in the input data sets, records to be inserted or deleted by exit E15, and records to be deleted by the INCLUDE/OMIT statement, SKIPREC, and STOPAFT. x should be changed whenever the number of records to be sorted changes significantly.

FILSZ=E0 will always be ignored.

Limit: 28 digits (15 significant digits)

Ux

specifies the number of records to be sorted. This value is not used for the record check, but is always used for file size calculations. FILSZ=Ux can be used to force DFSORT to perform file size calculations based on x, while avoiding termination if x is not exact.

The FSZEST installation option has no effect on FILSZ=Ux processing.

The specified value (x) should take into account the number of records in the input data sets, records to be inserted or deleted by exit E15, and records to be deleted by the INCLUDE/OMIT statement, SKIPREC, and STOPAFT. x should be changed whenever the number of records to be sorted changes significantly.

FILSZ=U0 causes Hipersorting, dataspace sorting,and dynamic allocation of work space not to be used, and can cause degraded performance or termination with the message ICE046A, if the actual number of records to be sorted is significantly larger than 0.

Limit: 28 digits (15 significant digits)

Table 3 summarizes the differences for the three FILSZ variations:

Table 3. FILSZ Variations Summary.

FILSZ=n is equivalent to FILSZ=En if installation option FSZEST=YES is specified.

	FILSZ=n	**FILSZ=Un**	**FILSZ=En**
Number of records	Exact	Estimate	Estimate
Applications	Sort, merge	Sort	Sort
Terminate if wrong?	Yes	No	No
Use for file size calculation?	Yes	Yes	When DFSORT cannot compute file size
n includes records:			
In input data set(s)	Yes	Yes	Yes
Inserted/deleted by E15	Yes	Yes	Yes
Inserted by E32	Yes	No	No
Deleted by INCLUDE/OMIT statement	Yes	Yes	Yes

Table 3. FILSZ Variations Summary (continued).

FILSZ=n is equivalent to FILSZ=En if installation option FSZEST=YES is specified.

	FILSZ=n	FILSZ=Un	FILSZ=En
Deleted by SKIPREC	Yes	Yes	Yes
Deleted by STOPAFT	Yes	Yes	Yes
Update n when number of records changes:	In any way	Significantly	Significantly
Effects of n=0	Hipersorting and DYNALLOC not used	Hipersorting and DYNALLOC not used	None

Note: Using the FILSZ parameter to supply inaccurate information to DFSORT can negatively affect DFSORT's performance, and when work space is dynamically allocated, can result in wasted DASD space or termination with message ICE083A or ICE046A. Therefore, it is important to update the record count value whenever the number of records to be sorted changes significantly.

Default: None; optional. See "Appendix B. Specification/Override of DFSORT Options" on page 549 for full override details.

Applicable Functions: See "Appendix B. Specification/Override of DFSORT Options" on page 549.

HIPRMAX

```
►►──HIPRMAX=──┬─OPTIMAL─┬──────────────────────────────────────────►◄
              ├─n───────┤
              └─p%──────┘
```

Temporarily overrides the HIPRMAX installation option, which specifies the maximum amount of Hiperspace to be committed for Hipersorting. For more information, see the discussion of the HIPRMAX option in "OPTION Control Statement" on page 134.

OPTIMAL
specifies that DFSORT determines dynamically the maximum amount of Hiperspace to be used for Hipersorting.

n specifies that DFSORT determines dynamically the maximum amount of Hiperspace to be used for Hipersorting, subject to a limit of nMB. n must be a value between 0 and 9999. If n is 0, Hipersorting is not used.

p%
specifies that DFSORT determines dynamically the maximum amount of hiperspace to be used for Hipersorting, subject to a limit of p percent of the configured expanded storage. p must be a value between 0 and 100. If p is 0, Hipersorting is not used. The value calculated for p% is limited to 9999MB, and is rounded down to the nearest MB.

Default: Usually the installation default. See "Appendix B. Specification/Override of DFSORT Options" on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 *Messages, Codes and Diagnosis* 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 B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 *Messages, Codes and Diagnosis* for full details on use of the message data set.

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

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

Notes:

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

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

LOCALE

```
►►──LOCALE=──┬─name────────┬──────────────────────────────────────────►◄
             ├─CURRENT─────┤
             └─NONE────────┘
```

Temporarily overrides the LOCALE installation option, which specifies whether locale processing is to be used and, if so, designates the active locale. For more information, see the discussion of the LOCALE option in “OPTION Control Statement” on page 134.

name specifies that locale processing is to be used and designates the name of the locale to be made active during DFSORT processing.

The locales are designated using a descriptive name. For example, to set the active locale to represent the French language and the cultural conventions of Canada, specify LOCALE=FR_CA. You can specify up to 32 characters for the descriptive locale name. The locale names themselves are not case-sensitive. See *Using Locales* for complete locale naming conventions.

You can use IBM-supplied and user-defined locales. See “Appendix F. Locales Supplied with C/370” on page 599 for examples of some IBM-supplied locales.

The state of the active locale prior to DFSORT being entered will be restored on DFSORT’s completion.

CURRENT
specifies that locale processing is to be used, and the current locale active when DFSORT is entered will remain the active locale during DFSORT processing.

NONE specifies that locale processing is not to be used. DFSORT will use the binary encoding of the code page defined for your data for collating and comparing.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

MSGDDN

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

Temporarily overrides the MSGDDN installation option, which specifies an alternate ddname for the message data set. For more information, see the discussion of the MSGDDN option in “OPTION Control Statement” on page 134.

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 run-time, SYSOUT is used instead. For details on using the message data set, see *Messages, Codes and Diagnosis*.

Note: MSGDD=ddname can be used instead of MSGDDN=ddname.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 *Messages, Codes and Diagnosis* for full details on use of the message data set.

ALL
specifies that all messages except diagnostic messages ICE800I to ICE999I are printed on the message data set. Control statements are printed only if LIST is in effect.

CRITICAL
specifies that only critical messages are printed on the message data set. Control statements are printed only if LIST is in effect.

NONE
specifies that no messages or control statements are printed.

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

Option	MSGPRT	MSGCON
NO	NONE	NONE
NOF	NONE	NONE
AB	ALL	ALL
AP	ALL	CRITICAL
AC	NONE	ALL
CB	CRITICAL	CRITICAL
CC	NONE	CRITICAL
CP	CRITICAL	CRITICAL
PC	ALL	ALL
SC	ALL	CRITICAL
SP	CRITICAL	ALL
NOFLAG	NONE	CRITICAL
FLAG(I)	ALL	CRITICAL
FLAG(U)	CRITICAL	CRITICAL

Figure 4. Aliases for MSGPRT/MSGCON Options

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

ODMAXBF

```
►►──ODMAXBF=──┬─n──┬──────────────────────────────────────────────────►◄
              ├─nK─┤
              └─nM─┘
```

Temporarily overrides the ODMAXBF installation option, which specifies the maximum buffer space DFSORT can use for each OUTFIL data set. For more information, see the discussion of the ODMAXBF option in “OPTION Control Statement” on page 134.

n specifies that a maximum of n bytes of buffer space is to be used for each OUTFIL data set. If you specify less than 262144, 262144 is used. If you specify more than 16777216, 16777216 is used.

Limit: 8 digits

nK
specifies that a maximum of n times 1024 bytes of buffer space is to be used for each OUTFIL data set. If you specify less than 256K, 256K is used. If you specify more than 16384K, 16384K is used.

Limit: 5 digits

nM
specifies that a maximum of n times 1048576 bytes of buffer space is to be used for each OUTFIL data set. If you specify 0M, 256K is used. If you specify more than 16M, 16M is used.

Limit: 2 digits

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

OUTREL or NOOUTREL

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

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

OUTREL
specifies that unused temporary output data set space is released.

NOOUTREL
specifies that unused temporary output data set space is not released.

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

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

OVFLO

```
►►──OVFLO=──┬─RC0──┬──────────────────────────────────────────►◄
            ├─RC4──┤
            └─RC16─┘
```

Temporarily overrides the OVFLO installation option, which specifies the action to be taken by DFSORT when BI, FI, PD or ZD summary fields overflow. For more information, see the discussion of the OVFLO option in “OPTION Control Statement” on page 134.

RC0
specifies that DFSORT should issue message ICE152I (once), set a return code of 0 and continue processing when summary fields overflow.

RC4
specifies that DFSORT should issue message ICE152I (once), set a return code of 4 and continue processing when summary fields overflow.

RC16
specifies that DFSORT should issue message ICE195A, terminate and give a return code of 16 when summary fields overflow.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

PAD

```
►►──PAD=──┬─RC0──┬──────────────────────────────────────────►◄
          ├─RC4──┤
          └─RC16─┘
```

Temporarily overrides the PAD installation option, which specifies the action to be taken by DFSORT when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL, for the cases where DFSORT allows LRECL padding. For more information, see the discussion of the PAD option in “OPTION Control Statement” on page 134.

RC0
specifies that DFSORT should issue message ICE171I (once), set a return code of 0 and continue processing when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL.

RC4
specifies that DFSORT should issue message ICE171I, set a return code of 4 and continue processing when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL.

RC16
specifies that DFSORT should issue message ICE196A, terminate and give a return code of 16 when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

RESALL

```
►►──RESALL=──┬─n──┬──────────────────────────────────────────────────►◄
             ├─nK─┤
             └─nM─┘
```

Temporarily overrides the RESALL installation option, which specifies the number of bytes to be reserved in a REGION for system use when SIZE/MAINSIZE=MAX is in effect. For more information, see the discussion of the RESALL option in “OPTION Control Statement” on page 134.

n specifies that n bytes of storage are to be reserved. If you specify less than 4096, 4096 is used.

Limit: 8 digits.

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

Limit: 5 digits.

nM
specifies that n times 1048576 bytes of storage are to be reserved. If you specify 0M, 4K is used.

Limit: 2 digits.

Note: RESERVE=value can be used instead of RESALL=value.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

RESET or NORESET

```
►►──┬─RESET───┬──────────────────────────────────────────────────────►◄
    └─NORESET─┘
```

Temporarily overrides the RESET installation option, which specifies whether DFSORT should process a VSAM output data set defined with REUSE as a NEW or MOD data set.

RESET
specifies that DFSORT processes a VSAM output data set defined with REUSE as a NEW data set. The high-used RBA is reset to zero and the output data set is effectively treated as an initially empty cluster.

NORESET
specifies that DFSORT processes a VSAM output data set defined with REUSE as a MOD data set. The high-used RBA is not reset and the output data set is effectively treated as an initially non-empty cluster.

Note: A VSAM output data set defined without REUSE is processed as a MOD data set.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SDB

```
►►──SDB=──┬─LARGE─┬──────────────────────────────────────────────►◄
          ├─YES───┤
          ├─INPUT─┤
          └─NO────┘
```

Temporarily overrides the SDB installation option, which specifies whether DFSORT should use the system-determined optimum block size for output data sets when the block size is specified as zero or defaulted to zero. For more information, see the discussion of the SDB option in “OPTION Control Statement” on page 134.

LARGE
specifies that DFSORT is to use the system-determined optimum block size for an output data set when its block size is zero. With OS/390 Release 10 and above, SDB=LARGE allows DFSORT to select a block size greater than 32760 bytes for a tape output data set, when appropriate.

YES
specifies that DFSORT is to use the system-determined optimum block size for an output data set when its block size is zero, but is to limit the selected block size to a maximum of 32760 bytes.

INPUT
specifies that DFSORT is to use the system-determined optimum block size for an output data set when its block size is zero, but is to limit the selected block size to a maximum of 32760 bytes if the input block size is less than or equal to 32760 bytes.

NO
specifies that DFSORT is not to use the system-determined optimum block size.

Note: SDB, SDB=ON, and SDB=SMALL can be used instead of SDB=YES. NOSDB and SDB=OFF can be used instead of SDB=NO.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SIZE

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

Temporarily overrides the SIZE installation option, which specifies the amount of main storage available to DFSORT. For more information, see the discussion of the MAINSIZE option in “OPTION Control Statement” on page 134.

n specifies that n bytes of storage are to be allocated. If you specify more than 2097152000, 2097152000 is used.

Limit: 10 digits.

nK
specifies that n times 1024 bytes of storage are to be allocated. If you specify more than 2048000K, 2048000K is used.

Limit: 7 digits.

nM
specifies that n times 1048576 bytes of storage are to be allocated. If you specify more than 2000M, 2000M is used.

Limit: 4 digits.

MAX
instructs DFSORT to calculate the amount of main storage available and allocates this maximum amount, up to the TMAXLIM or MAXLIM installation value, as appropriate for the application.

If you specify less than 4K, 4K is used.

MAX-m
specifies the RESALL value (m) in bytes. MAX-m instructs DFSORT to calculate the amount of storage available and allocate this amount up to the MAX value *minus* the amount of storage reserved for system and application use (RESALL).

If you specify less than 4096 for m, 4096 is used.

Limit for m: 8 digits.

MAX-mK
specifies the RESALL value (m times 1024) in KBs. MAX-mK instructs DFSORT to calculate the amount of storage available and allocate this amount up to the MAX value *minus* the amount of storage reserved for system and application use (RESALL).

If you specify less than 4K for m, 4K is used.

Limit for m: 5 digits.

MAX-mM
specifies the RESALL value (m times 1048576) in s. MAX-mM instructs the program to calculate the amount of storage available and allocate this amount up to the MAX value *minus* the amount of storage reserved for system and application use (RESALL).

If you specify 0M for m, 4K is used.

Limit for m: 2 digits.

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

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 134.

z specifies the number of records to be skipped.

Limit: 28 digits (15 significant digits).

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SOLRF or NOSOLRF

```
►►──┬─SOLRF──┬──────────────────────────────────────────────────────────►◄
    └─NOSOLRF─┘
```

Temporarily overrides the SOLRF installation option, which specifies whether DFSORT should set the SORTOUT LRECL to the reformatted record length when the SORTOUT LRECL is unknown. For more information, see the discussion of the SOLRF and NOSOLRF options in “OPTION Control Statement” on page 134.

SOLRF
specifies that DFSORT should use the reformatted record length for the SORTOUT LRECL when the SORTOUT LRECL is not specified or available.

NOSOLRF
specifies that DFSORT should not use the reformatted record length for the SORTOUT LRECL.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SPANINC

```
►►─SPANINC=──┬─RC0──┬──────────────────────────────────────────►◄
             ├─RC4──┤
             └─RC16─┘
```

Temporarily overrides the SPANINC installation option, which specifies the action to be taken by DFSORT when one or more incomplete spanned records are detected in a variable spanned input data set. For more information, see the discussion of the SPANINC option in “OPTION Control Statement” on page 134.

RC0
specifies that DFSORT should issue message ICE197I (once), set a return code of 0 and eliminate all incomplete spanned records it detects.

RC4
specifies that DFSORT should issue message ICE197I (once), set a return code of 4 and eliminate all incomplete spanned records it detects.

RC16
specifies that DFSORT should issue message ICE204A, terminate and give a return code of 16 when an incomplete spanned record is detected.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549

STIMER or NOSTIMER

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

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

STIMER
specifies that STIMER can be used. Processor-time data appears in SMF records and ICETEXIT statistics.

NOSTIMER
specifies that STIMER cannot be used. Processor-time data does not appear in SMF records or ICETEXIT statistics.

Note: If a user exit takes checkpoints, then STIMER must not be issued.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

STOPAFT

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

Specifies 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 an INCLUDE/OMIT statement). For more information, see the discussion of the STOPAFT option in “OPTION Control Statement” on page 134.

n specifies the maximum number of records to be accepted.

Limit: 28 digits (15 significant digits).

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, DFSORT issues an error message and terminates unless FSZEST=YES was specified at installation time.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SZERO or NOSZERO

```
►►──┬─SZERO───┬──────────────────────────────────────────────────────────►◄
    └─NOSZERO─┘
```

Temporarily overrides the SZERO installation option, which specifies whether DFSORT should treat numeric -0 and +0 values as signed (that is, different) or unsigned (that is, the same) for collation, comparisons, editing, conversions, minimums and maximums. For more information, see the discussion of the SZERO and NOSZERO options in “OPTION Control Statement” on page 134.

SZERO
specifies that DFSORT should treat numeric zero values as signed.

NOSZERO
specifies that DFSORT should treat numeric zero values as unsigned.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

TRUNC

```
►►──TRUNC=──┬─RC0──┬──────────────────────────────────────────►◄
            ├─RC4──┤
            └─RC16─┘
```

Temporarily overrides the TRUNC installation option, which specifies the action to be taken by DFSORT when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL, for the cases where DFSORT allows LRECL truncation. For more information, see the discussion of the TRUNC option in "OPTION Control Statement" on page 134.

RC0
specifies that DFSORT should issue message ICE171I, set a return code of 0 and continue processing when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL.

RC4
specifies that DFSORT should issue message ICE171I, set a return code of 4 and continue processing when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL.

RC16
specifies that DFSORT should issue message ICE196A, terminate and give a return code of 16 when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL.

Default: Usually the installation default. See "Appendix B. Specification/Override of DFSORT Options" on page 549.

Applicable Functions: See "Appendix B. Specification/Override of DFSORT Options" on page 549.

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 performed.

NOVERIFY
specifies that sequence checking is not performed.

Notes:

1. Using VERIFY can degrade performance.
2. SEQ=YES can be used instead of VERIFY. SEQ=NO can be used instead of NOVERIFY.

Default: Usually the installation default. See "Appendix B. Specification/Override of DFSORT Options" on page 549 for full override details.

Applicable Functions: See "Appendix B. Specification/Override of DFSORT Options" on page 549.

VLLONG or NOVLLONG

```
►►──┬─VLLONG──┬──────────────────────────────────────────────────────►◄
    └─NOVLLONG─┘
```

Temporarily overrides the VLLONG installation option, which specifies whether DFSORT is to truncate ″long″ variable-length output records. For more information, see the discussion of the VLLONG and NOVLLONG options in “OPTION Control Statement” on page 134.

VLLONG
specifies that DFSORT truncates long variable-length output records to the LRECL of the SORTOUT or OUTFIL data set.

NOVLLONG
specifies that DFSORT terminates if a long variable-length output record is found.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

VLSCMP or NOVLSCMP

```
►►──┬─VLSCMP──┬──────────────────────────────────────────────────────►◄
    └─NOVLSCMP─┘
```

Temporarily overrides the VLSCMP installation option, which specifies whether DFSORT is to pad ″short″ variable-length INCLUDE/OMIT compare fields with binary zeroes. For more information, see the discussion of the VLSCMP and NOVLSCMP options in “OPTION Control Statement” on page 134.

VLSCMP
specifies that short variable-length compare fields are padded with binary zeros.

NOVLSCMP
specifies that short variable-length compare fields are not padded.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

VLSHRT or NOVLSHRT

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

Temporarily overrides the VLSHRT installation option, which specifies whether DFSORT continues processing if a ″short″ variable-length SORT/MERGE control field or INCLUDE/OMIT compare field is found. For more information, see the discussion of the VLSHRT and NOVLSHRT options in “OPTION Control Statement” on page 134.

VLSHRT
specifies that DFSORT continues processing if a short control field or compare field is found.

NOVLSHRT
specifies that DFSORT terminates if a short control field or compare field is found.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

VSAMEMT or NVSAMEMT

```
►►──┬─VSAMEMT──┬──────────────────────────────────────────────────►◄
    └─NVSAMEMT─┘
```

Temporarily overrides the VSAMEMT installation option, which specifies whether DFSORT should accept an empty VSAM input data set.

VSAMEMT
specifies that DFSORT accepts an empty VSAM input data set and processes it as having zero records.

NVSAMEMT
specifies that DFSORT terminates if an empty VSAM input data set is found.

Note: VSAMEMT=YES can be used instead of VSAMEMT. VSAMEMT=NO can be used instead of NVSAMEMT.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549

VSAMIO or NOVSAMIO

```
►►──┬─VSAMIO──┬──────────────────────────────────────────────────►◄
    └─NOVSAMIO─┘
```

Temporarily overrides the VSAMIO installation option, which specifies whether DFSORT should allow a VSAM data set defined with REUSE to be sorted in-place.

VSAMIO
specifies that DFSORT can use the same VSAM data set for input and output when all of the following conditions are met:

- The application is a sort.
- RESET is in effect.
- The VSAM data set was defined with REUSE.

These conditions ensure that the VSAM data set is processed as NEW for output and will contain the sorted input records; that is it will be sorted in-place.

DFSORT terminates if the same VSAM data set is specified for input and output, and any of the above conditions are not met.

NOVSAMIO
specifies that DFSORT terminates if the same VSAM data set is specified for input and output.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

WRKREL or NOWRKREL

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

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

WRKREL
specifies that unused space is released.

NOWRKREL
specifies that unused space is not released.

Notes:

1. If you have dedicated certain volumes for SORTWKdd data sets, and you do not want unused temporary space to be released, you should specify NOWRKREL.
2. If WRKREL is in effect, DFSORT releases space for the SORTWKdd data sets just prior to termination. Space is released only for those SORTWKdd data sets that were used for the sort application.
3. RELEASE=OFF and RLS=0 can be used instead of NOWRKREL. RELEASE=ON and RLS=n (n greater than 0) can be used instead of WRKREL.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

WRKSEC or NOWRKSEC

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

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

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

NOWRKSEC
specifies that automatic secondary allocation for temporary JCL SORTWKdd data sets is not used.

Note: SECOND=OFF and SEC=0 can be used instead of NOWRKSEC. SECOND=ON and SEC=n (n greater than 0) can be used instead of WRKSEC.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Y2PAST

```
►►──Y2PAST=──┬─s─┬──────────────────────────────────────────────────►◄
             └─f─┘
```

Temporarily overrides the Y2PAST installation option, which specifies the sliding (s) or fixed (f) century window. The century window is used with DFSORT’s Y2 formats to correctly interpret two-digit year data values as four-digit year data values.

s specifies the number of years DFSORT is to subtract from the current year to set the beginning of the sliding century window. Since the Y2PAST value is subtracted from the current year, the century window slides as the current year changes. For example, Y2PAST=81 would set a century window of 1915-2014 in 1996 and 1916-2015 in 1997. s must be a value between 0 and 100.

f specifies the beginning of the fixed century window. For example, Y2PAST=1962 would set a century window of 1962-2061. f must be a value between 1000 and 3000.

Note: CENTWIN=value can be used instead of Y2PAST=value.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

ZDPRINT or NZDPRINT

```
►►──┬─ZDPRINT──┬──────────────────────────────────────────────────►◄
    └─NZDPRINT─┘
```

Temporarily overrides the ZDPRINT installation option, which specifies whether positive zoned-decimal (ZD) fields resulting from summing must be converted to printable numbers. For more information, see the discussion of the ZDPRINT and NZDPRINT options in “OPTION Control Statement” on page 134.

ZDPRINT
means convert positive ZD summation results to printable numbers.

NZDPRINT
means do not convert positive ZD summation results to printable numbers.

Note: ZDPRINT=YES can be used instead of ZDPRINT. ZDPRINT=NO can be used instead of NZDPRINT.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Aliases for PARM Options

For compatibility reasons, the following EXEC/DFSPARM PARM options can be specified by using the aliases listed below. See the indicated PARM options for complete details.

Table 4. Aliases for PARM Options

Alias	PARM Option
CENTWIN=value	Y2PAST=value
CORE=value	SIZE=value
FLAG(I)	MSGPRT=ALL
FLAG(U)	MSGPRT=CRITICAL
L5=value	AVGRLEN=value
MSG=value	MSGPRT=value
MSGDD=value	MSGDDN=value
NOFLAG	MSGPRT=NONE
NORC16	NOABEND
NORLSOUT	NOOUTREL
NOSDB	SDB=NO
RC16=ABE	ABEND
RELEASE=ON	WRKREL
RELEASE=OFF	NOWRKREL
RESERVE=value	RESALL=value
RESERVEX=value	ARESALL=value
RLS=n	WRKREL
RLS=0	NOWRKREL
RLSOUT	OUTREL
SDB	SDB=YES
SDB=ON	SDB=YES
SDB=OFF	SDB=NO
SDB=SMALL	SDB=YES
SEC=n	WRKSEC
SEC=0	NOWRKSEC

Table 4. Aliases for PARM Options (continued)

Alias	PARM Option
SECOND=ON	WRKSEC
SECOND=OFF	NOWRKSEC
SEQ=YES	VERIFY
SEQ=NO	NOVERIFY
VSAMEMT=YES	VSAMEMT
VSAMEMT=NO	NVSAMEMT
ZDPRINT=YES	ZDPRINT
ZDPRINT=NO	NZDPRINT

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 62)
- Program DD statements (discussed in detail in “Program DD Statements” on page 64).

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 5. The subparameters of the DCB parameter (a DD statement parameter) are described similarly in Table 6 on page 60.

Notes:

1. 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 12.
2. When using DFSORT applications, FREE=CLOSE cannot be used on any DD statements except DFSPARM.

Table 5. DD Statement Parameters Used by DFSORT

Parameter	When Required	Parameter Values	Default Value
{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.
DCB	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 Table 6 on page 60.)
DISP	When the default value is not applicable.	Indicates the status and disposition of the data set.	The system assumes (NEW, DELETE).

Table 5. DD Statement Parameters Used by DFSORT (continued)

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.
LABEL	When the default value is not applicable.	Specifies information about labeling and retention for the data set.	The system assumes standard labeling.
SPACE	When the DD statement defines a new data set on direct access.	Specifies the amount of space needed to contain the data set.	None.
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.	None.
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.	None.

Table 6. DCB Subparameters Used by DFSORT

Subparameter	Condition When Required	Subparameter Values	Default Value
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.	
DEN	When the data set is located on a 7-track tape unit.	Specifies the density at which the tape was recorded.	800 bpi
OPTCD	When processing data in ISCII/ASCII format.	Specifies that the tape processed is in ISCII/ASCII format.	
TRTCH	When the data set is located on a tape device with IDRC and system IDRC is not used.	Specifies whether data set is compacted.	System default option.

Table 6. DCB Subparameters Used by DFSORT (continued)

Subparameter	Condition When Required	Subparameter Values	Default Value
BLKSIZE [1, 2]	When the DCB parameter is required and the default value is not suitable except on SORTWKdd statements.	Specifies the maximum length (in bytes) of the physical records in the data set.	• For old data sets, the value in the data set label. • For new output data sets, appropriate values based on the input data set or RECORD statement values. Unless SDB=NO is in effect, Blockset uses the system-determined optimum block size when the output data set block size is zero. Applications which require a specific output data set block size should be changed to specify that block size **EXPLICITY**. • No default if input on unlabeled tape or BLP or NSL specified.
LRECL [2, 3]		Specifies the maximum length (in bytes) of the logical records in the data set.	
RECFM		Specifies the format of the records in the data set.	

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, SORTIN0, SORTIN1...SORTIN9 can be specified *instead of* SORTIN00, SORTIN01...SORTIN09, respectively. If you specify both SORTINn and SORTIN0n 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 and only SORTIN2 is used.

Note: For a Conventional merge, SORTINn will not be recognized because of the existing restriction which allows only SORTIN01, SORTIN02...SORTIN16. Duplicates of these accepted ddnames will be ignored.

Duplicate OUTFIL ddnames are ignored at the OUTFIL statement level as explained in “OUTFIL Statements Notes” on page 236.

Shared Tape Units

The following pairs of DFSORT data sets can be assigned to a single tape unit:
- The SORTIN data set and the SORTWK01 data set (tape work data set sorts only)
- The SORTIN data set and the SORTOUT data set or one OUTFIL data set (sort applications only).

1. See “SORTIN DD Statement” on page 66 and “SORTINnn DD Statement” on page 67.
2. This is the only subparameter allowed for DD * data sets.
3. For padding and truncating fixed-length records, see “Data Set Notes and Limitations” on page 12.

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 the SORTIN data set with the SORTOUT data set or an OUTFIL data set by including UNIT=AFF=SORTIN in the SORTOUT or OUTFIL DD statement.

SORTINnn tape data sets must all be on different tape units because they are read concurrently. SORTOUT and OUTFIL tape data sets must all be on different tape units because they are written concurrently.

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
Defines your program link library if it is not already known to the system.

//STEPLIB DD
Same as //JOBLIB DD.

//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 be defined with a RECFM of F or FB. The LRECL can be 80, or more (when valid). If the LRECL is greater than 80, DFSORT will use the first 80 bytes of each record.

  If user exit routines are in SYSIN, the LRECL must be 80.
- DFSORT supports concatenated SYSIN data sets to the extent that the system supports “like” concatenated data sets for BSAM. Refer to *Using Data Sets* for further information about “like” concatenated data sets.

Note: The OPTION statement keywords EFS, LIST, NOLIST, LISTX, NOLISTX, LOCALE, MSGPRT, MSGDDN, SMF, SORTDD, SORTIN, and SORTOUT are used only when they are passed by an extended parameter list or when 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 74.

If user exit routines are in SYSIN, make sure that:

- The LRECL of SYSIN is 80.
- 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 DFSORT is program invoked, 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 DFSORT message data set. The default ddname is SYSOUT, but you can specify an alternate ddname for the message data set using the MSGDDN installation or run-time option. Always supply a DD statement for the message data set if a catalogued procedure is not used. (If you are invoking DFSORT from a COBOL program and are using the ddname SYSOUT for the message data set, the use of EXHIBIT or DISPLAY in your COBOL program can produce uncertain printing results.)

DFSORT uses RECFM=FBA, LRECL=121, and the specified BLKSIZE for the message data set. If the BLKSIZE you specify is not a multiple of 121, DFSORT uses BLKSIZE=121. If you do not specify the BLKSIZE, DFSORT selects the block size as directed by the SDBMSG installation option (see *Installation and Customization*).

If you use a temporary or permanent message data set, it is best to specify a disposition of MOD to ensure you see all messages and control statements in the message data set.

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

//SYSMDUMP DD
Same as //SYSUDUMP DD.

//SYSABEND DD
Same as //SYSUDUMP DD.

If you are using the supplied SORT cataloged procedure, the DD statements 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
Defines the intermediate storage data set for the linkage editor.

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

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

Note: If you do not include user routines, or if you include user routines that do *not* require link-editing, you can usethe 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. Can usually be omitted.

//SYMNAMES DD
Defines the SYMNAMES data set containing statements to be used for symbol processing. Required only if symbol processing is to be performed.

//SYMNOUT DD
Defines the data set in which SYMNAMES statements and the symbol table are to be listed. Optional if SYMNAMES DD is specified. Otherwise ignored.

//SORTIN DD
Defines the input data set for a sorting or copying application. Will not be used for a merging application.

//SORTINnn DD
Defines the input data sets for a merging application. Will not be used for a sorting or copying application.

//SORTWKdd 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.

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

//outfil DD
Defines an OUTFIL output data set for a sorting, merging, or copying application.

//SORTCKPT DD
Defines the data set used to store the information that the system needs to restart the sort from the last checkpoint. This is only needed 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 program-invoked.

//DFSPARM DD
Defines the data set from which both additional or changed DFSORT program control statements and EXEC statement PARM options can be read when DFSORT is directly invoked or program invoked.

//SORTDKdd DD
Defines the data set used for a VIO SORTWKdd allocation by DFSORT if it is dynamically reallocated; SORTDKdd must never be specified in the job stream.

//SORTDIAG DD
Specifies that all messages and control statements are printed. Used primarily for diagnostics and debugging.

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

//SORTMODS DD
Defines a temporary partitioned data set. This temporary data set must be large enough to contain all your user 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 processing.

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: If ICEMAC option SORTLIB=PRIVATE is in effect or dynamic link edit of user exits is specified:

- For sort applications using tape work data sets
- For 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 workdata set sorts and Conventional merges.

Example 1 SORTLIB DD Statement:

```
//SORTLIB DD DSNAME=USORTLIB,DISP=SHR
```

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 existed before this job step, that it should be kept after this job step, and that it can be used concurrently by several jobs (SHR). None of the jobs should change the data set in any way.

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 5 on page 59. The subparameters of the DCB

parameter are described in the same detail in Table 6 on page 60. For more detailed information, see *JCL Reference* and *JCL User's Guide*

SYMNAMES DD and SYMNOUT DD Statements

See "Chapter 7. Using Symbols for Fields and Constants" on page 433 for details.

SORTIN DD Statement

The SORTIN DD statement describes the characteristicsof the data set in which the records to be sorted or copied reside and also indicates its location.

When Required: A SORTIN DD statement is required for all sort or copy applications, unless you provide an E15 user 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 user exit in the parameter list.

Data Set Characteristics: DFSORT accepts empty and null non-VSAM data sets for sorting and copying (be sure to supply DCB parameters). DFSORT also accepts empty VSAM data sets for sorting and copying provided VSAMEMT is in effect. 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. If the data set is a null multivolume data set and the DS1IND80 flag is off in the format-1 DSCB of the first volume of the multivolume data set, DFSORT opens the data set for output to force an end of file (EOF) mark before using the data set for input.

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. The "System Code" field in the data set label in the DASD Volume Table of Contents (DSCB in the VTOC) indicates a data set created by the VSE operating system if it contains the letters DOS or VSE within it. Such data sets are never treated as null; however, they may be empty. DFSORT cannot process VSE DASD data sets that do not have DOS or VSE within the System Code field.

See "Data Set Considerations" on page 11 for additional considerations.

The following rules apply to concatenated data sets:

- RECFM must be either all fixed-length or all variable-length for the data sets in the concatenation.
- BLKSIZE can vary. However, if a tape data set has the largest block size and is not first in the concatenation, you must specify BLKSIZE explicitly on its DD statement in the following two situations:
  - Blockset is selected and the tape data set has a block size greater than 32760 bytes, but is not DFSMSrmm-managed, or
  - Blockset is not selected.
- With fixed-length records, LRECL must be the same for all data sets. With variable-length records, LRECL can vary, but the first data set must have the largest LRECL.
- If the data sets are on unlike devices, you cannot use the EXLST parameter at user exit E18.
- If Blockset is not selected and BSAM is used, all null data sets must precede all non-null data sets; otherwise, the results are unpredictable.
- DFSORT forces an EOF mark on all null data sets whose format-1 DSCB DS1IND80 flag is off before using BSAM to process the null data sets.

- If you define a data set using the DUMMY parameter, do not concatenate other data sets to it; the system ignores data sets concatenated to a DUMMY data set.
- VSAM data sets must not be concatenated (system restriction).
- Input cannot consist of both VSAM and non-VSAM data sets.

General Coding Notes:

- For a copy application, the SORTIN data set should not be the same as the SORTOUT data set or any OUTFIL data set because this can cause lost or incorrect data or unpredictable results.
- For a sort application, the SORTIN data set should not be the same as any SORTWKdd data set because this can cause lost or incorrect data or unpredictable results. The SORTIN data set can be the same as the SORTOUT data set or an OUTFIL data set, but this situation can lead to the loss of the data set if the sort application does not end successfully.
- FREE=CLOSE cannot be specified. User labels are not copied.

Example 2 SORTIN DD Statement:

```
//SORTIN DD DSNAME=INPUT,DISP=SHR
```

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 existed before this job step, that it should be kept after this job step, and that it can be used concurrently by several jobs (SHR). None of the jobs should change the data set in any way.

Example 3 Volume Parameter on SORTIN DD:

```
//SORTIN   DD  DSN=SORTIN,DISP=(OLD,KEEP),UNIT=3490,
//               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 characteristicsof 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 user exit E32.

Data Set Characteristics: Input data sets can be either non-VSAM or VSAM, but not both. DFSORT accepts empty and null non-VSAM data sets for merging (be sure to supply DCB parameters). DFSORT also accepts empty VSAM data sets for merging provided VSAMEMT is in effect. 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. 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.

RECFM must be the same for all input data sets.

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

With fixed-length records, LRECL must be the *same* for all data sets. With variable-length records, LRECL can vary.

Data sets can be multivolume but not concatenated. If a SORTINnn data set is multivolume and null, DFSORT forces an EOF mark on the data set before use.

See “Data Set Notes and Limitations” on page 12 for additional considerations.

General Coding Notes:

- A SORTINnn data set should not be the same as the SORTOUT data set or any OUTFIL data set because this can cause lost or incorrect data or unpredictable results.
- You can merge up to 100 data sets with Blockset merge or up to 16 data sets with Conventional merge. If Conventional merge is selected, check message ICE800I for the reason Blockset could not be used and correct the indicated condition, if possible.
  - With Blockset merge: nn can be any integer from 00 (the initial zero is optional) to 99, in any order. Blockset merge treats ddnames of the form SORTINn and SORTIN0n as duplicates, and ignores any occurrences after the first. For example, if you have :

    ```
    //SORTIN4 DD . . .
    //SORTIN04 DD . . .
    ```

    the SORTIN04 DD will be ignored.
  - With Conventional merge: nn can range from 01 to 16. The first number you use must be 01 and the remainder must follow in numeric order. Numbers cannot be skipped. Conventional merge cannot use ddnames of the form SORTIN0-SORTIN9, SORTIN00 or SORTIN17-SORTIN99.
- FREE=CLOSE cannot be specified. User labels are not copied.

Example 4 SORTIN01-03 DD Statements (Merge):

```
//SORTIN01 DD  DSNAME=MERGE1,VOLUME=SER=000111,DISP=OLD,
//             LABEL=(,NL),UNIT=3590,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=32000)
//SORTIN02 DD  DSNAME=MERGE2,VOLUME=SER=000121,DISP=OLD,
//             LABEL=(,NL),UNIT=3590,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=32000)
//SORTIN03 DD  DSNAME=MERGE3,VOLUME=SER=000131,DISP=OLD,
//             LABEL=(,NL),UNIT=3590,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=32000)
```

Example 5 SORTIN01-02 DD Statements (Merge):

```
//SORTIN01 DD  DSNAME=INPUT1,VOLUME=SER=000101, *
//             UNIT=3390,DISP=OLD               *DCB PARAMETERS
//SORTIN02 DD  DSNAME=INPUT2,VOLUME=SER=000201, *SUPPLIED FROM
//             UNIT=3390,DISP=OLD               *LABELS
```

SORTWKdd DD Statement

The SORTWKdd 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.

Up to 255 SORTWKdd DD statements can be specified. However, if you specify more than 32 and the Blockset technique is not selected, only the first 32 are used.

When Required: One or more SORTWKdd statements are required for each sort application (but not a merge or copy), unless:

- Input can be contained in main storage
- Dynamic work space allocation has been requested (DYNALLOC)
- Hipersorting or dataspace sorting is used.

For information on using work data sets, see “Appendix A. Using Work Space” on page 539.

Diagnostic message ICE803I gives information on intermediate storage allocation and use.

Devices: SORTWKdd data sets can be on DASD or on tape, but not both. DASD 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 seven-track.

General Coding Notes:

- Unless the input file is very large, two or three SORTWKdd data sets are usually sufficient. Two or three large SORTWKdd data sets are preferable to several small data sets. Placing each SORTWKdd data set on a separate device can improve performance.

  For optimum allocation of resources such as virtual storage, avoid specifying a large number of work data sets unnecessarily.
- A SORTWKdd data set should not be the same as the SORTIN data set, the SORTOUT data set, any OUTFIL data set, or any other SORTWKdd data set because this can cause lost or incorrect data or unpredictable results.

- Cylinder allocation is preferable for performance reasons. Temporary SORTWKdd data sets allocated in tracks or blocks (without ROUND) are readjusted to cylinders by DFSORT.
- For DASD work data sets, any valid ddname of the form SORTWKdd or SORTWKd can be used (for example, SORTWK01, SORTWKC3, SORTWK2, SORTWK#5, SORTWKA, SORTWKXY and so on). The ddnames can be in any order. SORTWKd and SORTWK0d are not treated as duplicate ddnames (for example, SORTWK5 and SORTWK05 will both be used if specified, as will SORTWKQ and SORTWK0Q). If you specify more than 255 ddnames and the Blockset technique is selected, only the first 255 ddnames are used. If you specify more than 32 ddnames and the Blockset technique is not selected, only the first 32 ddnames are used.
- For tape work data sets, at least three SORTWKdd data sets must be specified. The first three ddnames must be SORTWK01, SORTWK02 and SORTWK03. Subsequent ddnames, if specified, must be in order from SORTWK04 through SORTWK32, with no numbers skipped,
- FREE=CLOSE cannot be specified.
- Spool, dummy, pipe, and HFS data sets, and HFS files, must not be specified as work data sets.
- Parameters relating to ISCII/ASCII data should not be included for tape work data sets.

DASD Work Data Set 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 SORTWKdd 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 and 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 SORTWKdd data sets are, therefore, treated as single-volume SORTWKdd data sets.
- If primary space is fragmented, all but the first fragment are handled as secondary space.

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

- With VIO=NO: DFSORT performs dynamic reallocation using the ddname SORTDKdd 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 an ICE083A message; see *Messages, Codes and Diagnosis* for more information about this error. Non-VIO SORTWKdd data sets are also reallocated when VIO SORTWKdd data sets are present.
- With VIO=YES: the virtual device is used; performance may be degraded.

The following is an example of a SORTWKdd DD statement using a DASD work data set:

Example 6 SORTWK01 DD Statement, DASD Work Data Set:

```
//SORTWK01  DD  SPACE=(CYL,(15,5)),UNIT=3390
```

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,CATLG)
```

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

```
//SORTWK01  DD  DSNAME=name1,UNIT=3390,SPACE=(CYL,(15,5)),
//              DISP=(NEW,DELETE,CATLG)
```

This following is an example of SORTWKdd DD statements using three tape devices.

Example 7 SORTWK01-03 DD Statement, Tape Intermediate Storage:

```
//SORTWK01  DD  UNIT=3480,LABEL=(,NL)
//SORTWK02  DD  UNIT=3480,LABEL=(,NL)
//SORTWK03  DD  UNIT=3480,LABEL=(,NL)
```

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 3480 tape units. Because the DSNAME parameters are omitted, the system assigns unique names.

SORTOUT and OUTFIL DD Statements

The SORTOUT and OUTFIL DD statements describe the characteristics of the data sets in which the processed records are to be placed and indicate their location.

The SORTOUT DD statement specifies the single non-OUTFIL output data set for a sort, copy, or merge application. OUTFIL processing does not apply to SORTOUT.

The FNAMES and/or FILES parameters of one or more OUTFIL statements specify the ddnames of the OUTFIL data sets for a sort, copy, or merge application. The parameters specified for each OUTFIL statement define the OUTFIL processing to be performed for the OUTFIL data sets associated with that statement. For specific information about OUTFIL processing, see “OUTFIL Control Statements” on page 182.

Although the ddname SORTOUT can actually be used for an OUTFIL data set, the term “SORTOUT” will be used to denote the single non-OUTFIL output data set.

When Required: Each ddname specified in an OUTFIL statement requires a corresponding DD statement for that OUTFIL data set.

If you do not specify OUTFIL statements, a SORTOUT DD statement is required unless you provide an E35 user exit that disposes of all output. A SORTOUT DD statement is ignored if your program invokes DFSORT and passes the address of an E35 user exit in the parameter list.

SORTOUT and OUTFIL DD Statements

If you specify OUTFIL statements, you do not have to specify a SORTOUT DD statement or an E35 user exit, although you can use either or both.

Data Set Characteristics: See “Data Set Considerations” on page 11 for additional considerations.

Block size: Unless SDB=NO is in effect, Blockset uses the system-determined optimum block size in most cases when the output data set block size is zero. See the discussion of the SDB option in “OPTION Control Statement” on page 134 for complete details about DFSORT’s use of system-determined block size.

For some jobs, the selection of a larger output data set block size can require an increase in the amount of storage needed for successful DFSORT processing.

Applications which require a specific output data set block size should be changed to specify that block size **explicitly**.

If SDB=NO is in effect, DFSORT selects an appropriate (though not necessarily optimum) block size for the output data set based on the available attributes from the output data set, the input data set, and the RECORD statement. The output data set block size will not necessarily be the same as the input block size.

Reblockable Indicator: DFSORT sets the reblockable indicator in the output data set label when:

Blockset is selected and

- DFSORT sets the system-determined optimum block size for the output data set (see “Block size”) or
- Allocation sets the system-determined optimum block size for the output data set before DFSORT gets control.

General Coding Notes:

- For a copy application, neither the SORTOUT data set nor any OUTFIL data set should be the same as the SORTIN data set because this can cause lost or incorrect data or unpredictable results.
- For a merge application, neither the SORTOUT data set nor any OUTFIL data set should be the same as any SORTINnn data set because this can cause lost or incorrect data or unpredictable results.
- For a sort application, the SORTOUT data set or an OUTFIL data set can be the same as the SORTIN data set, but this situation can lead to the loss of the data set if the sort application does not end successfully.
- An OUTFIL data set should not be the same as the SORTOUT data set or any other OUTFIL data set because this can cause lost or incorrect data or unpredictable results.
- Do not specify OPTCD=W for a full function IBM 3480 tape unit; it is overridden. For a 3480 operating in 3420 compatibility mode (specified as 3400-9), the OPTCD=W request is not overridden, but performance might be degraded.
- If no secondary allocation is requested for a temporary or new output data set, automatic secondary allocation will be used unless NOOUTSEC is in effect.
- The RECFM, LRECL, and BLKSIZE in a tape label are used only for a tape output data set with DISP=MOD, a DD volser present, and an AL, SL, or NSL label, when appropriate.
- FREE=CLOSE cannot be specified.

- See the discussion of the SOLRF and NOSOLRF options in “OPTION Control Statement” on page 134 for information related to the SORTOUT LRECL.

Example 8 SORTOUT DD Statement:

```
//SORTOUT DD  DSN=C905460.OUTPT,UNIT=3390,SPACE=(CYL,5),
//            DISP=(NEW,CATLG)
```

DISP
specifies the data set unknown to the operating system (NEW) and catalogs (CATLG) it under the name C905460.OUTPT.

DSNAME
specifies that the data set is called C905460.OUTPT.

SPACE
requests five cylinders of storage for the data set.

UNIT
Indicates that the data set is on a 3390.

SORTCKPT DD Statement

The SORTCKPT data set can be allocated on any device that operateswith 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,
//             DSP=(NEW,KEEP),UNIT=3480
```

When you allocate the SORTCKPT data set, you must include at least one work data set.

If the CKPT operand is specified on the OPTION or SORT control statement, more intermediate storage could be required.

If you want to use the Checkpoint/Restart Facility, refer to “Checkpoint/Restart” on page 595.

SORTCNTL DD Statement

The SORTCNTL data set can be used to supplyDFSORT 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 be defined with RECFM of F or FB. The LRECL can be 80, or more (when valid). If the LRECL is greater than 80, DFSORT will use the first 80 bytes of each record.
- DFSORT supports concatenated SORTCNTL data sets to the extent that the system supports “like” concatenated data sets for BSAM. Refer to *Using Data Sets* for further information about “like” 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.

SORTCNTL DD Statement

Example 10 SORTCNTL DD Statement:

```
//SORTCNTL DD  *
  OPTION MAINSIZE=8M
```

Notes:

1. The OPTION statement keywords EFS, LIST, NOLIST, LISTX, NOLISTX, LOCALE, MSGPRT, MSGDDN, SMF, SORTDD, SORTIN, and SORTOUT are used only when they are passed by an extended parameter list or when 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 in “OPTION Control Statement” on page 134.
2. If you use the DFSPARM DD statement instead of the SORTCNTL DD statement, 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 B. Specification/Override of DFSORT Options” on page 549.

DFSPARM DD Statement

The DFSPARM DD statement can be used to supply DFSORT program control statements and EXEC statement PARM optionsfrom a single DD source. Because statements in the DFSPARM data set are read whether DFSORT is program invoked or directly invoked, 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.

For examples of using DFSPARM when you call DFSORT from a program, see “Overriding DFSORT Control Statements from Programs” on page 332.

Full override and applicability details are listed below and in “Appendix B. Specification/Override of DFSORT Options” on page 549.

- 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 be defined with RECFM of F or FB. The LRECL can be 80, or more (when valid). If the LRECL is greater than 80, DFSORT will use the first 80 bytes of each record.
- DFSORT supports concatenated DFSPARM data sets to the extent that the system supports “like” concatenated data sets for BSAM. Refer to *Using Data Sets* for further information about “like” 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. The following exceptions apply:

- 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.

FREE=CLOSE can be used for applicable DFSPARM data sets (for example, with temporary and permanent sequential data sets, but not with DD * data sets).

When DFSORT is called from another program, FREE=CLOSE causes the DFSPARM data set to be released when DFSORT returns to the caller. This allows another DFSPARM data set to be used for a subsequent call.

For example, if a COBOL program contains three SORT verbs, the following would cause the control statements in DP1 to be used for the first SORT verb, the control statements in DP2 to be used for the second SORT verb, and the control statements in DP3 to be used for the third SORT verb:

```
//DFSPARM  DD  DSN=DP1,DISP=SHR,FREE=CLOSE
//DFSPARM  DD  DSN=DP2,DISP=SHR,FREE=CLOSE
//DFSPARM  DD  DSN=DP3,DISP=SHR,FREE=CLOSE
```

Without FREE=CLOSE, DP1 would be used for all three SORT verbs.

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. SORT and OPTION are control statements. ABEND and STOPAFT=500 are 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/DFSPARM PARM Options” on page 29 and “Chapter 3. Using DFSORT Program Control Statements” on page 77.

SORTDKdd DD Statement

SORTWKdd data sets can be assigned to VIO. If the ICEMAC parameter VIO is specified or defaults to NO, SORTWKdd data sets are deallocated and reallocated by DFSORT using SORTDKdd ddnames. SORTDKdd ddnames are 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 and 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, a SYSOUT DD statement or a ddname DD statement (where ddname is the alternate message data set ddname specified during installation or run-time) should be provided. If ICEMAC option NOMSGDD=QUIT is in effect and 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 wherethe snap dumps requested by the ESTAE recovery routine, or the 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

Using Program Control Statements . . . 79
Control Statement Summary . . . 80
  Describing the Primary Task . . . 80
  Including or Omitting Records . . . 80
  Reformatting and Editing Records . . . 80
  Producing Multiple Output and Reports and Converting Records . . . 81
  Invoking Additional Functions and Options . . . 81
  Using Symbols . . . 81
General Coding Rules . . . 81
  Continuation Lines . . . 83
  Inserting Comment Statements . . . 84
  Coding Restrictions . . . 84
    EFS Restrictions When an EFS Program Is in Effect . . . 84
    Using Control Statements from Other IBM Programs . . . 85
ALTSEQ Control Statement . . . 85
  Altering EBCDIC Collating Sequence—Examples . . . 86
    Example 1 . . . 86
    Example 2 . . . 86
    Example 3 . . . 86
    Example 4 . . . 86
DEBUG Control Statement . . . 87
  Specifying Diagnostic Options—Examples . . . 91
    Example 1 . . . 91
    Example 2 . . . 92
END Control Statement . . . 92
  Discontinue Reading Control Statements—Examples . . . 92
    Example 1 . . . 92
    Example 2 . . . 92
INCLUDE Control Statement . . . 93
  Relational Condition . . . 95
  Comparisons . . . 95
    Relational Condition Format . . . 95
    Padding and Truncation . . . 100
    Cultural Environment Considerations . . . 100
  Including Records in the Output Data Set—Comparison Examples . . . 100
    Example 1 . . . 100
    Example 2 . . . 101
    Example 3 . . . 101
    Example 4 . . . 101
    Example 5 . . . 102
  Substring Comparison Tests . . . 102
    Relational Condition Format. . . . 102
  Including Records in the Output Data Set—Substring Comparison Example 103
    Example . . . 103
  Bit Logic Tests . . . 104
  Method 1: Bit Operator Tests . . . 104
    Relational Condition Format. . . . 104
    Fields . . . 105
    Mask . . . 105
  Padding and Truncation . . . 106
  Including Records in the Output Data Set—Bit Operator Test Examples 106
    Example 1 . . . 106
    Example 2 . . . 106

© Copyright IBM Corp. 1973, 2000

Example 3 . . . 106
Method 2: Bit Comparison Tests . . . 107
Relational Condition Format. . . . 107
Fields . . . 107
Bit Constant . . . 107
Padding and Truncation . . . 108
Including Records in the Output Data Set—Bit Comparison Test Examples 108
Example 1 . . . 108
Example 2 . . . 108
Example 3 . . . 108
Date Comparisons . . . 109
Relational Condition Format. . . . 109
Including Records in the Output Data Set—Date Comparisons . . . 111
Example 1 . . . 111
Example 2 . . . 111
INCLUDE/OMIT Statement Notes. . . . 112
INREC Control Statement . . . 112
INREC Statement Notes . . . 119
Reformatting Records Before Processing — Examples. . . . 120
Example 1 . . . 120
Example 2 . . . 121
Example 3 . . . 122
Example 4 . . . 122
Example 5 . . . 123
MERGE Control Statement . . . 124
Specifying a MERGE or COPY—Examples . . . 126
Example 1 . . . 126
Example 2 . . . 127
Example 3 . . . 127
Example 4 . . . 127
MODS Control Statement . . . 127
Identifying User Exit Routines—Examples . . . 130
Example 1 . . . 130
Example 2 . . . 130
OMIT Control Statement . . . 130
Omitting Records from the Output Data Set—Example . . . 133
Example . . . 133
OPTION Control Statement . . . 134
Aliases for OPTION Statement Options . . . 177
Specifying DFSORT Options or COPY—Examples . . . 178
Example 1 . . . 178
Example 2 . . . 178
Example 3 . . . 179
Example 4 . . . 179
Example 5 . . . 179
Example 6 . . . 179
Example 7 . . . 180
Example 8 . . . 181
Example 9 . . . 181
OUTFIL Control Statements. . . . 182
OUTFIL Statements Notes . . . 236
OUTFIL Features—Examples . . . 239
Example 1 . . . 239
Example 2 . . . 240
Example 3 . . . 241
Example 4 . . . 243

Example 5 246
Example 6 247
Example 7 247
Example 8 248
Example 9 248
Example 10 249
Example 11. 250
Example 12 250
OUTREC Control Statement 251
OUTREC Statement Notes 256
Reformatting the Output Record—Examples 257
Example 1 257
Example 2 258
Example 3 258
Example 4 258
Example 5 258
Example 6 259
Example 7 259
RECORD Control Statement 260
Describing the Record Format and Length—Examples 263
Example 1 263
Example 2 264
SORT Control Statement. 264
SORT/MERGE Statement Notes 272
Specifying a SORT or COPY—Examples. 272
Example 1 272
Example 2 272
Example 3 273
Example 4 273
Example 5 273
Example 6 274
SUM Control Statement 274
SUM Statement Notes. 275
Adding Summary Fields—Examples 277
Example 1 277
Example 2 277
Example 3 277
Example 4 277

Using Program Control Statements

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

- Indicate whether a sort, merge, or copy is performed.
- Describe the control fields to be used.
- Indicate program exits for transferring control to your own routines.
- Describe DFSORT functions you want to have invoked.
- Describe input and output files.
- Indicate various options you want to use during 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 B. Specification/Override of DFSORT Options” on page 549 for an explanation of when 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 online. See *Panels Guide* for details.

This chapter begins with a summary of DFSORT program control statements and coding rules. A detailed description of each statement follows.

Control Statement Summary

Describing the Primary Task

The only required program control statement in a DFSORT application 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 control fields if you are coding a sort application, or specifies a copy application. Indicates whether you want ascending or descending order for the sort.

MERGE
Describes control fields if you are coding a merge application, or specifies a copy application. Indicates whether you want ascending or descending order for the merge.

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

Including or Omitting Records

You can specify whether certain records are included in the output data sets or omitted from them.

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

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

OUTFIL
Specifies the records to be included or omitted in multiple output data sets.

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.

OUTFIL
Specifies how records are reformatted in multiple output data sets.

Producing Multiple Output and Reports and Converting Records

You can produce multiple output data sets and reports and convert variable-length records to fixed-length records.

OUTFIL
Specifies the output data sets and which records are to appear in each. Specifies the reports to be produced. Specifies how records are to be converted from variable-length to fixed-length.

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
Specifies various diagnostic options.

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

MODS Specifies use of one or more user exit routines in a DFSORT application. See “Chapter 4. Using Your Own User Exit Routines” on page 279 for information about user exit routines.

RECORD
Can be used to supply length and type information.

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

Using Symbols

You can define and use a symbol for any field or constant in the following DFSORT control statements: INCLUDE, INREC, MERGE, OMIT, OUTFIL, OUTREC, SORT and SUM. This makes it easy to create and reuse collections of symbols (that is, mappings) representing information associated with various record layouts. See “Chapter 7. Using Symbols for Fields and Constants” on page 433 for complete details.

General Coding Rules

See “Inserting Comment Statements” on page 84 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 74 for special coding conventions that apply to this DD source.

All other DFSORT control statements have the same general format, shown in Figure 5 on page 82. The illustrated format does *not* apply to control statements you

supply in a parameter list. See “Chapter 5. Invoking DFSORT from a Program” on page 331 for information on the special rules that apply.

Figure 5. 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 EQUALS,FORMAT=CH,FIELDS=(10,30,A)
```

- **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 remark 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 6.

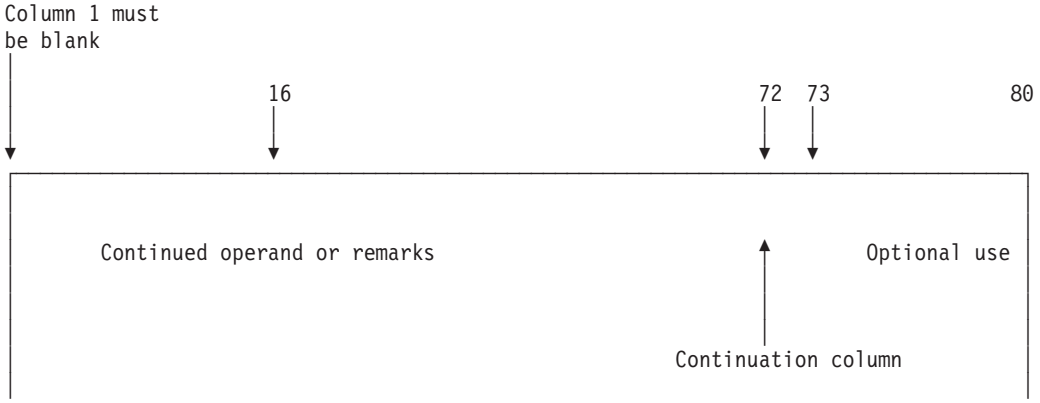


Figure 6. Continuation Line Format

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 logicalextension of the preceding line. Either an operand or a remark field can begin on one line and continue on the next. The following rules apply and are demonstrated in the example.

- If a remark 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.

```
1             16                                                        72
|             |                                                          |
v             v                                                          v
 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')
```

Figure 7. Examples of Valid Continuation Lines:

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.
- In general, values can contain no more than eight alphanumeric characters. Values that specify record counts (such as those for SKIPREC, STOPAFT, and FILSZ) can contain up to 28 digits, the last 15 of which are allowed to be significant (non-zero) digits. Values specified for LOCALE can contain up to 32 alphanumeric 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.
  - It is the only operation definer on a line; column 72 must contain a blank.

Using Control Statements from Other IBM Programs

The INPFIL control statement,which is used by other IBM sort programs, is accepted but not processed. However, control statement errors can result from continuation of an INPFIL statement. The information contained in the INPFIL statement for other IBM sort programs is supplied to DFSORT with DD statements.

Because DFSORT 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.

ALTSEQ Control Statement

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

Changes thecollating 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 field whose format you specify in SORT, MERGE, INCLUDE, or OMIT 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.

CODE

Specifies the original and modified EBCDIC collating positions.

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

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

tt specifies, 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.

Notes:

1. If CHALT is in effect, control fields with format CH are collated using the ALTSEQ table, in addition to those with format AQ.
2. If you use locale processing for SORT, MERGE, INCLUDE, or OMIT fields, you must not use CHALT. If you need alternate sequence processing for a particular field, use format AQ.
3. Using ALTSEQ can degrade performance.

Default: Usually the installation option. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Altering EBCDIC Collating Sequence—Examples

Example 1

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

The character $ (X'5B') is to collate at position X'EA', that is, after uppercase Z (X'E9').

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).

Example 3

```
ALTSEQ CODE=(C1F1,C2F2)
```

The uppercase A (X'C1') is to collate at the **same** position as the numeral 1 (X'F1') and the uppercase B (X'C2') is to collate at the **same** position as the numeral 2 (X'F2').

Note that this ALTSEQ statement does NOT cause collating of A before or after 1, or of B before or after 2.

Example 4

```
ALTSEQ   CODE=(81C1,82C2,83C3,84C4,85C5,86C6,87C7,
         88C8,89C9,91D1,92D2,93D3,94D4,95D5,96D6,
         97D7,98D8,99D9,A2E2,A3E3,A4E4,A5E5,A6E6,
         A7E7,A8E8,A9E9)
```

Each lowercase letter is to collate at the **same** position as the corresponding uppercase letter. For example, the lowercase a (X'81') is to collate at the same position as the uppercase A (X'C1'). This results in case-insensitive collating.

DEBUG Control Statement

```
                  ,
                  ↓
►►──DEBUG──┬──┬─ABEND──┬──┬──────────────────────────────────────────────────────────►◄
           │  └─NOABEND─┘  │
           ├─ABSTP─────────┤
           ├─BSAM──────────┤
           ├──┬─CFW────┬──┤
           │  └─NOCFW──┘  │
           ├─CTRx=n────────┤
           │          ,   │
           │          ↓   │
           ├─EFSDPAFT=(──n──)──┤
           │          ,   │
           │          ↓   │
           ├─EFSDPBFR=(──n──)──┤
           ├─EQUCOUNT──────┤
           ├──┬─ESTAE──┬──┤
           │  └─NOESTAE─┘  │
           └─NOASSIST──────┘
```

The DEBUG control statement is not intended for regular use; only ABEND, NOABEND, and BSAM 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 Messages, Codes and Diagnosis Guide*.

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, DFSORT 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: If DFSORT determines that a SmartBatch pipe data set is being used, it automatically forces the ABEND option on, to ensure that an abend will be generated if an error is detected. This allows for appropriate error propagation by the system to other applications that may be accessing the same SmartBatch pipe data set.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

BSAM

```
►►──BSAM───────────────────────────────────────────────────────────────────►◄
```

Temporarily bypasses the EXCP access method for input and output data sets. BSAM is ignored for VSAM input and output data sets. Note that if Blockset is not selected and BSAM processing is used with concatenated SORTIN input, and both null and non-null data sets are specified, all null data sets must precede all non-null data sets; otherwise, the results are unpredictable.

Note: This option can degrade performance.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

CFW or NOCFW

```
►►──┬─CFW──┬──────────────────────────────────────────────────────────────►◄
    └─NOCFW─┘
```

Temporarily overrides the CFW installation option, which specifies whether DFSORT can use cache fast write when processing SORTWKdd data sets that reside on devices connected to cached 3990 control units.

CFW
Specifies that DFSORT can use cache fast write when processing SORTWKdd data sets.

NOCFW
Specifies that DFSORT cannot use cache fast write.

Note: The NOCFW option can degrade performance.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

CTRx

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

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 Counts the input records being moved from the input buffer (not used for a copy).

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

4 Counts the input records inserted by E15 (not used for Blockset).

5 Counts the output records deleted by E35 (not used for Blockset).

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

EFSDPAFT

```
                  ┌─,──┐
                  ▼    │
►►──EFSDPAFT=(────n────┘)──────────────────────────────────────────────►◄
```

Initiates 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 Takes the SNAP dump after Major Call 2 to the EFS program.

3 Takes the SNAP dump after Major Call 3 to the EFS program.

4 Takes the SNAP dump after Major Call 4 to the EFS program.

5 Takes the SNAP dump after Major Call 5 to the EFS program.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

EFSDPBFR

```
                  ┌─,─────┐
►►──EFSDPBFR=(──────n─────┴──)──────────────────────────────────────────►◄
```

Initiates 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 Takes the SNAP dump before Major Call 2 to the EFS program.

3 Takes the SNAP dump before Major Call 3 to the EFS program.

4 Takes the SNAP dump before Major Call 4 to the EFS program.

5 Takes the SNAP dump before Major Call 5 to the EFS program.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

EQUCOUNT

```
►►──EQUCOUNT──────────────────────────────────────────────────────────►◄
```

Determines the number of records having equal keys (that is, duplicate keys) which have been sorted by the Blockset technique (printed in message ICE184I). For variable-length records, EQUCOUNT can only be used with either Hiperspace (when Hipersorting is used) or work data sets.

Notes:

1. Using EQUCOUNT can degrade performance.
2. ICETOOL’s UNIQUE and OCCUR operators provide unique and non-unique key reporting capabilities that may be more useful for your application than EQUCOUNT.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 application after successful SORTOUT 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.

Note: See “Appendix E. DFSORT Abend Processing” on page 595 for more information on the DFSORT ESTAE recovery routine.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See Appendix B. Specification/Override of DFSORT Options.

NOASSIST

```
►►──NOASSIST──────────────────────────────────────────────────────────►◄
```

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

Note: This option can degrade performance.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Specifying Diagnostic Options—Examples

Example 1

```
SORT FIELDS=(1,4,CH,A)
DEBUG EQUCOUNT
```

If the input records contain the following keys:
KEYA, KEYA, KEYB, KEYB, KEYC, KEYD, KEYD, KEYE

the following message will be issued:
ICE184I THE NUMBER OF RECORDS SORTED WITH EQUAL KEYS IS 3

The three equal keys are KEYA, KEYB, and KEYD.

Note: ICETOOL's UNIQUE and OCCUR operators provide full equal key reporting capabilities and should be used instead of EQUCOUNT.

Example 2

```
SORT FIELDS=(12,2,BI,D)
DEBUG BSAM,ABEND
```

Directs DFSORT to use the BSAM access method for the SORTIN and SORTOUT data sets and to abend if the sort application is unsuccessful.

END Control Statement

```
►►──END──────────────────────────────────────────────────────────►◄
```

The END control statement allows DFSORT to discontinue reading SYSIN, DFSPARM, or SORTCNTL before end of file (EOF).

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.

Discontinue Reading Control Statements—Examples

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 is not read.

Example 2

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

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

INCLUDE Control Statement

```
►►─INCLUDE─COND=─┬─(logical expression)─┬──────────────┬─────────────────────►◄
                 │                     └─FORMAT=f─────┘│
                 ├─ALL─────────────────────────────────┤
                 ├─(ALL)───────────────────────────────┤
                 ├─NONE────────────────────────────────┤
                 └─(NONE)──────────────────────────────┘
```

Use an INCLUDE statement if you want only certain records to appear in the output data set. The INCLUDE statement selects the records you want to include.

You can specify either an INCLUDE statement or an OMIT statement in the same DFSORT run, but not both.

The way in which DFSORT processes short INCLUDE/OMIT compare fields depends on the settings for VLSCMP/NOVLSCMP and VLSHRT/NOVLSHRT. A short field is one where the variable-length record is too short to contain the entire field, that is, the field extends beyond the record. For details about including or omitting short records, see the discussion of the VLSCMP and NOVLSCMP options in “OPTION Control Statement” on page 134.

A logical expression is one or more relational conditions logically combined, based on fields in the input record, and can be represented at a high level as follows:

```
►►─relational condition1──┬──────────────────────────────────────────┬──────►◄
                          │   ┌─.──────────────────────────────────┐ │
                          │   ▼                                    │ │
                          └───┴─,──┬─AND─┬──,relational condition2──┴─┘
                                   └─OR──┘
```

If the logical expression is true for a given record, the record is included in the output data set.

Four types of relational conditions can be used as follows:

1. **Comparisons:**

   Compare two compare fields or a compare field and a decimal, hexadecimal, or character constant.

   For example, you can 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 can compare a field with a specified date, and include only those records with a more recent date.

   See “Comparisons” on page 95 for information about comparisons.

2. **Substring Comparison Tests:**

   Search for a constant within a field value or a field value within a constant.

   For example, you can search the value in a 6-byte field for the character constant C'OK', and include only those records for which C'OK' is found somewhere in the field. Or you can search the character constant C'J69,L92,J82' for the value in a 3-byte field, and include only those records for which C'J69', C'L92', or C'J82' appears in the field.

   See “Substring Comparison Tests” on page 102 for information about substring comparison tests.

3. **Bit Logic Tests:**

   Test the state (on or off) of selected bits in a binary field using a bit or hexadecimal mask or a bit constant.

   For example, you can include only those records which have bits 0 and 2 on in a 1-byte field. Or you can include only those records which have bits 3 and 12 on and bits 6 and 8 off in a 2-byte field.

   See “Bit Logic Tests” on page 104 for information about bit logic tests.

4. **Date Comparisons:**

   Compare a two-digit year date field to a two-digit year date constant or another two-digit year date field, using the century window in effect.

   For example, you can include only those records for which a Z'yymm' date field is between January 1996 and March 2005. Or you can include only those records for which a P'dddyy' field is less than another P'dddyy' field.

   See “Date Comparisons” on page 109 for information about date comparisons.

By nesting relational conditions within parentheses, you can create logical expressions of higher complexity.

Although comparisons, substring comparison tests, bit logic tests, and date comparisons are explained separately below for clarity, they can be combined to form logical expressions.

The INCLUDE control statement differs from the INCLUDE parameter of the OUTFIL statement in the following ways:

- The INCLUDE statement applies to all input records; the INCLUDE parameter applies only to the OUTFIL input records for its OUTFIL group.
- FORMAT=f can be specified with the INCLUDE statement but not with the INCLUDE parameter.
- D2 format can be specified with the INCLUDE statement but not with the INCLUDE parameter.

See “OUTFIL Control Statements” on page 182 for more details on the OUTFIL INCLUDE parameter.

COND

```
►►──COND=──┬─(logical expression)─┬──────────────────────────────────────►◄
           ├─ALL──────────────────┤
           ├─(ALL)────────────────┤
           ├─NONE─────────────────┤
           └─(NONE)───────────────┘
```

logical expression
specifies one or more relational conditions logically combined, based on fields in the input record. If the logical expression is true for a given record, the record is included in the output data sets.

ALL or (ALL)
specifies that all of the input records are to be included in the output data sets.

NONE or (NONE)
specifies that none of the input records are to be included in the output data sets.

Default: ALL. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

FORMAT

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

FORMAT=f can be used only when all the input fields in the entire logical expression have the same format. The permissible field formats for comparisons are shown in Table 7 on page 96. SS (substring) is the only permissible field format for substring comparison tests. BI (unsigned binary) is the only permissible field format for bit logic tests. The Y2x formats are the only permissible field formats for date comparisons.

Default: None. Must be specified if not included in the COND=(logical expression) parameter. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Note: If format values are specified in both FORMAT and COND, DFSORT issues an informational message, uses the format values from COND (f must be specified for each compare field), and does not use the format values from FORMAT.

Relational Condition

The relational condition specifies that a comparison or bit logic test 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.

Comparisons

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, if the FORMAT=f operand is used:

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

Comparison operators are as follows:

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 compare field relative to the beginning of the input record.[4] 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 record descriptor word). All compare fields must start on a byte boundary, and no compare field can extend beyond byte 4092.
- m1 specifies the length of the compare field. Acceptable lengths for different formats are in Table 7.
- f1 specifies the format of the data in the compare field. Permissible formats are given in Table 7.

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

Table 7. Compare Field Formats and Lengths

Format Code	Length	Description
CH	1 to 256 bytes	Character [5]
AQ	1 to 256 bytes	Character with alternate collating sequence
ZD	1 to 256 bytes	Signed zoned decimal
PD	1 to 255 bytes	Signed packed decimal
FI	1 to 256 bytes	Signed fixed-point
BI	1 to 256 bytes	Unsigned binary
AC	1 to 256 bytes	ISCII/ASCII character
CSF or FS	1 to 16 bytes	Signed numeric with optional leading floating sign
CSL or LS	2 to 256 bytes	Signed numeric with leading separate sign
CST or TS	2 to 256 bytes	Signed numeric with trailing separate sign

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

Table 7. Compare Field Formats and Lengths (continued)

Format Code	Length	Description
CLO or OL	1 to 256 bytes	Signed numeric with leading overpunch sign
CTO or OT	1 to 256 bytes	Signed numeric with trailing overpunch sign
ASL	2 to 256 bytes	Signed ISCII/ASCII numeric with leading separate sign
AST	2 to 256 bytes	Signed ISCII/ASCII numeric with trailing separate sign
D2	1 to 256 bytes	User-defined data type (requires an EFS program)

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

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

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

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

Field Format	BI	CH	ZD	PD	FI	AC	ASL	AST	CSF or FS	CSL or LS	CST or TS	CLO or OL	CTO or OT	AQ	D2
BI	X	X													
CH	X	X													
ZD			X	X											
PD			X	X											
FI					X										
AC						X									
ASL							X	X							
AST							X	X							
CSF or FS									X	X	X				
CSL or LS									X	X	X				
CST or TS									X	X	X				
CLO or OL												X	X		
CTO or OT												X	X		
AQ														X	
D2															X

Note: D2 field formats are user-defined.

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

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

Table 9. Permissible Field-to-Constant Comparisons for INCLUDE/OMIT.

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		
CSF or FS	X		
CSL or LS	X		
CST or TS	X		
CLO or OL	X		
CTO or OT	X		
AQ		X	X
D2	X	X	X
Note: 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 compare field of FI format, it cannot be larger than 2147483647 nor smaller than -2147483648.

Examples of valid and invalid decimal constants are:

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

Figure 8. Valid and Invalid Decimal Constants

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	Explanation
C'JDCO'	C''''	Apostrophes not paired
C'$@#'	'ABCDEF'	C identifier missing
C'+0.193'	C'ABCDEF	Apostrophe missing
C'Frank''s'	C'Frank's'	Two single apostrophes needed for one

Figure 9. Valid and Invalid Character String Constants

Double-byte data may be used in a character string for INCLUDE/OMIT comparisons. The start of double-byte data is delimited by the shift-out (SO) control character (X'0E'), and the end by the shift-in (SI) control character (X'0F'). SO and SI control characters are part of the character string and must be paired with zero or an even number of intervening bytes. Nested shift codes are not allowed. All characters between SO and SI must be valid double-byte characters. No single-byte meaning is drawn from the double-byte data.

Examples of valid and invalid character string constants containing double-byte characters are shown below using:

< to represent SO
> to represent SI
Dn to represent a double-byte character

Valid	Invalid	Explanation
C'Q<D1D2>T'	C'Q<R>S'	Single-byte data within SO/SI
C'<D1D2D3>'	C'D1D2D3'	Missing SO/SI; treated as single-byte data
C'Q<D1>R<D2>'	C'Q<D1<D2>>'	Nested SO/SI

Figure 10. Valid and Invalid Strings with Double-Byte Data

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 in the following table.

Valid	Invalid	Explanation
X'ABCD'	X'ABGD'	Invalid hexadecimal digit
X'BF3C'	X'BF3'	Incomplete pair of digits
X'AF050505'	'AF050505'	Missing X identifier
X'BF3C'	'BF3C'X	X identifier in wrong place

Figure 11. Valid and Invalid Hexadecimal Constants

Padding and Truncation

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

Cultural Environment Considerations

DFSORT's collating behavior can be modified according to your cultural environment. The cultural environment is established by selecting the active locale. The active locale's collating rules affect INCLUDE and OMIT processing as follows:

- DFSORT includes or omits records for output according to the collating rules defined in the active locale. This provides inclusion or omission for single- or multi-byte character data, based on defined collating rules which retain the cultural and local characteristics of a language.

If locale processing is to be used, the active locale will only be used to process character (CH) compare fields and character and hexadecimal constants compared to character (CH) compare fields.

For more information on locale processing, see "Cultural Environment Considerations" on page 6 or LOCALE in "OPTION Control Statement" on page 134.

Including Records in the Output Data Set—Comparison Examples

Example 1

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

This example illustrates how to only include records in which:

- The floating sign number in bytes 5 through 12 is greater than the floating sign number in bytes 13 through 20

  OR
- The floating sign number in bytes 105 through 108 is less than or equal to 1000.

Note that all three compare fields have the same format.

Example 2

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

This example illustrates how to only include 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 50 000

  OR
- Bytes 31 through 34 do not contain HERR.

Note that the AND is evaluated before the OR. (“Omitting Records from the Output Data Set—Example” on page 133 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.

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 example illustrates how to only include 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'.

Example 4

```
INCLUDE COND=(7,2,CH,EQ,C'T1',OR,
             (1,2,BI,GE,X'001A',AND,20,2,CH,EQ,25,2,CH))
```

This example shows the effects of VLSCMP/NOVLSCMP and VLSHRT/NOVLSHRT on INCLUDE processing when short records are present.

Consider the records shown in Figure 12 on page 102:

- If VLSCMP is in effect, the first record is included because bytes 7-8 are equal to C’T1’, even though the comparison of bytes 20-21 to 25-26 involves short fields. The second record is included or omitted based on the comparison of bytes 20-21 to bytes 25-26.
- If NOVLSCMP and VLSHRT are in effect, the first record is omitted because the comparison of bytes 20-21 to 25-26 involves short fields. The second record is included or omitted based on the comparison of bytes 20-21 to bytes 25-26.

- If NOVLSCMP and NOVLSHRT are in effect, the first record causes message ICE015A or ICE218A to be issued because the comparison of bytes 20-21 to bytes 25-26 involves short fields.

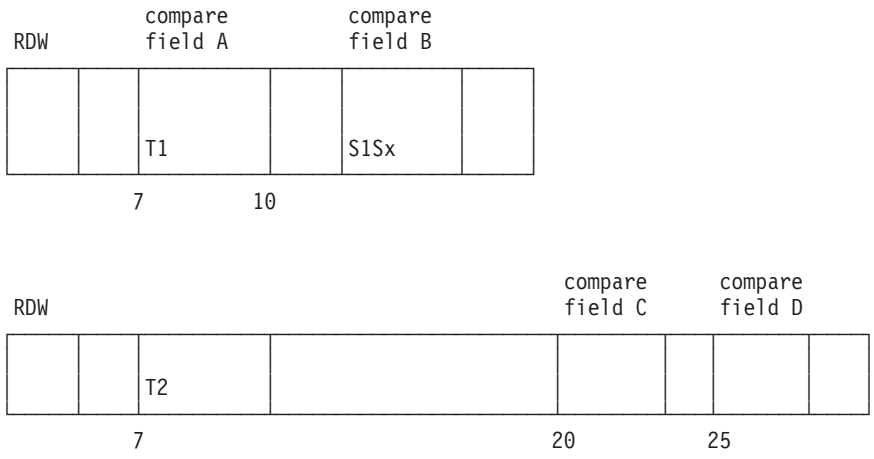


Figure 12. Sample Records

Example 5

```
INCLUDE   COND=(21,2,GE,C'85',OR,21,2,LE,C'03'),FORMAT=CH
```

This example illustrates how to include records based on a two-digit character year field in bytes 21-22. The condition includes records with a value greater than or equal C’85’ in bytes 21-22 or with a value less than or equal to C’03’ in bytes 21-22. This can effectively only include records where the year is between 1985 and 2003.

Substring Comparison Tests

Two types of substring comparison tests are offered, as follows:

1. Find a constant within a field value. For example, you can search the value in a 6-byte field for the character constant C'OK'. If the field value is, for example, C'**OK**' or C'****OK', the relational condition is true; if the field value is C'**ERR*', the relational condition is false.
2. Find a field value within a constant. For example, you can search the character constant C'J69,L92,J82' for the value in a 3-byte field. If the field value is C'J69', C'L92', or C'J82', the relational condition is true; if the field value is C'X24', the relational condition is false. Note that the comma is used within the constant to separate the valid 3-character values; any character that will not appear in the field value can be used as a separator in the constant.

Relational Condition Format

Two formats for the relational condition can be used:

```
►►─(p1,m1,SS,──┬─EQ─┬──,─constant─)──────────────────────────────────►◄
               └─NE─┘
```

Or, if the FORMAT=SS operand is used:

```
►►──(p1,m1,──┬─EQ─┬──,──constant──)──────────────────────────────────►◄
             └─NE─┘
```

Note: FORMAT=SS can precede COND but cannot follow it.

Substring comparison operators are as follows:
EQ Equal to
NE Not equal to

Fields:

p1,m1: These variables specify the character field in the input record for the substring test.

- p1 specifies the first byte of the character input field for the substring test, relative to the beginning of the input record.[6] 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 record descriptor word). All fields to be tested must start on a byte boundary and must not extend beyond byte 4092.
- m1 specifies the length of the field to be tested. The length can be 1 to 256 bytes.

Constant: The constant can be a character string or a hexadecimal string. See “Character String Format” on page 98 and “Hexadecimal String Format” on page 99 for details.

If m1 is greater than the length of the constant, the field value will be searched for the constant and the condition will be true if a match is found when the EQ comparison operator is specified or if a match is not found when the NE comparison operator is specified.

If m1 is smaller than the length of the constant, the constant will be searched for the field value and the condition will be true if a match is found when the EQ comparison operator is specified or if a match is not found when the NE comparison operator is specified.

Including Records in the Output Data Set—Substring Comparison Example

Example

```
INCLUDE   FORMAT=SS,COND=(11,6,EQ,C'OK',OR,21,3,EQ,C'J69,L92,J82')
```

This example illustrates how to include only records in which:

- OK is found somewhere within bytes 11 through 16

  OR
- Bytes 21 through 23 contain J69, L92 or J82.

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

Bit Logic Tests

Two methods for bit logic testing are offered as follows:

- Bit operator with hexadecimal or bit mask
- Bit comparison tests

While any bit logic test can be specified using either of the two methods, each of them offers unique advantages not found with the other.

The ability to specify selected bits in a field, by either of the two methods, can greatly reduce the number of INCLUDE conditions that must be specified to achieve a given result, because the need to account for unspecified bits is eliminated.

Method 1: Bit Operator Tests

This method of bit logic testing allows you to test whether selected bits in a binary field are all on, all off, in a mixed on-off state, or in selected combinations of these states. While this method allows you to test many different possible bit combinations with a single operation, similar to the Test Under Mask (TM) machine instruction, it is less suited to determine if a field contains exactly one particular combination of on and off bits than Method 2 described below.

Relational Condition Format

Two formats for the relational condition can be used:

```
►►──(p1,m1,BI,──┬─ALL────────┬──,──mask──)──────────────────────────────►◄
                │─SOME───────│
                │─NONE───────│
                │─NOTALL─────│
                │─NOTSOME────│
                │─NOTNONE────│
                │─BO─────────│
                │─BM─────────│
                │─BZ─────────│
                │─BNO────────│
                │─BNM────────│
                └─BNZ────────┘
```

Or, if the FORMAT=BI operand is used:

```
►►──(p1,m1,──┬─ALL────────┬──,──mask──)──────────────────────────────►◄
             │─SOME───────│
             │─NONE───────│
             │─NOTALL─────│
             │─NOTSOME────│
             │─NOTNONE────│
             │─BO─────────│
             │─BM─────────│
             │─BZ─────────│
             │─BNO────────│
             │─BNM────────│
             └─BNZ────────┘
```

Bit operators describe the input field to mask relationship to be tested as follows:

ALL or BO

All mask bits are on in the input field

SOME or BM
Some, but not all mask bits are on in the input field
NONE or BZ
No mask bits are on in the input field
NOTALL or BNO
Some or no mask bits are on in the input field
NOTSOME or BNM
All or no mask bits are on in the input field
NOTNONE or BNZ
All or some mask bits are on in the input field

The first set of operators (ALL, SOME, and so on) are intended for those who like meaningful mnemonics. The second set of operators (BO, BM, and so on) are intended for those familiar with the conditions associated with the Test Under Mask (TM) instruction.

Fields

p1,m1: These variables specify the binary field in the input record to be tested against the mask.

- p1 specifies the first byte of the binary input field to be tested against the mask, relative to the beginning of the input record.[7] 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 record descriptor word). All fields to be tested must start on a byte boundary and must not extend beyond byte 4092.
- m1 specifies the length of the field to be tested. The length can be 1 to 256 bytes.

Mask

A hexadecimal string or bit string that indicates the bits in the field selected for testing. If a mask bit is on (1), the corresponding bit in the field is tested. If a mask bit is off (0), the corresponding bit in the field is ignored.

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

X'*yy...yy*'

The value yy represents any pair of hexadecimal digits that constitute a byte (8 bits). Each bit must be 1 (test bit) or 0 (ignore bit). You can specify up to 256 pairs of hexadecimal digits.

Bit String Format: The format for coding a bit string mask is:

B'*bbbbbbbb...bbbbbbbb*'

The value bbbbbbbb represents 8 bits that constitute a byte. Each bit must be 1 (test bit) or 0 (ignore bit). You can specify up to 256 groups of 8 bits. The total number of bits in the mask must be a multiple of 8. A bit mask string can only be used with a bit operator.

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

Padding and Truncation

The hexadecimal or bit mask is truncated or padded on the right to the byte length of the binary field. The padding character is X’00’ (all bits off and thus not tested).

Including Records in the Output Data Set—Bit Operator Test Examples

Example 1

```
INCLUDE   COND=(27,1,CH,EQ,C'D',AND,18,1,BI,ALL,B'10000000')
```

This example illustrates how to only include records in which:

- Byte 27 contains D

  AND
- Byte 18 has bit 0 on.

Example 2

```
INCLUDE   COND=(11,1,BI,BM,X'85')
```

This example illustrates how to only include records in which byte 11 has some, but not all of bits 0, 5 and 7 on. Results for selected field values are shown below:

Table 10. Bit Comparison Example 2: Results for Selected Field Values

11,1,BI Value	11,1,BI Result	Action
X’85’	False	Omit Record
X’C1’	True	Include Record
X’84’	True	Include Record
X’00’	False	Omit Record

Example 3

```
INCLUDE   COND=(11,2,ALL,B'0001001000110100',
               OR,21,1,NONE,B'01001100'),FORMAT=BI
```

This example illustrates how to only include records in which:

- Bytes 11 through 12 have all of bits 3, 6, 10, 11 and 13 on

  OR
- Byte 21 has none of bits 1, 4, or 5 on.

Results for selected field values are shown below:

Table 11. Bit Comparison Example 3: Results for Selected Field Values

11,2,BI Value	11,2,BI Result	21,1,BI Value	21,1,BI Result	Action
X’1234’	True	X’4C’	False	Include Record
X’02C4’	False	X’81’	True	Include Record
X’0204’	False	X’40’	False	Omit Record
X’F334’	True	X’00’	True	Include Record
X’1238’	False	X’4F’	False	Omit Record

Method 2: Bit Comparison Tests

This method of bit logic testing allows you to test whether selected bits in a binary field are either in an exact pattern of on and off bits, or not in that exact pattern. Unlike Method 1 described above, only “equal” and “unequal” comparisons are allowed; however, this method has the advantage of being able to test for a precise combination of on and off bits.

Relational Condition Format

Two formats for the relational condition can be used:

```
►►─(p1,m1,BI,─┬─EQ─┬─,─constant─)──────────────────────────────►◄
              └─NE─┘
```

Or, if the FORMAT=BI operand is used:

```
►►─(p1,m1,─┬─EQ─┬─,─constant─)──────────────────────────────►◄
           └─NE─┘
```

Bit comparison operators are as follows:

EQ Equal to
NE Not equal to

Fields

p1,m1: These variables specify the binary field in the input record to be compared to the bit constant.

- p1 specifies the first byte of the binary input field to be compared to the bit constant, relative to the beginning of the input record.[8] 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 record descriptor word). All fields to be tested must start on a byte boundary and must not extend beyond byte 4092.
- m1 specifies the length of the field to be tested. The length can be 1 to 256 bytes.

Bit Constant

A bit string constant that specifies the pattern to which the binary field is compared. If a bit in the constant is 1 or 0, the corresponding bit in the field is compared to 1 or 0, respectively. If a bit in the constant is . (period), the corresponding bit in the field is ignored.

Bit String Format: The format for coding a bit string constant is:

B'*bbbbbbbb...bbbbbbbb*'

The value bbbbbbbb represents 8 bits that constitute a byte. Each bit must be 1 (test bit for 1), 0 (test bit for 0) or . (ignore bit). You can specify up to 256 groups of 8 bits. The total number of bits in the mask must be a multiple of 8. A bit constant can only be used for bit comparison tests (BI format and EQ or NE operator).

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

Padding and Truncation

The bit constant is truncated or padded on the right to the byte length of the binary field. The padding character is B'00000000' (all bits equal to 0). Note that the padded bytes are compared to the excess bytes in the binary field; you can ensure that this does not cause unwanted results by shortening the field length to eliminate the padding characters, or by increasing the length of the bit constant to specify the exact test pattern you want.

Including Records in the Output Data Set—Bit Comparison Test Examples

Example 1

```
INCLUDE   COND=(27,1,CH,EQ,C'D',AND,18,1,BI,EQ,B'1.......')
```

This example illustrates how to only include records in which:

- Byte 27 contains D

  AND
- Byte 18 is equal to the specified pattern of bit 0 on.

Example 2

```
INCLUDE   COND=(11,1,BI,NE,B'10...1.1')
```

This example illustrates how to only include records in which byte 11 is not equal to the specified pattern of bit 0 on, bit 1 off, bit 5 on and bit 7 on. Results for selected field values are shown below:

Table 12. Bit Comparison Example 2: Results for Selected Field Values

11,1,BI Value	11,1,BI Result	Action
X'85'	False	Omit Record
X'C1'	True	Include Record
X'84'	True	Include Record
X'97'	False	Omit Record

Example 3

```
INCLUDE   COND=(11,2,EQ,B'..01....0......1',
               OR,21,1,EQ,B'01......'),FORMAT=BI
```

This example illustrates how to only include records in which:

- Bytes 11 through 12 are equal to the specified pattern of bit 2 off, bit 3 on, bit 8 off and bit 15 on

  OR
- Byte 21 is equal to the specified pattern of bit 0 off and bit 1 on.

Results for selected field values are shown below:

Table 13. Bit Comparison Example 3: Results for Selected Field Values

11,2,BI Value	11,2,BI Result	21,1,BI Value	21,1,BI Result	Action
X'1221'	True	X'C0'	False	Include Record
X'02C4'	False	X'41'	True	Include Record
X'1234'	False	X'00'	False	Omit Record
X'5F7F'	True	X'7F'	True	Include Record
X'FFFF'	False	X'2F'	False	Omit Record

Date Comparisons

You can use DFSORT's Y2 formats in conjunction with the century window in effect, as follows:

- Use the full date formats (Y2T, Y2U, Y2V, Y2W, Y2X and Y2Y) to compare a two-digit year date field to a two-digit year date constant (Y constant) or to another two-digit year date field.
- Use the year formats (Y2C, Y2Z, Y2S, Y2P, Y2D and Y2B) to compare a two-digit year field to a two-digit year constant (Y constant) or to another two-digit year field.

For example, you can include only those records for which a Z'yymm' date field is between January 1996 and March 2005. Or you can include only those records for which a P'dddyy' field is less than another P'dddyy' field.

The ordering of dates and special indicators used for comparisons with Y2 fields and Y constants is the same as the ascending orders for sorting and merging Y2 fields (see "SORT Control Statement" on page 264 for details).

Relational Condition Format

Two formats for the relational condition can be used:

```
►►─(p1,m1,Y2x,──┬─EQ─┬──,──┬─p2,m2,Y2x─┬──)──────────────────────────────►◄
                ├─NE─┤     └─constant───┘
                ├─GT─┤
                ├─GE─┤
                ├─LT─┤
                └─LE─┘
```

Or, if the FORMAT=Y2x operand is used:

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

Comparison operators are as follows:

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,Y2x: These variables specify a two-digit year date field in the input record to be compared either to another two-digit year date field in the input record or to a two-digit year date constant.

- p1 specifies the first byte of the date field relative to the beginning of the input record.[9] 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 record descriptor word). All date fields must start on a byte boundary, and no date field can extend beyond byte 4092.
- m1 specifies the length of the date field. “Appendix C. Data Format Descriptions” on page 581 describes the length and format for each type of date field.
- Y2x specifies the Y2 format. “Appendix C. Data Format Descriptions” on page 581 describes the length (m) and format (Y2x) for each type of date field.

  If the same Y2 format is used for all date fields, Y2x can be omitted, in which case you must use the FORMAT=Y2x operand.

p2,m2,Y2x: These variables specify another two-digit year date field in the input record with which the p1,m1,Y2x field will be compared.

Constant: A two-digit year date constant in the form Y'string' with which the p1,m1,Y2x field will be compared.

Comparisons: A date field can be compared to a date constant or another date field with the same number of non-year (x) digits. Table 14 shows the type of field-to-field and field-to-constant comparisons you can use. The fields shown for any type of date (for example, yyx and xyy) can be compared to any other fields shown for that type of date or to the Y constant shown for that type of date.

Table 14. Permissible Comparisons for Dates

Type of Date	Fields (m,f)		Y Constant
yyx and xyy	3,Y2T 3,Y2W	2,Y2U 2,Y2X	Y'yyx'
yyxx and xxyy	4,Y2T 4,Y2W	3,Y2V 3,Y2Y	Y'yyxx'
yyxxx and xxxyy	5,Y2T 5,Y2W	3,Y2U 3,Y2X	Y'yyxxx'
yyxxxx and xxxxyy	6,Y2T 6,Y2W	4,Y2V 4,Y2Y	Y'yyxxxx'
yy	2,Y2C 2,Y2S 1,Y2D	2,Y2Z 2,Y2P 1,Y2B	Y'yy'

You must use the same number of digits in a Y constant as the type of date; leading zeros must be specified (for example, for Y'yymm', use Y'0001' for January 2000 and Y'0101' for January 2001).

You can also use Y constants for special indicators as follows:

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

- Y'0...0' (CH/ZD/PD zeros) and Y'9...9' (CH/ZD/PD nines) can be used with Y2T, Y2U, Y2V, Y2W, Y2X and Y2Y dates. You must use the same number of digits as the type of date (for example, Y'000' for yyq or qyy, Y'0000' for yymm or mmyy, and so forth).
- Y'LOW' (BI zeros), Y'BLANKS' (blanks) and Y'HIGH' (BI ones) can be used with Y2T, Y2W and Y2S dates.

Including Records in the Output Data Set—Date Comparisons

Example 1

```
INCLUDE FORMAT=Y2T,
  COND=(3,4,GE,Y'9901',AND,
        3,4,LE,Y'0312',OR,
        3,4,LE,Y'0000')
```

This example illustrates how to only include records in which:

- A C'yymm' date field in bytes 3 through 6 is between January 1999 and December 2003

  OR
- Bytes 3 through 6 contain CH zeros (C'0000'), ZD zeros (Z'0000') or BI zeros (X'00000000').

Note that the century window in effect will be used to interpret the Y'9901' and Y'0312' date constants, as well as real dates in the C'yymm' date field. However, the century window will not be used to interpret the Y'0000' special indicator constant or special indicators in the C'yymm' date field.

Example 2

```
INCLUDE COND=(2,3,Y2X,LT,36,5,Y2T)
```

This example illustrates how to only include records in which a P'dddyy' date field in bytes 2 through 4 is less than a Z'yyddd' date field in bytes 36 through 40.

Note that the century window in effect will be used to interpret real dates in the P'dddyy' and Z'yyddd' date fields. However, the century window will not be used to interpret special indicators in the P'dddyy' and Z'yyddd' date fields.

INCLUDE/OMIT Statement Notes

- Floating point compare 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 user 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.

- If SZERO is in effect, -0 compares as less than +0 when numeric fields and constants are used. If NOSZERO is in effect, -0 compares as equal to +0 when numeric fields and constants are used.

Table 15 on page 112 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, the table in Table 15 helps you get the result that you want.

Table 15. Logic Table for INCLUDE/OMIT.

	Relational Condition	**Program action if next logical operator is:**	
Statement	**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.

INREC Control Statement

```
                  ┌─,──────────────────────────────┐
                  ▼                                │
►►─INREC─FIELDS=─(─┬──────┬─┬─s──────────────────┬─)──────────────►◄
                   └─c:───┘ │ p,m──────────────  │
                            │        └─,a──┘     │
                            │ p──────────────────│
                            │ p,m,HEX────────────│
                            │ p,HEX──────────────│
                            │ p,m,Y2x────────────│
                            │ p,m,Y2x(c)─────────│
                            │ p,m,Y2xP───────────│
                            │ p,m,f──────────────│
                            │        └─,edit─┘   │
                            │ p,m,f,to───────────│
                            │ p,m,Y2x,edit───────│
                            │ p,m,Y2x,to─────────│
                            │ p,m,lookup─────────│
                            └─seqnum─────────────┘
```

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.

The INREC control statement also provides for:

- Insertion of blanks, zeros, strings and sequence numbers before, between, and after the input fields in the reformatted records.
- Sophisticated editing and conversion capabilities, such as hexadecimal display, conversion of numeric values from one format to another, and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.
- Transformation of various forms of two-digit year dates to various forms of four-digit year dates using a specified fixed or sliding century window.
- Selection of a character or hexadecimal string from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).

For information concerning the interaction of INREC and OUTREC, see “INREC Statement Notes” on page 119.

FIELDS

```
                  ┌─,──────────────────────────────────┐
                  ▼                                    │
►►─FIELDS=─(──────┬──────┬──────────┬──────────────────┬─)──────────────────────►◄
                  └─c:───┘          ├─s────────────────┤
                                    ├─p,m──┬────┬──────┤
                                    │      └─,a──┘      │
                                    ├─p──────────────────┤
                                    ├─p,m,HEX────────────┤
                                    ├─p,m,Y2x────────────┤
                                    ├─p,m,Y2x(c)─────────┤
                                    ├─p,m,Y2xP───────────┤
                                    ├─p,m,f──┬────┬──────┤
                                    │        └─,edit─┘   │
                                    ├─p,m,f,to───────────┤
                                    ├─p,m,Y2x,edit───────┤
                                    ├─p,m,Y2x,to─────────┤
                                    ├─p,m,lookup─────────┤
                                    └─seqnum─────────────┘
```

Specifies the order and alignment of the separation fields, unedited and edited input fields and sequence numbers in the reformatted input record.

c: Specifies the position (column) for a separation field, input field or sequence number, 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 32752.
- 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.

Both valid and invalid examples are shown in Table 16.

Table 16. Examples of Valid and Invalid Column Alignment

Validity	Specified	Result
Valid	33:C'State '	Columns 1-32 — blank Columns 33-38 — '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	32753:21,8	Invalid — column value must be less than 32753.
Invalid	5:10:2,5	Column values cannot be adjacent.
Invalid	20,10,6:C'AB'	Column value overlaps previous field.

s Specifies that a separation field is to appear in the reformatted input record. 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, nZ, nC'xx...x', and nX'yy...yy'.

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

Examples of valid and invalid blank separation are shown in Table 17.

Table 17. 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 to appear in the reformatted input records. n can range from 1 to 4095. If n is omitted, 1 is used.

Examples of valid and invalid binary zero separation are shown in Table 18 on page 115.

Table 18. 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 (4000Z,450Z 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 to appear in the reformatted input records. n can range 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'
```

Examples of valid and invalid character string separation are shown in Table 19.

Table 19. 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
Valid	C'Frank''s'	Frank's	7
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
Invalid	C'Frank's'	Two single apostrophes needed for one	n/a

nX'yy...yy'

Hexadecimal string separation. n repetitions of the hexadecimal string constant (X'yy...yy') are to appear in the reformatted input records. n can range 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. Examples of valid and invalid hexadecimal string separation are shown in Table 20 on page 116.

Table 20. 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

Specifies that an unedited input field is to appear in the reformatted input record.

p Specifies the first byte of the input field relative to the beginning of the input record.[10] 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 32752. For special rules concerning variable-length records, see “INREC Statement Notes” on page 119.

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 “INREC Statement Notes” on page 119 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.

Permissible values of **a** are:

H Halfword aligned. 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.

10. If your E15 user exit reformats the record, p must refer to the record as reformatted by the exit.

p specifies that the unedited part of the input record (that part beyond the minimum record length) is to appear in the reformatted input record as the last field. Note that if the reformatted input record includes only the RDW and the variable part of the input record, “null” records containing only an RDW may result.

A value must be specified for p that is less than or equal to the minimum record length (RECORD statement L4 value) plus 1 byte.

p,m,HEX
specifies that the hexadecimal representation of an input field is to appear in the reformatted input record.

See p,m,HEX under OUTFIL OUTREC for details.

p,HEX
specifies that the hexadecimal representation of the variable part of the input record (that part beyond the minimum record length) is to appear in the reformatted input record as the last field. Note that if the reformatted input record includes only the RDW and the variable part of the input record, ″null″ records containing only an RDW may result.

See p,HEX under OUTFIL OUTREC for details.

Sample Syntax:

```
 Fixed input records
    INREC FIELDS=(1:5,10,15:8C'0',25:20,15,80
  Variable input records
    INREC FIELDS=(1,4,C' RDW=',1,4,HEX,C' FIXED=',
                  5,20,HEX,C' VARIABLE=',21,HEX)
```

p,m,Y2x
specifies that the four-digit CH date representation of a two-digit year input date field is to appear in the reformatted input record. Real dates are transformed using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2T transforms C’000000’ to C’00000000’).

See p,m,Y2x under OUTFIL OUTREC for details.

Sample Syntax:

```
INREC FIELDS=(21,3,Y2V,X,12,5,Y2W)
```

p,m,Y2x(c)
specifies that the four-digit CH date representation with separators of a two-digit year input date field is to appear in the reformatted input record. Real dates are transformed using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2T(/) transforms C’000000’ to C’0000/00/00’).

See p,m,Y2x(c) under OUTFIL OUTREC for details.

Sample Syntax:

```
INREC FIELDS=(25,6,Y2T(-),X,14,2,Y2U(/))
```

p,m,Y2xP
specifies that the four-digit PD date representation of a two-digit year input date field is to appear in the reformatted input record. Real dates are transformed

using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2TP transforms C'000000' to P'00000000').

See p,m,Y2xP under OUTFIL OUTREC for details.

Sample Syntax:

```
INREC FIELDS=(11,3,Y2XP,X,21,4,Y2WP)
```

p,m,f,edit
specifies that an edited numeric input field is to appear in the reformatted input record. You can edit BI, FI, PD, PD0, ZD, and CSF/FS fields using either pre-defined edit masks (M0-M25) or specific edit patterns you define. You can control the way the edited fields look with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on.

See p,m,f,edit under OUTFIL OUTREC for details.

Sample Syntax:

```
INREC FIELDS=(5:21,8,ZD,M19,X,46,5,ZD,M13,
              31:35,6,FS,SIGNS=(,,+,-),LENGTH=10,
              51:8,4,PD,EDIT=(**II,IIT.TTXS),SIGNS=(,,+,-))
```

p,m,f,to
specifies that a converted numeric input field is to appear in the reformatted input record. You can convert BI, FI, PD, PD0, ZD or CSF/FS fields to BI, FI, PD, ZD or CSF/FS fields.

See p,m,f,to under OUTFIL OUTREC for details.

Sample Syntax:

```
INREC FIELDS=(21,5,ZD,TO=PD,X,8,4,ZD,FI,LENGTH=2)
```

p,m,Y2x,edit
specifies that an edited four-digit year CH date representation of a two-digit year input date field is to appear in the reformatted input record.

See p,m,Y2x,edit under OUTFIL OUTREC for details.

p,m,Y2x,to
specifies that a converted four-digit year date representation of a two-digit year input date field is to appear in the reformatted input record.

See p,m,Y2x,to under OUTFIL OUTREC for details.

p,m,lookup
specifies that a character or hexadecimal string from a lookup table is to appear in the reformatted input record. You can use p,m,lookup to select a specified character or hexadecimal string based on matching an input value against character, hexadecimal, or bit constants.

See p,m,lookup under OUTFIL OUTREC for details.

Sample Syntax:

```
INREC FIELDS=(11,1,
              CHANGE=(6,
                C'R',C'READ',
                C'U',C'UPDATE',
```

```
                                X'FF',C'EMPTY',
                                C'A',C'ALTER'),
                              NOMATCH=(11,6),
                              4X,
                              21,1,
                              CHANGE=(10,
                                B'.1......',C'VSAM',
                                B'.0......',C'NON-VSAM'))
```

seqnum
specifies that a sequence number is to appear in the reformatted input record. The sequence numbers are assigned in the order in which the records are received for INREC processing. You can create BI, PD, ZD or CSF/FS sequence numbers and control their lengths, starting values and increment values.

See seqnum under OUTFIL OUTREC for details.

Sample Syntax:

```
INREC FIELDS=(SEQNUM,6,ZD,START=1000,INCR=50,1,60)
```

Default: None; must be specified. See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

INREC Statement Notes

- When INREC is specified, DFSORT reformats the input records after user exit E15 or INCLUDE/OMIT statement processing is finished. Thus, references to fields by your E15 user exit and INCLUDE/OMIT statements are not affected, whereas your SORT, OUTREC, and SUM statements must refer to fields in the reformatted input records. Your E35 user exit must refer to fields in the reformatted output record.
- In general, OUTREC should be used rather than INREC so your SORT and SUM statements can refer to fields in the original input records.
- If you use locale processing for SORT, MERGE, INCLUDE, or OMIT fields, you must not use INREC. Use the OUTREC statement or the OUTREC operand of the OUTFIL statement instead of INREC.
- When you specify INREC, you must be aware of the change in record size and layout of the resulting reformatted input records.
- Performance can be improved if you can significantly reduce the length of your records with INREC. INREC and OUTREC should not be used unless they are actually needed to reformat your records.
- 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.
- If the SORTOUT LRECL is specified or available, DFSORT will use it even if it does not match the reformatted INREC record length; this can cause padding or truncation of the reformatted INREC records, or termination. If the SORTOUT LRECL is not specified or available, DFSORT can automatically use the

reformatted INREC record length as the SORTOUT LRECL, when appropriate. See the discussion of the SOLRF and NOSOLRF options in “OPTION Control Statement” on page 134.

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 14 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, asspecified 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 control fields or both.
- If input is variable records, the output is also variable. This means that each record is given the correct RDW by DFSORT before output.
- 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.
- If SZERO is in effect, -0 is treated as negative and +0 is treated as positive for edited or converted input fields. If NOSZERO is in effect, -0 and +0 are treated as positive for edited or converted input fields.

Reformatting Records Before Processing — Examples

Example 1

INREC Method:

```
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 Method:

```
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. Unnecessary fields are eliminated from the output records using INREC or OUTREC. 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. SOLRF (the IBM-supplied default) is in effect, so unless the SORTOUT LRECL is specified or available, it will automatically be set to the reformatted record length of 23. The reformatted records look as follows after INREC or OUTREC processing:

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 OUTREC makes it easier to code the SORT and SUM statements. 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.

Example 2

```
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 expands 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.

Example 3

```
INREC FIELDS=(20,4,12,3)
SORT FIELDS=(1,4,D,5,3,D),FORMAT=CH
OUTREC FIELDS=(5X,1,4,H,19:1,2,5,3,80X'FF')
```

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.

The reformatted input records are fixed-length, and have a record size of seven 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. SOLRF (the IBM-supplied default) is in effect, so unless the SORTOUT LRECL is specified or available, it will automatically be set to the reformatted record length of 103.. The reformatted output records 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	Hexadecimal FF's

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.

Example 4

```
INREC FIELDS=(8100,16,1,8099,8116,200)
SORT FIELDS=(1,16,CH,A)
OUTREC FIELDS=(17,8099,1,16,8116,200)
```

This example illustrates how you can sort on a field beyond DFSORT's normal limit of byte 4092 by using INREC and OUTREC.

The "sort" field is at input positions 8100 through 8115. The INREC statement is used to reformat the input records so that the "sort" field is within the first 4092 bytes. The reformatted input records look as follows:

Position	Contents
1-16	Input positions 8100 through 8115

17-8115	Input positions 1 through 8099
8116-8315	Input positions 8116 through 8315

The SORT statement can now refer to the “sort” field in the reformatted input records. The OUTREC statement is used to restore the records to their original format.

Example 5

```
OPTION COPY,Y2PAST=1985
INREC FIELDS=(SEQNUM,4,BI,
              8,5,ZD,TO=PD,
              31,2,PD,TO=FI,LENGTH=2,
              15,6,Y2TP,
              25,3,CHANGE=(1,C'L92',X'01',C'M72',X'02',C'J42',X'03'),
              NOMATCH=(X'FF'))
```

This example illustrates how a sequence number can be generated, how values in one numeric or date format can be converted to another format, and how a lookup table can be used.

The reformatted input records will look as follows:

Position	Contents
1-4	A binary sequence number that starts at 1 and increments by 1.
5–7	A PD field containing the converted ZD field from input positions 8 through 12.
8–9	An FI field containing the converted PD field from input positions 31 through 32.
10–14	A P’yyyymmdd’ date field containing the C’yymmdd’ date field from input positions 15-20 transformed according to the specified century window of 1985-2084.
15	A BI field containing X’01’, X’02’, X’03’ or X’FF’ as determined by using a lookup table for the input field in positions 25-27.

The SORT statement can now refer to the “sort” field in the reformatted input records. The OUTREC statement is used to restore the records to their original format.

MERGE Control Statement

The MERGE control statement must be used when a merge operation is to be

```
                                  ┌──,──────────┐
                                  ↓             │
►►─MERGE─FIELDS=──┬─(──────p,m,f,s──────────)──────────────────────────────►
                  │                                                        │
                  │                  ┌──,──────────┐                       │
                  │                  ↓             │                       │
                  ├─(──────p,m,s──────────)──,─FORMAT=f──────────────────┤
                  │  ┌─COPY──────────┐                                     │
                  │  └─(COPY)────────┘                                     │
```

```
►──┬──────────────────────────────────────────────────────────────────────►◄
   │                  ┌──,──────────────────────┐
   │                  ↓                         │
   └──,──┬─┬─EQUALS──────┬─┬──────────────────┬──┘
         │ └─NOEQUALS────┘ │                  │
         ├─FILES=n─────────┤                  │
         ├─┬─FILSZ=x──────┬─┤                  │
         │ └─SIZE=y───────┘ │                  │
         ├─SKIPREC=z───────┤                  │
         ├─STOPAFT=n───────┤                  │
         └─Y2PAST=──┬─s──┬─┘                  │
                    └─f──┘                    │
```

performed; this statement describes the control fields in the input records on which the input data sets have previously been sorted.

A MERGE statement can also be used to specify a copy application. User labels will not be copied to the output data sets.

You can merge up to 100 data sets with Blockset merge or up to 16 data sets with Conventional merge. If Blockset merge is not selected, you can use a SORTDIAG DD statement to force message ICE800I, which gives a code indicating why Blockset could not be used.

The way in which DFSORT processes short MERGE control fields depends on the setting for VLSHRT/NOVLSHRT. A short field is one where the variable-length record is too short to contain the entire field, that is, the field extends beyond the record. For details about merging short records, see the discussion of the VLSHRT and NOVLSHRT options in “OPTION Control Statement” on page 134.

The options available on the MERGE statement can be specified in other sources as well. A table showing all possible sources for these options and the order of override are given in “Appendix B. Specification/Override of DFSORT Options” on page 549. When an option can be specified on either the MERGE or OPTION statement, it is preferable to specify it on the OPTION statement.

DFSORT accepts but does not process the following MERGE operands: WORK=value and ORDER=value.

DFSORT’s collating behavior can be modified according to your cultural environment. The cultural environment is established by selecting the active locale. The active locale’s collating rules affect MERGE processing as follows:

- DFSORT produces merged records for output according to the collating rules defined in the active locale. This provides merging for single- or multi-byte character data, based on defined collating rules that retain the cultural and local characteristics of a language.

If locale processing is to be used, the active locale will only be used to process character (CH) control fields.

For more information on locale processing, see “Cultural Environment Considerations” on page 6 or LOCALE in “OPTION Control Statement” on page 134.

Note: For a merge application, records deleted during an E35 exit routine are not sequence checked. If you use an E35 exit routine without an output data set, sequence checking is not performed at the time the records are passed to the E35 user exit; therefore, you must ensure that input records are in correct sequence.

FIELDS

```
                  ┌─,──────────┐
►►─FIELDS=─(──────┴─p,m,f,s────┴─)──────────────────────────────────────────►◄
```

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 264.

FIELDS=COPY or FIELDS=(COPY)

```
►►─FIELDS=──┬─COPY────┬──────────────────────────────────────────────────►◄
            └─(COPY)──┘
```

See the discussion of the COPY option on the OPTION statement, in “OPTION Control Statement” on page 134.

FORMAT=f

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

See the discussion of the FORMAT option in “SORT Control Statement” on page 264. Used the same way for a merge as for a sort.

EQUALS or NOEQUALS

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

See the discussion of these options on the OPTION statement, in “OPTION Control Statement” on page 134.

FILES=n

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

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

Default: None; must be specified when an E32 exit is used.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

FILSZ or SIZE

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

See the discussion of these options on the OPTION statement, in “OPTION Control Statement” on page 134.

SKIPREC

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

See the discussion of this option on the OPTION statement, in “OPTION Control Statement” on page 134.

STOPAFT

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

See the discussion of this option on the OPTION statement, in “OPTION Control Statement” on page 134.

Y2PAST

```
►►──Y2PAST=──┬─s─┬──────────────────────────────────────────────────►◄
             └─f─┘
```

See the discussion of this option on the OPTION statement, in “OPTION Control Statement” on page 134.

Note: CENTURY=value and CENTWIN=value can be used instead of Y2PAST=value.

Specifying a MERGE or COPY—Examples

Example 1

```
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 in ascending order.

FILSZ
The input data sets contain exactly 29483 records.

Example 2

```
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.

Example 3

```
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.

Example 4

```
MERGE FIELDS=COPY
```

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

MODS Control Statement

```
                  ┌──,──────────────────────────────────────┐
                  ▼                                         │
►►──MODS──────exit=──(──n,m──┬──────────────────────────┬──)──┬──────────────────────────┬──►◄
                             └──,──┬──────────────────┬──┘     └──HILEVEL=YES──┘
                                   └──s──┬──────────┬──┘
                                         └──,e──┘
```

The MODS statement is needed only when DFSORTpasses control to your routines at user exits. The MODS statement associates user routines with specific DFSORT exits and provides DFSORT with descriptions of these routines. For details about DFSORT user exits and how user routines can be used, see “Chapter 4. Using Your Own User Exit Routines” on page 279.

To use one of the user 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 user 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 the exit parameter describe the user routine. These values are:

n specifies 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 specifies 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 specifies 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. SYSIN is not valid for copy processing.

If a value is not specified 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 specifies the linkage editor requirements of your routine or indicates that your routine is written in COBOL. The following values are allowed:

N specifies 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' or by the HILEVEL=YES parameter.

C specifies 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. The COBEXIT option of the OPTION statement specifies the library for the COBOL exits.

T specifies that your routine must be link-edited together with other routines to be used in the same phase (for example, E1n routines) of DFSORT. See “Dynamically Link-Editing User Exit Routines” on page 289 for additional information. This value is not valid for copy processing.

S specifies 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. See “Dynamically Link-Editing User Exit Routines” on page 289 for additional information. This value is not valid for copy processing.

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

HILEVEL=YES

```
►►──HILEVEL=YES──────────────────────────────────────────────────────────────►◄
```

specifies that:

- if an E15 routine is identified on the MODS statement, it is written in COBOL
- if an E35 routine is identified on the MODS statement, it is written in COBOL.

If you identify an E15 routine and an E35 routine on the MODS statement, specify HILEVEL=YES only if both routines are written in COBOL. If you do not identify an E15 or E35 routine on the MODS statement, HILEVEL=YES is ignored.

Note: COBOL=YES can be used instead of HILEVEL=YES.

Notes:

1. 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.
2. 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.
3. If the same routine is used in both input (that is, E1n routines) and output (that is, E3n routines) DFSORT program phases, a separate copy of the routine must be provided for each exit.
4. HILEVEL=YES can be used instead of C as the fourth parameter, to indicate that an E15 or E35 routine is written in COBOL. In this case, if T is specified as the fourth parameter for E15 or E35, DFSORT terminates. If you identify an E15 routine and an E35 routine on the MODS statement, specify HILEVEL=YES only if both routines are written in COBOL.
5. EXEC PARM parameter E15=COB can be used instead of C as the fourth parameter, to indicate that an E15 is written in COBOL. In this case, if T is specified as the fourth parameter for E15, DFSORT terminates.
6. EXEC PARM parameter E35=COB can be used instead of C as the fourth parameter, to indicate that an E35 is written in COBOL. In this case, if T is specified as the fourth parameter for E35, DFSORT terminates.
7. If HILEVEL=YES, E15=COB, or E35=COB is used instead of C as the fourth parameter, to indicate that an exit is written in COBOL, the fourth parameter for that exit must be specified as N or not specified.
8. If a COBOL E15 or E35 is specified for a conventional merge or tape work data set sort, DFSORT terminates.
9. exit=(n,m) can be used to omit both the s and e parameters.
10. exit=(n,m,,e) can be used to omit the s parameter, but not the parameter.
11. 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.

Default: None; must be specified if you use exit routines. N is the default for the fourth parameter.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

For information on user exit routines in SYSIN, see “System DD Statements” on page 62.

For details on how to design your routines, refer to “Summary of Rules for User Exit Routines” on page 287.

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.

Identifying User Exit Routines—Examples

Example 1

```
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.

Example 2

```
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

```
►►──OMIT──COND=──┬─(logical expression)─┬──────────────┬──────────────────────────►◄
                 │                      └─FORMAT=f─────┘                          │
                 ├─ALL──────────────────────────────────────────────────────────┤
                 ├─(ALL)────────────────────────────────────────────────────────┤
                 ├─NONE─────────────────────────────────────────────────────────┤
                 └─(NONE)───────────────────────────────────────────────────────┘
```

Use an OMIT statement if you do not want all of the input records to appear in the output data sets. The OMIT statement selects the records you do *not* want to include.

You can specify either an INCLUDE statement or an OMIT statement in the same DFSORT run, but not both.

The way in which DFSORT processes short INCLUDE/OMIT compare fields depends on the settings for VLSCMP/NOVLSCMP and VLSHRT/NOVLSHRT. A short field is one where the variable-length record is too short to contain the entire field, that is, the field extends beyond the record. For details about including or omitting short records, see the discussion of the VLSCMP and NOVLSCMP options in “OPTION Control Statement” on page 134.

A logical expression is one or more relational conditions logically combined, based on fields in the input record, and can be represented at a high level as follows:

```
►►──relational condition1──┬──────────────────────────────────────┬──►◄
                           │   ┌─.──────────────────────────────┐ │
                           └───┴──,──┬─AND─┬──,relational condition2──┴─┘
                                     └─OR──┘
```

If the logical expression is true for a given record, the record is omitted from the output data set.

Four types of relational conditions can be used as follows:

1. **Comparisons:**

   Compare two compare fields or a compare field and a decimal, hexadecimal, or character constant.

   For example, you can compare the first 6 bytes of each record with its last 6 bytes, and omit those records in which those fields are identical. Or you can compare a field with a specified date, and omit those records with a more recent date.

2. **Substring Comparison Tests:**

   Search for a constant within a field value or a field value within a constant.

   For example, you can search the value in a 6-byte field for the character constant C'OK', and omit those records for which C'OK' is found somewhere in the field. Or you can search the character constant C'J69,L92,J82' for the value in a 3-byte field, and omit those records for which C'J69', C'L92', or C'J82' appears in the field.

3. **Bit Logic Tests:**

   Test the state (on or off) of selected bits in a binary field using a bit or hexadecimal mask or a bit constant.

   For example, you can omit those records which have bits 0 and 2 on in a 1-byte field. Or you can omit those records which have bits 3 and 12 on and bits 6 and 8 off in a 2-byte field.

4. **Date Comparisons:**

   Compare a two-digit year date field to a two-digit year date constant or another two-digit year date field, using the century window in effect.

   For example, you can omit only those records for which a Z'yymm' date field is between January 1996 and March 2005. Or you can omit only those records for which a P'dddyy' field is less than another P'dddyy' field.

For complete details on the parameters of the OMIT control statement, see “INCLUDE Control Statement” on page 93.

OMIT Control Statement

The OMIT control statement differs from the OMIT parameter of the OUTFIL statement in the following ways:

- The OMIT statement applies to all input records; the OMIT parameter applies only to the OUTFIL input records for its OUTFIL group.
- FORMAT=f can be specified with the OMIT statement but not with the OMIT parameter.
- D2 format can be specified with the OMIT statement but not with the OMIT parameter.

See “OUTFIL Control Statements” on page 182 for more details on the OUTFIL OMIT parameter.

COND

```
►►──COND=──┬─(logical expression)─┬──────────────────────────────────────►◄
           ├─ALL──────────────────┤
           ├─(ALL)────────────────┤
           ├─NONE─────────────────┤
           └─(NONE)───────────────┘
```

logical expression
specifies one or more relational conditions logically combined, based on fields in the input record. If the logical expression is true for a given record, the record is omitted from the output data sets.

ALL or (ALL)
specifies that all of the input records are to be omitted from the output data sets.

NONE or (NONE)
specifies that none of the input records are to be omitted from the output data sets.

Default: NONE. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

FORMAT

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

FORMAT=f can be used only when all the input fields in the entire logical expression have the same format. The permissible field formats for comparisons are shown in Table 7 on page 96. SS (substring) is the only permissible field format for substring comparison tests. BI (unsigned binary) is the only permissible field format for bit logic tests. The Y2x formats are the only permissible field formats for date comparisons.

Default: None. Must be specified if not included in the COND=(logical expression) parameter. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Note: If format values are specified in both FORMAT and COND, DFSORT issues an informational message, uses the format values from COND (f must be specified for each compare field), and does not use the format values from FORMAT.

Omitting Records from the Output Data Set—Example

Example

```
OMIT COND=(27,1,CH,EQ,C'D',&,
           (22,2,BI,SOME,X'C008',|,
            28,1,BI,EQ,B'.1....01'))
```

This statement omits records in which:

- Byte 27 contains D

  AND
- Bytes 22 through 23 have some, but not all of bits 0, 1 and 12 on OR byte 28 is equal to the specified pattern of bit 1 on, bit 6 off and bit 7 on.

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.

For additional examples of logical expressions, see “INCLUDE Control Statement” on page 93.

OPTION Control Statement

```
                  ,
                  ┌──────────────────────────────────────────────────────┐
►►──OPTION──┬──┬──┬──ARESALL=──┬──n──┬──┬──────────────────────────────┴──►◄
            │  │  │            ├──nK─┤  │
            │  │  │            └──nM─┘  │
            │  │  ├──ARESINV=──┬──n──┬──┤
            │  │  │            ├──nK─┤  │
            │  │  │            └──nM─┘  │
            │  │  ├──AVGRLEN=n──────────┤
            │  │  ├──┬──CHALT──┬────────┤
            │  │  │  └──NOCHALT─┘        │
            │  │  ├──┬──CHECK──┬────────┤
            │  │  │  └──NOCHECK─┘        │
            │  │  ├──┬──CINV───┬────────┤
            │  │  │  └──NOCINV──┘        │
            │  │  ├──CKPT───────────────┤
            │  │  ├──COBEXIT=──┬──COB1─┬──────────────────────────────┤
            │  │  │            └──COB2─┘                              │
            │  │  ├──COPY───────────────┤
            │  │  ├──DSPSIZE=──┬──MAX──┬──────────────────────────────┤
            │  │  │            └──n────┘                              │
            │  │  ├──DYNALLOC──┬──────────────────────────────────────┤
            │  │  │            └──=──┬──d──────────┬──────────────────┘
            │  │  │                  ├──(d)────────┤
            │  │  │                  ├──(,n)───────┤
            │  │  │                  ├──(d,n)──────┤
            │  │  │                  ├──OFF────────┤
            │  │  │                  └──(OFF)──────┘
            │  │  ├──EFS=──┬──name─┬──────────────┤
            │  │  │        └──NONE─┘              │
            │  │  ├──┬──EQUALS──┬──────────────────┤
            │  │  │  └──NOEQUALS─┘                  │
            │  │  ├──EXITCK=──┬──STRONG─┬──────────┤
            │  │  │            └──WEAK───┘          │
            │  │  ├──┬──FILSZ=──┬──x──┬──┬──────────┤
            │  │  │  │          ├──Ex─┤  │          │
            │  │  │  │          └──Ux─┘  │          │
            │  │  │  └──SIZE=──┬──y──┬──┘          │
            │  │  │             ├──Ey─┤             │
            │  │  │             └──Uy─┘             │
            │  │  ├──HIPRMAX=──┬──OPTIMAL─┬──────────┤
            │  │  │            ├──n───────┤          │
            │  │  │            └──p%──────┘          │
            │  │  ├──┬──LIST───┬──────────────────────┤
            │  │  │  └──NOLIST─┘                      │
            │  │  ├──┬──LISTX──┬──────────────────────┤
            │  │  │  └──NOLISTX─┘                      │
            │  │  └──LOCALE=──┬──name────┬────────────┤
            │  │               ├──CURRENT─┤            │
            │  │               └──NONE────┘            │
```

Figure 13. Syntax Diagram for the Option Control Statement (Part 1 of 3)

```
├─┬─MAINSIZE=──┬─n──┬──────────────────────────────────────────────┤
  │            ├─nK─┤                                              │
  │            ├─nM─┤                                              │
  │            └─MAX┘                                              │
  ├─MSGDDN=ddname──────────────────────────────────────────────────┤
  ├─MSGPRT=──┬─ALL──────┬──────────────────────────────────────────┤
  │          ├─NONE─────┤                                          │
  │          └─CRITICAL─┘                                          │
  ├─NOBLKSET───────────────────────────────────────────────────────┤
  ├─NOOUTREL───────────────────────────────────────────────────────┤
  ├─NOOUTSEC───────────────────────────────────────────────────────┤
  ├─NOSTIMER───────────────────────────────────────────────────────┤
  ├─ODMAXBF=──┬─n──┬──────────────────────────────────────────────┤
  │           ├─nK─┤                                               │
  │           └─nM─┘                                               │
  ├─OVFLO──┬─RC0──┬──────────────────────────────────────────────┤
  │        ├─RC4──┤                                               │
  │        └─RC16─┘                                               │
  ├─PAD──┬─RC0──┬──────────────────────────────────────────────┤
  │      ├─RC4──┤                                                 │
  │      └─RC16─┘                                                 │
  ├─RESALL=──┬─n──┬──────────────────────────────────────────────┤
  │          ├─nK─┤                                               │
  │          └─nM─┘                                               │
  ├─┬─RESET───┬──────────────────────────────────────────────────┤
  │ └─NORESET─┘                                                   │
  ├─RESINV=──┬─n──┬──────────────────────────────────────────────┤
  │          ├─nK─┤                                               │
  │          └─nM─┘                                               │
  ├─SDB=──┬─LARGE──┬──────────────────────────────────────────────┤
  │       ├─YES────┤                                               │
  │       ├─INPUT──┤                                               │
  │       └─NO─────┘                                               │
  ├─SKIPREC=z──────────────────────────────────────────────────────┤
  ├─SMF=──┬─SHORT──┬──────────────────────────────────────────────┤
  │       ├─FULL───┤                                               │
  │       └─NO─────┘                                               │
  ├─┬─SOLRF───┬──────────────────────────────────────────────────┤
  │ └─NOSOLRF─┘                                                   │
  ├─SORTDD=cccc────────────────────────────────────────────────────┤
  ├─SORTIN=ddname──────────────────────────────────────────────────┤
  ├─SORTOUT=ddname─────────────────────────────────────────────────┤
  ├─SPANINC=──┬─RC0──┬──────────────────────────────────────────┤
  │           ├─RC4──┤                                             │
  │           └─RC16─┘                                             │
  ├─STOPAFT=n──────────────────────────────────────────────────────┤
  ├─┬─SZERO───┬──────────────────────────────────────────────────┤
  │ └─NOSZERO─┘                                                   │
  ├─TRUNC──┬─RC0──┬──────────────────────────────────────────────┤
  │        ├─RC4──┤                                               │
  │        └─RC16─┘                                               │
  └─USEWKDD──────────────────────────────────────────────────────┘
```

Figure 13. Syntax Diagram for the Option Control Statement (Part 2 of 3)

```
├──┬─VERIFY───┬──────────────────────────────────────────────────────────┤
   │ └NOVERIFY┘│
   ├─VLLONG───┤
   │ └NOVLLONG┘│
   ├─VLSCMP───┤
   │ └NOVLSCMP┘│
   ├─VLSHRT───┤
   │ └NOVLSHRT┘│
   ├─VSAMEMT──┤
   │ └NVSAMEMT┘│
   ├─VSAMIO───┤
   │ └NOVSAMIO┘│
   ├─WRKREL───┤
   │ └NOWRKREL┘│
   ├─WRKSEC───┤
   │ └NOWRKSEC┘│
   ├─Y2PAST=──┬─s─┬──┤
   │          └─f─┘  │
   └─ZDPRINT──┬──────┘
     └NZDPRINT┘
```

Figure 13. Syntax Diagram for the Option Control Statement (Part 3 of 3)

Note for Syntax Diagram: The keywords EFS, LIST, NOLIST, LISTX, NOLISTX, MSGDDN, MSGPRT, SMF, SORTDD, SORTIN, SORTOUT, and USEWKDD are used only when they are specified on the OPTION control statement passed by an extended parameter list or when specified 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.

The OPTION control statement allows you to override some of the options available at installation time (such asEQUALS 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 B. Specification/Override of DFSORT Options” on page 549.

Details of aliases for OPTION statement options are given under the description of individual options. Table 23 on page 177 summarizes the available aliases.

DFSORT accepts but does not process the following OPTION operands: ALGQ, APP, APPEND, BIAS=value, BLKSET, CASCADE, DIAG, ERASE, EXCPVR=value, MAXPFIX=value, NEW, NEWFILE, NODIAG, NOERASE, NOINC, NOSWAP, OPT=value, REP, REPLACE, WRKADR=value, WRKDEV=value, and WRKSIZ=value.

ARESALL

```
►►─ARESALL=──┬─n──┬──────────────────────────────────────────────────►◄
             ├─nK─┤
             └─nM─┘
```

Temporarily overrides the ARESALL installation option, which specifies the number of bytes to be reserved above virtual for system use.

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

n specifies that n bytes of storage are to be reserved.

Limit: 8 digits.

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

Limit: 5 digits.

nM specifies that n times 1048576 bytes of storage are to be reserved.

Limit: 2 digits.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See“Appendix B. Specification/Override of DFSORT Options” on page 549.

ARESINV

```
►►─ARESINV=──┬─n──┬──────────────────────────────────────────────────►◄
             ├─nK─┤
             └─nM─┘
```

Temporarily overrides the ARESINV installation option, which specifies the number of bytes to be reserved for an invoking program’s user exits that reside in or use space above 16MB virtual. The reserved space is not meant to be used for the invoking program’s executable code. ARESINV is used only when DFSORT is dynamically invoked.

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

n specifies that n bytes of storage are to be reserved.

Limit: 8 digits.

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

Limit: 5 digits.

nM
specifies that n times 1048576 bytes of storage are to be reserved.

Limit: 2 digits.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

AVGRLEN

```
►►──AVGRLEN=n──────────────────────────────────────────────────────────►◄
```

Specifies the average input record length in bytes for variable-length record sort applications. This value is used when necessary to determine the input file size. The resulting value is important for sort applications, since it is used for several internal optimizations as well as for dynamic work data set allocation (see OPTION DYNALLOC). See “Specify Input/Output Data Set Characteristics Accurately” on page 492 and “Allocation of Work Data Sets” on page 541 for more information on file size considerations.

n specifies the average input record length. n must be between 4 and 32767 and must include the 4-byte record descriptor word (RDW).

Notes:

1. AVGRLEN=n on the OPTION statement overrides the L5 value on the RECORD statement (LENGTH operand) if both are specified. The L5 value on the RECORD statement is ignored for Blockset.
2. L5=n can be used instead of AVGRLEN=n.

Default: If AVGRLEN=n is not specified, DFSORT uses one-half of the maximum record length as the average record length. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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

NOCHALT
specifies that format CH fields are not translated.

Note: If you use locale processing for SORT, MERGE, INCLUDE, or OMIT fields,you must not use CHALT. If you need alternate sequence processing for a particular field, use format AQ.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

CHECK or NOCHECK

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

Temporarily overrides the CHECK installation option, which specifies whether the record count should be checked for applications that use the E35 user exit routine without an output data set.

CHECK
specifies that the record count should be checked.

NOCHECK
specifies that the record count should not be checked.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See“Appendix B. Specification/Override of DFSORT Options” on page 549.

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
specifies that DFSORT should use control interval access when possible for VSAM data sets.

NOCINV
specifies that DFSORT should not use control interval access.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

CKPT

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

Activates the Checkpoint/Restart facility for sorts that use the Peerage or Vale techniques.

Since CKPT is only supported in the Peerage and Vale techniques, the Blockset technique must be bypassed for the Checkpoint/Restart facility to be used. Installation option IGNCKPT=NO causes Blockset to be bypassed when CKPT is specified at run-time. The NOBLKSET option can also be used to bypass Blockset at run-time.

A SORTCKPT DD statement must be coded when you use the Checkpoint/Restart facility (see SORTCKPT DD Statement).

Notes:

1. CHKPT can be used instead of CKPT.
2. Functions such as OUTFIL processing, which are supported only by the Blockset technique, cannot be used if the Checkpoint/Restart facility is used.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

COBEXIT

```
►►─COBEXIT──┬─COB1─┬──────────────────────────────────────────►◄
            └─COB2─┘
```

Temporarily overrides the COBEXIT installation option, which specifies the library for COBOL E15 and E35 routines.

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

COB2
specifies that COBOL E15 and E35 routines are run with either the VS COBOL II run-time library or the Language Environment run-time library.

Note: See “COBOL User Exit Requirements” on page 309 for additional information on the use of COBEXIT=COB2.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

COPY

```
►►─COPY──────────────────────────────────────────────────►◄
```

Causes DFSORT to copy a SORTIN data set or inserted records to the output data sets unless all records are disposed of by an E35 exit routine. Records can be edited by E15 and E35 exit routines; INCLUDE/OMIT, INREC, OUTREC, and OUTFIL statements; and SKIPREC and STOPAFT parameters. E35 is entered after each SORTIN or E15 record is copied.

The following must not be used in copy applications:
- BDAM data sets
- Dynamic link-editing.

See message ICE160A in *Messages, Codes and Diagnosis* for additional restrictions that apply to copy applications.

Note: User labels will not be copied to the output data sets.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

DSPSIZE

```
►►─DSPSIZE=──┬─MAX─┬──────────────────────────────────────────────────►◄
             └─n───┘
```

Temporarily overrides the DSPSIZE installation option, which specifies the maximum amount of data space to be used with dataspace sorting. A data space is an area of contiguous virtual storage that is backed by real, expanded, and auxiliary storage, whichever is necessary as determined by the system. Because DFSORT is able to sort large pieces of data using data space, CPU time and elapsed time are reduced.

The amount of data space used by DFSORT is limited to the installation or user-specified DSPSIZE value and by the IEFUSI exit of your system. DSPSIZE=MAX (IBM-supplied default) means that DFSORT selects the maximum amount of data space to use based on the size of the input file and the paging activity of the system. You can further limit the amount of data space that DFSORT uses by specifying a maximum value in MB.

If the amount of data space DFSORT decides to use is sufficient, DFSORT sorts your data in main storage and does not require additional temporary work space. If the amount of data space is not sufficient, DFSORT uses DASD as temporary work space. Installation option DYNAUTO=NO is changed to DYNAUTO=YES whenever dataspace sorting is possible. Hiperspace is not used when dataspace sorting is used.

MAX
specifies that DFSORT dynamically determines the maximum amount of data space to be used for dataspace sorting. In this case, DFSORT bases its data space usage on the size of the file being sorted and the paging activity of the system.

n specifies the maximum amount, in MB, of data space to be used for dataspace sorting. n must be a value between 0 and 9999. The actual amount of data space used does not exceed n, but may be less than n depending on the size of the file being sorted and the paging activity of the system.

If n is zero, dataspace sorting is not used.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

DYNALLOC

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

Assigns DFSORT the task of dynamically allocating needed work space. You do not need to calculate and use JCL to specify the amount of work space needed by the program. DFSORT uses the dynamic allocation facility of the operating system to allocate work space for you.

Refer to “Appendix A. Using Work Space” on page 539 for guidelines on the use of DYNALLOC.

d specifies the device name. You can specify any IBM direct access storage device or tape device supported by your operating system in the same way you would specify it in the JCL UNIT parameter. You can also specify a group name, such as DISK or SYSDA.

For best performnce, specify an emulated 3390-9 device (such as RAMAC) or another high-speed IBM DASD device, and avoid specifying a tape, virtual (VIO), or real 3390-9 device.

n specifies the maximum number of requested work data sets. If you specify more than 255, a maximum of 255 data sets is used. If you specify 1 and the Blockset technique is selected, a maximum of 2 data sets is used. If you specify more than 32 and the Blockset technique is not selected, a maximum of 32 data sets is used.

Note: For optimum allocation of resources such as virtual storage, avoid specifying a large number of work data sets unnecessarily.

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 SORTWKdd DD statements are provided unless ICEMAC DYNAUTO=IGNWKDD is specified and OPTION USEWKDD is not in effect.

If VIO=NO is in effect

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

Note: Message ICE165I gives information about work data set allocation/use.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

- DYNALLOC can automatically be activated by using the ICEMAC DYNAUTO option.
- 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 IBM–supplied default for the n value of the DYNALOC installation option is chosen, then:

- If one of the Blockset techniques is chosen, four 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 B. Specification/Override of DFSORT Options” on page 549.

DYNALLOC=OFF

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

Directs DFSORT *not* to allocate work space dynamically, overriding that function of ICEMAC installation option DYNAUTO=YES, or DYNAUTO=IGNWKDD, or the run-time option DYNALLOC (without OFF). Use this option when you know that an in-core sort can be performed, and you want to suppress dynamic allocation of work space.

OFF
directs DFSORT not to allocate work space dynamically.

Note: When Hipersorting or dataspace sorting is in effect, DFSORT uses dynamic allocation when necessary, even if DYNALLOC=OFF has been specified.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 “Chapter 8. Using Extended Function Support” on page 455 for more information.

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

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

Notes:

1. EFS is processed only if it is passed on the OPTION control statement in an extended parameter list or in DFSPARM.
2. If you use locale processing for SORT, MERGE, INCLUDE, orOMIT fields, you must not use an EFS program. DFSORT’s locale processing may

eliminate the need for an EFS program. See the LOCALE option later in this section for information related to locale processing.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

EQUALS or NOEQUALS

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

Temporarily overrides the EQUALS installation option, which specifies whether the original sequence of records that collate identically 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 applications, the sequence of the output records 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 applications, the sequence of the output records depends upon the order of:

- The records from a SORTINnn file. Records that collate identically are output in the order of their file increments. For example, records from SORTIN01 are output before any records that collate identically from SORTIN02.
- The records from an E32 user exit routine for the same file increment number. Records that collate identically from E32 are output in the order of their file increments. For example, records from the file with increment 0 are output before any records that collate identically from the file with increment 4.

Notes:

1. When EQUALS is in effect, the total number of bytes occupied by all control fields must not exceed 4088.
2. Using EQUALS can degrade performance.
3. EQUALS is not used if SUM is specified and a technique other than Blockset is selected.
4. Do not specify EQUALS if variable-length records are sorted using tape work files and the RDW is part of the control field.
5. The number of records to be sorted cannot exceed 4294967295 (4 gigarecords minus 1); if the number of records exceeds this number, message ICE121A is issued and DFSORT terminates.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See Appendix B. Specification/Override of DFSORT Options.

EXITCK

```
►►──EXITCK──┬─STRONG─┬──────────────────────────────────────────────────►◄
            └─WEAK───┘
```

Temporarily overrides the EXITCK installation option, which specifies whether DFSORT terminates or continues when it receives certain invalid return codes from E15 or E35 user exit routines. For full details of the return codes affected by this parameter, see “E15/E35 Return Codes and EXITCK” on page 327.

STRONG
specifies that DFSORT issues an error message and terminates when it receives an invalid return code from an E15 or E35 user exit routine.

WEAK
specifies that DFSORT interprets certain invalid return codes from E15 and E35 user exit routines as valid and continues processing. Use of EXITCK=WEAK can make it difficult to detect errors in the logic of E15 and E35 user exit routines.

Note: EXITCK=WEAK is treated like EXITCK=STRONG when:

- Tape work data sets are specified for a sort application.
- The Blockset technique is not selected for a merge application.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See Appendix B. Specification/Override of DFSORT Options.

FILSZ or SIZE

```
►►──┬─FILSZ=──┬─x──┬──┬──────────────────────────────────────────────────►◄
    │         ├─Ex─┤  │
    │         └─Ux─┘  │
    └─SIZE=──┬─y──┬──┘
             ├─Ey─┤
             └─Uy─┘
```

The FILSZ parameter specifies either the exact number of records to be sorted or merged, or an estimate of the number of records to be sorted. The SIZE parameter specifies either the exact number of records in the input data sets, or an estimate of the number of records in the input data sets. The supplied record count is used by DFSORT for two purposes:

1. To check that the actual number of records sorted or merged or the number of records in the input data sets is equal to the exact number of records expected. FILSZ=x or SIZE=y causes this check to be performed and results in termination with message ICE047A if the check fails.
2. To determine the input file size for a sort application. DFSORT performs calculations based on the user supplied record count and other parameters (such as AVGRLEN) to estimate the total number of bytes to be sorted. This value is important for sort runs, since it is used for several internal optimizations as well as for dynamic work data set allocation (see OPTION

DYNALLOC). If no input record count (or only an estimate) is supplied for the sort run, DFSORT attempts to automatically compute the file size to be used for the optimizations and allocations.

The type of FILSZ or SIZE value specified (x/y, Ux/Uy, Ex/Ey, or none) controls the way DFSORT performs the above two functions, and can have a significant effect on performance and work data set allocation. See “Chapter 9. Improving Efficiency” on page 489 and “File Size and Dynamic Allocation” on page 543 for more information on file size considerations.

x or y
specifies the exact number of records to be sorted or merged (x) or the exact number of records in the input data sets (y). This value is always used for both the record check and the file size calculations. FILSZ=x or SIZE=y can be used to force DFSORT to perform file size calculations based on x or y, and to cause DFSORT to terminate the sort or merge application if x or y is not exact.

If the FSZEST=NO installation option is in effect and either FILSZ=x or SIZE=y is specified, DFSORT terminates if the actual number of records is different from the specified exact value (x or y). In this case, the actual number of records is placed in the IN field of message ICE047A (or message ICE054I in some cases) before termination. However, if the FSZEST=YES installation option is in effect, DFSORT treats FILSZ=x or SIZE=y like FILSZ=Ex or SIZE=Ey, respectively; it does not terminate when the actual number of records does not equal x or y.

FILSZ=0 causes Hipersorting, dataspace sorting, and dynamic allocation of work space not to be used, and results in termination with message ICE047A unless the number of records sorted or merged is 0. If no E15 user exit is present, SIZE=0 has the same effect in terms of Hipersorting and dynamic allocation of work space, and results in termination with message ICE047A unless the number of records in the input data sets is 0.

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 sets, records to be inserted or deleted by E15 or E32, and records to be deleted by the INCLUDE/OMIT statement, SKIPREC, and STOPAFT. x must be changed whenever the number of records to be sorted or merged changes in any way.

y specifies the exact number of records in the input data sets; it must take into account the number of records to be deleted by STOPAFT. y must be changed whenever the number of records in the input data sets changes in any way.

Limit: 28 digits (15 significant digits)

Ex or Ey
specifies an estimate of the number of records to be sorted (x) or an estimate of the number of records in the input data sets (y). This value is not used for the record check. It is used for the file size calculations, but only if DFSORT could not reasonably estimate the input file size itself. In all other cases, this value is ignored by DFSORT. See “Dynamic Allocation of Work Data Sets” on page 542 for details on exactly when an estimated record count is used and when it is ignored by DFSORT.

FILSZ=E0 or SIZE=E0 is always ignored.

x specifies an estimate of the number of records to be sorted; it should take into account the number of records in the input data sets, records to be inserted or deleted by E15, and records to be deleted by the INCLUDE/OMIT statement, SKIPREC, and STOPAFT. x should be changed whenever the number of records to be sorted changes significantly.

y specifies an estimate of the number of records in the input data sets; it should take into account the number of records to be deleted by STOPAFT. y should be changed whenever the number of records in the input data sets changes significantly.

Limit: 28 digits (15 significant digits)

Ux or Uy
specifies the number of records to be sorted (x) or the number of records in the input data sets (y). This value is not used for the record check, but is always used for the file size calculations. FILSZ=Ux or SIZE=Uy can be used to force DFSORT to perform file size calculations based on x or y, while avoiding termination if x or y is not exact.

The FSZEST installation option has no effect on FILSZ=Ux or SIZE=Uy processing.

FILSZ=U0 causes Hipersorting, dataspace sorting, and dynamic allocation of work space not to be used, and may cause degraded performance or termination with message ICE046A, if the actual number of records to be sorted is significantly larger than 0. If no E15 user exit is present, SIZE=U0 has the same effect in terms of Hipersorting, dataspace sorting,and dynamic allocation of work space, and may cause degraded performance or termination with message ICE046A, if the actual number of records in the input data sets is significantly larger than 0.

x specifies the number of records to be sorted; it should take into account the number of records in the input data sets, records to be inserted or deleted by E15, and records to be deleted by the INCLUDE/OMIT statement, SKIPREC, and STOPAFT. x should be changed whenever the number of records to be sorted changes significantly.

y specifies the number of records in the input data sets; it should take into account the number of records to be deleted by STOPAFT. y should be changed whenever the number of records in the input data sets changes significantly.

Limit: 28 digits (15 significant digits)

Table 21 summarizes the differences for the three FILSZ variations:

Table 21. FILSZ Variations Summary. FILSZ=n is equivalent to FILSZ=En if installation option FSZEST=YES is specified.

Conditions	FILSZ=n	FILSZ=Un	FILSZ=En
Number of records	Exact	Estimate	Estimate
Applications	Sort, merge	Sort	Sort
Terminate if n wrong?	Yes	No	No
Use for file size calculation?	Yes	Yes	When DFSORT cannot compute file size

Table 21. FILSZ Variations Summary (continued). FILSZ=n is equivalent to FILSZ=En if installation option FSZEST=YES is specified.

Conditions	FILSZ=n	FILSZ=Un	FILSZ=En
n includes records:			
In input data set(s)	Yes	Yes	Yes
Inserted/deleted by E15	Yes	Yes	Yes
Inserted by E32	Yes	No	No
Deleted by INCLUDE/OMIT	Yes	Yes	Yes
Deleted by SKIPREC	Yes	Yes	Yes
Deleted by STOPAFT	Yes	Yes	Yes
Update n when number of records changes:	In any way	Significantly	Significantly
Effects of n=0	Hipersorting and DYNALLOC not used	Hipersorting and DYNALLOC not used	None

Table 22 summarizes the differences for the three SIZE variations:

Table 22. SIZE Variations Summary. SIZE=n is equivalent to SIZE=En if installation option FSZEST=YES is specified.

Conditions	SIZE=n	SIZE=Un	SIZE=En
Number of records	Exact	Estimate	Estimate
Applications	Sort, merge	Sort	Sort
Terminate if n wrong?	Yes	No	No
Use for file size calculation?	Yes	Yes	When DFSORT cannot compute file size
n includes records:			
In input data set(s)	Yes	Yes	Yes
Inserted/deleted by E15	No	No	No
Inserted by E32	No	No	No
Deleted by INCLUDE/OMIT	No	No	No
Deleted by SKIPREC	No	No	No
Deleted by STOPAFT	Yes	Yes	Yes
Update n when number of records changes:	In any way	Significantly	Significantly
Effects of n=0	Hipersorting and DYNALLOC not used	Hipersorting and DYNALLOC not used	None

Note: Using the SIZE or FILSZ parameter to supply inaccurate information to DFSORT can negatively affect DFSORT performance, and, when work space is dynamically allocated, can result in wasted DASD space or termination with message ICE083A or ICE046A. Therefore, it is important to update the record count value whenever the number of records to be sorted changes significantly.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

HIPRMAX

```
►►──HIPRMAX=──┬─OPTIMAL─┬──────────────────────────────────────────────►◄
              ├─n───────┤
              └─p%──────┘
```

Temporarily overrides the HIPRMAX installation option, which specifies the maximum amount of Hiperspace to be used for Hipersorting. 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.

Several factors can limit the amount of Hiperspace an application uses:

- The IEFUSI exit can limit the total amount of Hiperspace and data space available to an application.
- HIPRMAX can limit the amount of Hiperspace available to an application, as detailed below.
- Sufficient available expanded storage must be present to back DFSORT’s Hiperspaces. Available expanded storage means expanded storage used to back new Hiperspace data and consists of the following two types:
  1. Free expanded storage. This is expanded storage not being used by any application.
  2. Old expanded storage. This is expanded storage used by another application whose data has been unreferenced for a sufficiently long time so that the systemmigrates it to auxiliary storage to make room for new Hiperspace data.

The amount of available expanded storage constantly changes, depending upon current system activity. Consequently, DFSORT checks the available expanded storage level throughout a Hipersorting application and switches from Hiperspace to work data sets if the available expanded storage level gets too low.

- Other concurrent Hipersorting applications further limit the amount of available expanded storage. A Hipersorting application knows the expanded storage needs of every other Hipersorting application on the system, and does not try to back its Hiperspace data with expanded storage needed by another Hipersorting application. This prevents overcommitment of expanded storage resources if multiple large concurrent Hipersorting applications start at similar times on the same system.
- The installation options EXPMAX, EXPOLD, and EXPRES can also be used to further limit the amount of expanded storage available to Hipersorting applications. EXPMAX limits the total amount of available expanded storage that can be used at any one time to back DFSORT Hiperspaces. EXPOLD limits the total amount of old expanded storage that can be used at any one time to back DFSORT Hiperspaces. EXPRES sets aside a specified amount of available expanded storage for use by non-Hipersorting applications.

Some of these limits depend on system and other Hipersorting activity throughout the time a Hipersorting application runs. Consequently, the amount of Hiperspace a Hipersorting application uses can vary from run to run.

HIPRMAX=n specifies a fixed value for HIPRMAX. HIPRMAX=p% specifies a value for HIPRMAX that varies as a percentage of the configured expanded storage on the system at run-time. If the configured expanded storage on a system changes, HIPRMAX=p% will cause a corresponding change in the HIPRMAX value selected by DFSORT, whereas HIPRMAX=n will not. When sharing DFSORT installation options between systems, such as in a sysplex, HIPRMAX=p% can be used to tailor the HIPRMAX value to the system selected for the application, providing a more dynamic HIPRMAX value than HIPRMAX=n.

If the amount of Hiperspace available for Hipersorting is insufficient for temporary storage of the records, intermediate DASD storage is used along with Hiperspace. If the amount of Hiperspace is too small to improve performance, Hipersorting is not used. DYNAUTO=NO is changed to DYNAUTO=YES for Hipersorting.

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

Note: HIPRLIM=OPTIMAL can be used instead of HIPRMAX=OPTIMAL. HIPRLIM=m can be used instead of HIPRMAX=n. HIPRLIM=m specifies a Hiperspace limit of m times 4096 bytes rounded up to the nearest megabyte. m must be a value between 0 and 2559744. If m is 0, Hipersorting is not used.

OPTIMAL
specifies that DFSORT determines dynamically the maximum amount of Hiperspace to be used for Hipersorting.

n specifies that DFSORT determines dynamically the maximum amount of Hiperspace to be used for Hipersorting, subject to a limit of nMB. n must be a value between 0 and 9999. If n is 0, Hipersorting is not used.

p%
specifies that DFSORT determines dynamically the maximum amount of hiperspace to be used for Hipersorting, subject to a limit of p percent of the configured expanded storage. p must be a value between 0 and 100. If p is 0, Hipersorting is not used. The value calculated for p% is limited to 9999MB, and is rounded down to the nearest MB.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 *Messages, Codes and Diagnosis* for details on use of the message data set.

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

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

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

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 *Messages, Codes and Diagnosis* for details on use of the message data set.

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

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

Notes:

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

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

LOCALE

```
►►──LOCALE=──┬─name────────┬──────────────────────────────────────────►◄
             ├─CURRENT─────┤
             └─NONE────────┘
```

Temporarily overrides the LOCALE installation option, which specifies whether locale processing is to be used and, if so, designates the active locale.

DFSORT's collating behavior can be modified according to your cultural environment. Your cultural environment is defined to DFSORT using the X/Open locale model. A locale is a collection of data grouped into categories that describes the information about your cultural environment.

The collate category of a locale is a collection of sequence declarations that defines the relative order between collating elements (single character and multi-character collating elements). The sequence declarations define the collating rules.

If locale processing is to be used, the active locale will affect the behavior of DFSORT's SORT, MERGE, INCLUDE, and OMIT functions. For SORT and MERGE, the active locale will only be used to process character (CH) control fields. For INCLUDE and OMIT, the active locale will only be used to process character (CH) compare fields, and character and hexadecimal constants compared to character (CH) compare fields.

name specifies that locale processing is to be used and designates the name of the locale to be made active during DFSORT processing.

The locales are designated using a descriptive name. For example, to set the active locale to represent the French language and the cultural conventions of Canada, specify LOCALE=FR_CA. You can specify up to 32 characters for the descriptive locale name. The locale names themselves are not case-sensitive. See *Using Locales* for complete locale naming conventions.

You can use IBM-supplied and user-defined locales. See "Appendix F. Locales Supplied with C/370" on page 599 for examples of some IBM-supplied locales.

The state of the active locale prior to DFSORT being entered will be restored on DFSORT's completion.

CURRENT
specifies that locale processing is to be used, and the current locale active when DFSORT is entered will remain the active locale during DFSORT processing.

NONE specifies that locale processing is not to be used. DFSORT will use the binary encoding of the code page defined for your data for collating and comparing.

Notes:

1. LOCALE is processed only if it is passed on the OPTION control statement in an extended parameter list or in DFSPARM.
2. To use an IBM-supplied locale, DFSORT must have access to the Language Environment run-time library. For example, this library might be called SYS1.SCEERUN. If you are unsure of the name of this library at your location, contact your system administrator. To use a user-defined locale, DFSORT must have access to the load library containing it.
3. If you use locale processing for SORT, MERGE, INCLUDE, or OMIT fields:
   - VLSHRT is not used for SORT or MERGE
   - CHALT, INREC, an EFS program, or an E61 user exit must not be used.

4. Locale processing for DFSORT's SORT, MERGE, INCLUDE, and OMIT functions can improve performance relative to applications which perform pre- and/or post-processing of data to produce the desired collating results. However, locale processing should be used only when required because it can show degraded performance relative to collating, using character encoding values.
5. DFSORT locale processing may require an additional amount of storage that depends on the environment supporting the locale as well as the locale itself. It may be necessary to specify a REGION of several MB or more for DFSORT applications that use locale processing.

Default: Usually the installation default. See "Appendix B. Specification/Override of DFSORT Options" on page 549 for full override details.

Applicable Functions:; See "Appendix B. Specification/Override of DFSORT Options" on page 549.

MAINSIZE

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

Temporarily overrides the SIZE installation option, which specifies the amount of main storage available to DFSORT. The value you specify must be greater than the MINLIM value set at DFSORT installation time.

MAINSIZE applies to the total amount of main storage above and below 16MB virtual. DFSORT determines how much storage to allocate above and below 16MB virtual, but the total amount of storage cannot exceed MAINSIZE.

Storage used for OUTFIL processing will be adjusted automatically, depending upon several factors, including:

- Total available storage
- Non-OUTFIL processing storage requirements
- Number of OUTFIL data sets and their attributes (for example, block size).

OUTFIL processing is subject to the ODMAXBF limit and your system storage limits (for example, IEFUSI) but not to DFSORT storage limits, that is, SIZE/MAINSIZE, MAXLIM, and TMAXLIM. DFSORT attempts to use storage above 16MB virtual for OUTFIL processing whenever possible.

For details on main storage allocation, see "Tuning Main Storage" on page 498.

n specifies that n bytes of storage are to be allocated. If you specify more that 2097152000, 2097152000 is used.

Limit: 10 digits

nK
specifies that n times 1024 bytes of storage are to be allocated. If you specify more than 2048000K, 2048000K is used.

Limit: 7 digits

nM
specifies that n times 1048576 bytes of storage are to be allocated. If you specify more than 2000M, 2000M is used.

Limit: 4 digits.

MAX
instructs DFSORT to calculate the amount of virtual storage available and allocate an amount of storage up to the TMAXLIM or DSA installation value when Blockset is selected, or up to the MAXLIM installation value when Blockset is not selected.

Note: CORE=value can be used instead of MAINSIZE=value.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 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 run-time, SYSOUT is used instead. For details on use of the message data set, see *Messages, Codes and Diagnosis*

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

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 *Messages, Codes and Diagnosis*.

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.

Notes:

1. MSGPRT is processed only if it is passed on the OPTION control statement in an extended parameter list or in DFSPARM.
2. PRINT=value can be used instead of MSGPRT=value.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

NOBLKSET

```
►►──NOBLKSET──────────────────────────────────────────────────────────►◄
```

Causes DFSORT to bypass the Blockset technique normally used for a sort or merge application. Using this option generally results in degraded performance.

Note: Functions such as OUTFIL processing, which are supported only by the Blockset technique, cause the NOBLKSET option to be ignored.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

NOOUTREL

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

Temporarily overrides the OUTREL installation option, which specifies whether unused temporary output data set space is released. NOOUTREL means that unused temporary output data set space is *not* released.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

NOOUTSEC

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

Temporarily overrides the OUTSEC installation option, which specifies whether automatic secondary allocation is used for temporary or new output data sets. NOOUTSEC means that automatic secondary allocation for output data sets is not used.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 exits take checkpoints and STIMER=YES is the installation default, you must specify this parameter.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

ODMAXBF

```
►►──ODMAXBF=──┬─n──┬──────────────────────────────────────────────────►◄
              ├─nK─┤
              └─nM─┘
```

Temporarily overrides the ODMAXBF installation option, which specifies the maximum buffer space DFSORT can use for each OUTFIL data set. The actual amount of buffer space used for a particular OUTFIL data set will not exceed the ODMAXBF limit, but can be less than the limit. OUTFIL processing is supported by the Blockset technique for sort, copy, and merge applications.

The storage used for OUTFIL processing is adjusted automatically according to the total storage available, the storage needed for non-OUTFIL processing, and the number of OUTFIL data sets and their attributes (for example, block size). OUTFIL processing is subject to the ODMAXBF limit in effect and the system storage limits (for example, IEFUSI), but not to the DFSORT storage limits (that is, SIZE, MAXLIM, and TMAXLIM). DFSORT attempts to use storage above 16MB virtual for OUTFIL processing whenever possible.

Lowering ODMAXBF below 2M can cause performance degradation for the application, but may be necessary if you consider the amount of storage used

for OUTFIL processing to be a problem. Raising ODMAXBF can improve EXCPs for the application but can also increase the amount of storage needed.

n specifies that a maximum of n bytes of buffer space is to be used for each OUTFIL data set. If you specify less than 262144, 262144 is used. If you specify more than 16777216, 16777216 is used.

Limit: 8 digits

nK
specifies that a maximum of n times 1024 bytes of buffer space is to be used for each OUTFIL data set. If you specify less than 256K, 256K is used. If you specify more than 16384K, 16384K is used.

Limit: 5 digits

nM
specifies that a maximum of n times 1048576 bytes of buffer space is to be used for each OUTFIL data set. If you specify 0M, 256K is used. If you specify more than 16M, 16M is used.

Limit: 2 digits

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

OVFLO

```
►►──OVFLO=──┬─RC0──┬──────────────────────────────────────────────►◄
            ├─RC4──┤
            └─RC16─┘
```

Temporarily overrides the OVFLO installation option, which specifies the action to be taken by DFSORT when BI, FI, PD or ZD summary fields overflow.

RC0
specifies that DFSORT should issue message ICE152I (once), set a return code of 0 and continue processing when summary fields overflow. The pair of records involved in a summary overflow is left unsummed and neither record is deleted. Summary overflow does not prevent further summation.

RC4
specifies that DFSORT should issue message ICE152I (once), set a return code of 4 and continue processing when summary fields overflow. The pair of records involved in a summary overflow is left unsummed and neither record is deleted. Summary overflow does not prevent further summation.

RC16
specifies that DFSORT should issue message ICE195A, terminate and give a return code of 16 when summary fields overflow.

Note: The return code of 0 or 4 set for summary overflow can be overridden by a higher return code set for some other reason.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

PAD

```
►►──PAD=──┬─RC0──┬──────────────────────────────────────────────────►◄
          ├─RC4──┤
          └─RC16─┘
```

Temporarily overrides the PAD installation option, which specifies the action to be taken by DFSORT when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL, for the cases where DFSORT allows LRECL padding.

RC0
specifies that DFSORT should issue message ICE171I, set a return code of 0 and continue processing when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL.

RC4
specifies that DFSORT should issue message ICE171I, set a return code of 4 and continue processing when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL.

RC16
specifies that DFSORT should issue message ICE196A, terminate and give a return code of 16 when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL.

Notes:

1. The return code of 0 or 4 set for LRECL padding can be overridden by a higher return code set for some other reason.
2. For an ICEGENER application, the GNPAD value is used and the PAD value is ignored.
3. For some LRECL padding situations (for example, a tape work data set sort), DFSORT issues ICE043A and terminates with a return code of 16. The PAD value has no effect in these cases.
4. DFSORT does not check for LRECL padding if:
   a. A SORTIN DD (sort/copy), SORTINnn DD (merge) or SORTOUT DD is not present
   b. A SORTIN DD (sort/copy), SORTINnn DD (merge) or SORTOUT DD specifies a VSAM data set.
5. DFSORT does not check OUTFIL data sets for LRECL padding.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

RESALL

```
►►──RESALL=──┬─n──┬──────────────────────────────────────────────────►◄
             ├─nK─┤
             └─nM─┘
```

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 may not be enough; for example, if your installation does not have BSAM/QSAM modules resident, you have user exits that open data sets, or you have COBOL exits. RESALL is used only when MAINSIZE/SIZE=MAX is in effect.

RESALL applies only to the amount of main storage below 16MB virtual. The ARESALL option applies to the amount of main storage above 16MB virtual.

n specifies that n bytes of storage are to be reserved. If you specify less than 4096, 4096 is used.

Limit: 8 digits.

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

Limit: 5 digits.

nM
specifies that n times 1048576 bytes of storage are to be reserved. If you specify 0M, 4K is used.

Limit: 2 digits.

Note: A better way to reserve the required storage for user exits activated by the MODS statement is to use the *m* parameter of the MODS statement.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

RESET or NORESET

```
►►──┬─RESET──┬──────────────────────────────────────────────────►◄
    └─NORESET─┘
```

Temporarily overrides the RESET installation option, which specifies whether DFSORT should process a VSAM output data set defined with REUSE as a NEW or MOD data set.

RESET
specifies that DFSORT processes a VSAM output data set defined with REUSE as a NEW data set. The high-used RBA is reset to zero and the output data set is effectively treated as an initially empty cluster.

NORESET
specifies that DFSORT processes a VSAM output data set defined with REUSE as a MOD data set. The high-used RBA is not reset and the output data set is effectively treated as an initially non-empty cluster.

Note: A VSAM output data set defined without REUSE is processed as a MOD data set.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

RESINV

```
►►─RESINV=──┬─n──┬──────────────────────────────────────────────────────►◄
            ├─nK─┤
            └─nM─┘
```

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.

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

This extra space is usually required for data handling by the invoking program or user exits while DFSORT is running (as is the case with some PL/I- and COBOL- invoked sort applications). Therefore, if your invoking program’s user exits do not perform data set handling, you do not need to specify this parameter. The reserved space is not meant to be used for the invoking program’s executable code.

The amount of space required depends upon what routines you have, how the data is stored, and which access method you use.

n specifies that n bytes of storage are to be reserved.

Limit: 8 digits

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

Limit: 5 digits

nM
specifies n times 1048576 bytes of main storage are to be reserved.

Limit: 2 digits.

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

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SDB

```
►►──SDB=──┬─LARGE─┬──────────────────────────────────────────────────►◄
          ├─YES───┤
          ├─INPUT─┤
          └─NO────┘
```

Temporarily overrides the SDB installation option, which specifies whether DFSORT should use the system-determined optimum block size for output data sets when the block size is specified as zero or defaulted to zero. System-determined block size applies to both SMS-managed and non-SMS-managed data sets and results in the most efficient use of space for the device on which the output data set resides.

With OS/390 Release 10 and above, DFSORT can select system-determined optimum block sizes greater than 32760 bytes for tape output data sets.

If you want DFSORT to use system-determined block sizes for DASD and tape output data sets, specify one of the following values:

- SDB=LARGE if you want to allow DFSORT to select tape output block sizes greater than 32760 bytes.
- SDB=YES (or its alias SDB=SMALL) if you want DFSORT to select tape output block sizes less than or equal to 32760 bytes.
- SDB=INPUT if you want to allow DFSORT to select tape output block sizes greater than 32760 bytes only when tape input data sets with block sizes greater than 32760 bytes are used.

DFSORT will not select a tape output block size greater than the BLKSZLIM in effect. In particular, if a default BLKSZLIM of 32760 is in effect, DFSORT will not select a tape output block size greater than 32760 bytes. Therefore, in order to allow DFSORT to select tape output block sizes greater than 32760 bytes for particular jobs, you may need to ensure that your JCL or data class supplies appropriately large BLKSZLIM values (for example, 1GB) for those jobs.

If you don't want DFSORT to use system-determined block sizes, specify SDB=NO (not recommended as an installation option).

LARGE
specifies that DFSORT is to use the system-determined optimum block size for an output data set when its block size is zero. With OS/390 Release 10 and above, SDB=LARGE allows DFSORT to select a block size greater than 32760 bytes for a tape output data set, when appropriate. A larger tape block size can improve elapsed time and tape utilization, but you must ensure that applications which subsequently use the resulting tape data set can handle larger block sizes.

DFSORT selects the system-determined optimum block size as follows:

- For a DASD output data set, the optimum block size for the device used is selected based on the obtained or derived RECFM and LRECL for the output data set. The maximum block size for DASD output data sets is 32760 bytes.
- For a tape output data set, the optimum block size is selected based on the obtained or derived RECFM and LRECL for the output data set, as shown in Figure 14 on page 162.

RECFM	BLKSIZE is set to:
F or FS	LRECL
FB or FBS	Highest possible multiple of LRECL that is less than or equal to the optimum block size for the device, subject to the BLKSZLIM in effect.
V, D, VS, or DS	LRECL + 4
VB, DB, VBS, or DBS	Optimum block size for the device, subject to the BLKSZLIM in effect.

Figure 14. SDB=LARGE Block Sizes for Tape Output Data Sets

DFSORT uses the system-determined optimum block size for the output data set in most cases when the block size is zero. However, the following conditions prevent DFSORT from using the system-determined block size:

- Output data set block size is available (that is, non-zero) in the JFCB (DASD or tape) or format 1 DSCB (DASD) or tape label (only for DISP=MOD with AL, SL, or NSL label, when appropriate)
- Not MVS/DFP 3.1.0 or above
- Output is a spool, dummy, VSAM, or unmovable data set, or an HFS file.
- The output data set is on tape with a label type of AL
- DFSORT's Blockset technique is **not** selected.

In the above cases, DFSORT uses the specified block size, or determines an appropriate (though not ncessarily optimum) block size for the output data set. The selected block size is limited to 32760 bytes.

YES
specifies that DFSORT is to use the system-determined optimum block size for an output data set when its block size is zero, but is to limit the selected block size to a maximum of 32760 bytes. See the discussion of SDB=LARGE for more information; the only difference between SDB=LARGE and SDB=YES is that SDB=LARGE allows block sizes greater than 32760 bytes for tape output data sets, whereas SDB=YES does not.

INPUT
specifies that DFSORT is to use the system-determined optimum block size for an output data set when its block size is zero, but is to limit the selected block size to a maximum of 32760 bytes if the input block size is less than or equal to 32760 bytes. Thus, SDB=INPUT works like SDB=LARGE if the input block size is greater than 32760 bytes (only possible for tape input data sets) and works like SDB=YES if the input block size is less than or equal to 32760 bytes. See the discussions of SDB=LARGE and SDB=YES for more information.

NO
specifies that DFSORT is not to use the system-determined optimum block size. When the output data set block size is zero, DFSORT selects an appropriate (though not necessarily optimum) block size for the output data set based on the obtained or derived output or input attributes. SDB=NO limits the selected block sizes to a maximum of 32760 bytes.

SDB=NO works like SDB=YES if the input block size is greater than 32760 bytes (only possible for tape input data sets). See the discussion of SDB=YES for more information.

Notes:

1. With OS/390 R9 and below, SDB=LARGE and SDB=INPUT work like SDB=YES.
2. SDB=NO does not prevent the use of system-determined block size for the output data set at allocation or in other cases where the output data set block size is set before DFSORT gets control.
3. When DFSORT uses system-determined block size, the selected output data set block size may be different from the block size selected previously. Applications that require a specific output data set block size should be changed to specify that block size explicitly.
4. SDB and SDB=SMALL can be used instead of SDB=YES. NOSDB can be used instead of SDB=NO.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SIZE

See FILSZ.

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 usually used if, on a preceding DFSORT run, you have processed only part of the input data set.

An application 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 4. Using Your Own User Exit Routines” on page 279) 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.

z specifies the number of records to be skipped.

Limit: 28 digits (15 significant digits)

Notes:

1. SKIPREC applies only to records read from SORTIN (not from E15 routines). (See Figure 2 on page 8.)
2. If SKIPREC=0 is in effect, SKIPREC is not used.
3. You may want to consider using the STARTREC parameter of the OUTFIL statement as an alternative to using SKIPREC.

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SMF

```
►►──SMF=──┬─SHORT─┬──────────────────────────────────────────────────►◄
          ├─FULL──┤
          └─NO────┘
```

Temporarily overrides the SMF installation option, which specifies whether a DFSORT SMF record is to be produced as described in *Installation and Customization*.

SHORT
specifies that DFSORT is to produce a short SMF type-16 record for a successful run. The short SMF record does not contain record-length distribution statistics or data set sections.

FULL
specifies that DFSORT is to produce a full SMF type-16 record for a successful run. The full SMF record contains the same information as the short record, as well as record-length distribution and data set sections, as appropriate.

NO
specifies that DFSORT is not to produce an SMF type-16 record for this run.

Notes:

1. SMF is processed only if it is passed on the OPTION control statement in an extended parameter list or in DFSPARM.
2. SMF=FULL can degrade performance for a variable-length record application.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SOLRF or NOSOLRF

```
►►──┬─SOLRF───┬──────────────────────────────────────────────────────►◄
    └─NOSOLRF─┘
```

Temporarily overrides the SOLRF installation option, which specifies whether DFSORT should set the SORTOUT LRECL to the reformatted record length when the SORTOUT LRECL is unknown.

SOLRF
specifies that DFSORT should use the reformatted record length for the SORTOUT LRECL when the SORTOUT LRECL is not specified or available. DFSORT will use one of the following for the SORTOUT LRECL, in the order listed:

1. The SORTOUT LRECL if available from the JFCB, format 1 DSCB, DFSMSrmm, or tape label
2. The L3 length if specified in the RECORD statement

3. The OUTREC length if the OUTREC statement is specified
4. The INREC length if the INREC statement is specified
5. The L2 length if specified in the RECORD statement providing an E15 user exit is present
6. The SORTIN or SORTINnn LRECL if available from the JFCB, format 1 DSCB, DFSMSrmm, or tape label
7. The L1 length in the RECORD statement

NOSOLRF
specifies that DFSORT should not use the reformatted record length for the SORTOUT LRECL. DFSORT will use one of the following for the SORTOUT LRECL, in the order listed:

1. The SORTOUT LRECL if available from the JFCB, format 1 DSCB, DFSMSrmm, or tape label
2. The L3 length if specified in the RECORD statement providing an E35 exit, OUTREC statement or INREC statement is present
3. The L2 length if specified in the RECORD statement providing an E15 user exit is present
4. The SORTIN or SORTINnn LRECL if available from the JFCB, format 1 DSCB, DFSMSrmm, or tape label
5. The L1 length in the RECORD statement

Notes:

1. With SOLRF in effect (the IBM-supplied default), DFSORT sets the SORTOUT LRECL to the INREC or OUTREC record length when appropriate, which is usually what you want when you use INREC or OUTREC. If you want DFSORT to use the input length for the SORTOUT LRECL even when INREC or OUTREC is present, you can use NOSOLRF, but be aware that this can cause padding or truncation of the reformatted records, or termination.
2. CAOUTREC can be used instead of SOLRF.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SORTDD

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

Specifies a four-character prefix for the ddnames to be used when you dynamically invoke DFSORT more than once in a program step. The four characters replace “SORT” in the following ddnames: SORTIN, SORTOUT, SORTINn, SORTINnn, SORTOFd, SORTOFdd, SORTWKd, SORTWKdd, and SORTCNTL. This allows you to use a different set of ddnames for each call to DFSORT.

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.

For example, if you use ABC# as replacement characters, DFSORT uses DD statements ABC#IN, ABC#CNTL, ABC#WKdd, and ABC#OUT instead of SORTIN, SORTCNTL, SORTWKdd, and SORTOUT.

Notes:

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

Default: SORT. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 data set.

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).

Notes:

1. SORTIN is processed only if it is passed on the OPTION control statement in an extended parameter list, or in DFSPARM.
2. 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.
3. If SORTIN is used for a tape work data set sort, DFSORT terminates.

Default: SORTIN, unless SORTDD=cccc is specified in which case ccccIN is the default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SORTOUT

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

Specifies a ddname to be associated with the SORTOUT data set. This allows you to dynamically invoke DFSORT more than once in a program step, passing a different ddname for each output data set.

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).

Notes:

1. SORTOUT is processed only if it is passed on the OPTION control statement in an extended parameter list or in DFSPARM.
2. 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.
3. If SORTOUT is specified for a conventional merge or for a tape work data set sort, DFSORT terminates.

Default: SORTOUT, unless SORTDD=cccc is specified, in which case ccccOUT is the default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SPANINC

```
►►──SPANINC=──┬─RC0──┬──────────────────────────────────────────►◄
              ├─RC4──┤
              └─RC16─┘
```

Temporarily overrides the SPANINC installation option, which specifies the action to be taken by DFSORT when one or more incomplete spanned records are detected in a variable spanned input data set.

RC0
specifies that DFSORT should issue message ICE197I (once), set a return code of 0 and eliminate all incomplete spanned records it detects. Valid records will be recovered.

RC4
specifies that DFSORT should issue message ICE197I (once), set a return code of 4 and eliminate all incomplete spanned records it detects. Valid records will be recovered.

RC16
specifies that DFSORT should issue message ICE204A, terminate and give a return code of 16 when an incomplete spanned record is detected.

Notes:

1. The return code of 0 or 4 set for incomplete spanned records can be overridden by a higher return code set for some other reason.
2. In cases where a spanned record cannot be properly assembled (for example, it has a segment length less than 4 bytes), DFSORT issues ICE141A and terminates with a return code of 16. The SPANINC value has no effect in these cases.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 the INCLUDE/OMIT statement). 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.

n specifies the maximum number of records to be accepted.

Limit: 28 digits (15 significant digits)

Default: None; optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

SZERO or NOSZERO

```
►►──┬─SZERO───┬──────────────────────────────────────────────────────►◄
    └─NOSZERO─┘
```

Temporarily overrides the SZERO installation option, which specifies whether DFSORT should treat numeric -0 and +0 values as signed (that is, different) or unsigned (that is, the same) for collation, comparisons, editing and conversions, minimums and maximums. The following DFSORT control statements are affected by this option: INCLUDE, INREC, MERGE, OMIT, OUTFIL, OUTREC and SORT.

SZERO
specifies that DFSORT should treat numeric zero values as signed. -0 and +0 are treated as **different** values, that is, -0 is treated as a negative value and +0 is treated as a positive value. SZERO affects DFSORT processing of numeric values as follows:

- For collation of SORT and MERGE fields, -0 collates before +0 in ascending order and after +0 in descending order.
- For comparisons of INCLUDE, OMIT and OUTFIL INCLUDE and OMIT fields and constants, -0 compares as less than +0.
- For editing and conversions of INREC, OUTREC and OUTFIL OUTREC fields, -0 is treated as negative and +0 is treated as positive.
- For minimums and maximums of OUTFIL TRAILERx fields, -0 is treated as negative and +0 is treated as positive.

NOSZERO
specifies that DFSORT should treat numeric zero values as unsigned. -0 and +0 are treated as **the same** value, that is, -0 and +0 are both treated as positive values. NOSZERO affects DFSORT processing of numeric values as follows:

- For collation of SORT and MERGE fields, -0 collates equally with +0.
- For comparisons of INCLUDE, OMIT and OUTFIL INCLUDE and OMIT fields and constants, -0 compares as equal to +0.

- For editing and conversions of INREC, OUTREC and OUTFIL OUTREC fields, -0 and +0 are treated as positive.
- For minimums and maximums of OUTFIL TRAILERx fields, -0 and +0 are treated as positive.

Note: OPTION SZERO or OPTION NOSZERO is ignored for OUTFIL INCLUDE=(...) or OUTFIL OMIT=(...) if the OPTION statement is ″found″ **after** the OUTFIL statement. To avoid this, specify SZERO or NOSZERO as an EXEC/DFSPARM PARM option, or in an OPTION statement **before** the OUTFIL statement in the same source, for example:

```
//SYSIN DD *
 OPTION NOSZERO,COPY
 OUTFIL INCLUDE=(...)
/*
```

Default: Usually, the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

TRUNC

```
►►─TRUNC=──┬─RC0──┬──────────────────────────────────────────►◄
           ├─RC4──┤
           └─RC16─┘
```

Temporarily overrides the TRUNC installation option, which specifies the action to be taken by DFSORT when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL, for the cases where DFSORT allows LRECL truncation.

RC0
specifies that DFSORT should issue message ICE171I, set a return code of 0 and continue processing when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL.

RC4
specifies that DFSORT should issue message ICE171I, set a return code of 4 and continue processing when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL.

RC16
specifies that DFSORT should issue message ICE196A, terminate and give a return code of 16 when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL.

Notes:

1. The return code of 0 or 4 set for LRECL truncation can be overridden by a higher return code set for some other reason.
2. For an ICEGENER application, the GNTRUNC value is used and the TRUNC value is ignored.
3. For some LRECL truncation situations (for example, a tape work data set sort), DFSORT issues ICE043A and terminates with a return code of 16. The TRUNC value has no effect in these cases.
4. DFSORT does not check for LRECL truncation if:

a. A SORTIN DD (sort/copy), SORTINnn DD (merge) or SORTOUT DD is not present
b. A SORTIN DD (sort/copy), SORTINnn DD (merge) or SORTOUT DD specifies a VSAM data set.

5. DFSORT does not check OUTFIL data sets for LRECL truncation.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

USEWKDD

```
►►──USEWKDD──────────────────────────────────────────────────────────►◄
```

Temporarily overrides the DYNAUTO=IGNWKDD option, which specifies that dynamic work data sets are used even if SORTWKdd DD statements are present. This option allows JCL SORTWKdd data sets to be used rather than deallocated.

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

Default: None, optional. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Function: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

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 performed.

NOVERIFY
specifies that sequence checking is not performed.

Notes:

1. Using VERIFY can degrade performance.
2. SEQ=YES can be used instead of VERIFY, SEQ=NO can be used instead of NOVERIFY.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

VLLONG or NOVLLONG

```
►►──┬─VLLONG──┬──────────────────────────────────────────────────────►◄
    └─NOVLLONG─┘
```

Temporarily overrides the VLLONG installation option, which specifies whether DFSORT is to truncate ″long″ variable-length output records. A long output record is one whose length is greater than the LRECL of the SORTOUT or OUTFIL data set it is to be written to.

VLLONG is not meaningful for fixed-length output record processing.

VLLONG
specifies that DFSORT truncates long variable-length output records to the LRECL of the SORTOUT or OUTFIL data set.

NOVLLONG
specifies that DFSORT terminates if a long variable-length output record is found.

Notes:

1. VLLONG should not be used unless you want the data at the end of long variable-length output records to be truncated for your DFSORT application; inappropriate use of VLLONG can result in unwanted loss of data.
2. VLLONG can be used to truncate long OUTFIL data records, but has no effect on long OUTFIL header or trailer records.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

VLSCMP or NOVLSCMP

```
►►──┬─VLSCMP──┬──────────────────────────────────────────────────────►◄
    └─NOVLSCMP─┘
```

Temporarily overrides the VLSCMP installation option, which specifies whether DFSORT is to pad ″short″ variable-length INCLUDE/OMIT compare fields with binary zeros. A short field is one where the variable-length record is too short to contain the entire field, that is, the field extends beyond the record. VLSCMP and NOVLSCMP apply to the INCLUDE and OMIT statements and to the INCLUDE and OMIT parameters of the OUTFIL statement.

The compare fields are only padded temporarily for testing; they are not actually changed for output.

VLSCMP is not meaningful for fixed-length record processing.

The settings for VLSCMP/NOVLSCMP and VLSHRT/NOVLSHRT provide three levels of processing for short INCLUDE/OMIT fields in the following hierarchy:

1. VLSCMP allows all of the INCLUDE/OMIT comparisons to be performed even if some fields are short. Since short fields are padded with binary zeros, comparisons involving short fields are false (unless a test against binary zero is relevant, as discussed below). Comparisons involving non-short fields can be true or false.

2. NOVLSCMP and VLSHRT treat the entire INCLUDE/OMIT logical expression as false if any field is short. Thus comparisons involving non-short fields are ignored if any comparison involves a short field.
3. NOVLSCMP and NOVLSHRT result in termination if any field is short.

To illustrate how this works, suppose the following INCLUDE statement is used:

```
INCLUDE COND=(6,1,CH,EQ,C'1',OR,70,2,CH,EQ,C'T1')
```

If a variable-length input record has a length less than 71 bytes, the field at bytes 70-71 is short and the following occurs:

- With VLSCMP, the record is included if byte 6 of the input record is C’1’ or omitted if byte 6 is not C’1’. The comparison of bytes 70-71 equal to C’T1’ is false because bytes 70-71 contain either X’hh00’ (for a record length of 70 bytes) or X’0000’ (for a record length of less than 70 bytes). The comparison involving the non-short field is performed even though a short field is present.
- With NOVLSCMP and VLSHRT, the record is omitted because any short field makes the entire logical expression false. The comparison involving the non-short field is not performed because a short field is present.
- With NOVLSCMP and NOVLSHRT, DFSORT terminates because any short field results in termination.

In general, comparisons involving short fields are false with VLSCMP. However, if a binary zero value is relevant to the comparison, the use of binary zeros for padding might make the comparison true. For example, suppose the following INCLUDE statement is used:

```
INCLUDE COND=(21,2,CH,EQ,C'JX',OR,
              (55,2,CH,EQ,58,2,CH,AND,
               70,1,BI,LT,X'08'))
```

If a variable-length input record has a length less than 70 bytes, the field at byte 70 is short and is padded to X’00’. This makes the comparison of byte 70 less than X’08’ true even though byte 70 is a short field and so probably irrelevant.

Likewise, if a variable-length record has a length less than 55 bytes, the fields at bytes 55-56 and 58-59 are short and are each padded to X’0000’, and the field at byte 70 is short and is padded to X’00’. This makes the comparison of bytes 55-56 equal to 58-59 true and the comparison of byte 70 less than X’08’ true even though all three fields are short and probably irrelevant.

In such cases where padding of short fields with binary zeros may result in unwanted true comparisons, you can get the result you want by adding an appropriate check of the record length to the INCLUDE/OMIT logical expression, such as:

```
INCLUDE COND=(21,2,CH,EQ,C'JX',OR,
              (1,2,BI,GE,X'0046',AND,
               55,2,CH,EQ,58,2,CH,AND,
               70,1,BI,LT,X'08'))
```

Now the comparisons involving bytes 55-56, 58-59 and 70 can only be true for records that are 70 bytes (X’0046’) or longer. Thus, the irrelevant comparisons involving short fields are eliminated.

Keep in mind that short compare fields are padded with zeros when VLSCMP is in effect and code your INCLUDE/OMIT logical expressions to allow for that or even take advantage of it.

VLSCMP
specifies that short variable-length compare fields are padded with binary zeros.

NOVLSCMP
specifies that short variable-length compare fields are not padded.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

VLSHRT or NOVLSHRT

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

Temporarily overrides the VLSHRT installation option, which specifies whether DFSORT is to continue processing if a ″short″ variable-length SORT/MERGE control field or INCLUDE/OMIT compare field is found. A short field is one where the variable-length record is too short to contain the entire field, that is, the field extends beyond the record. VLSHRT applies to the SORT, MERGE, INCLUDE and OMIT statements, and to the INCLUDE and OMIT parameters of the OUTFIL statement.

VLSHRT processing is not meaningful for fixed-length record processing.

VLSCMP/NOVLSCMP and VLSHRT/NOVLSHRT provide three levels of processing for short INCLUDE/OMIT fields. For details, see the discussion of the VLSCMP and NOVLSCMP options.

VLSHRT
specifies that DFSORT continues processing if a short control field or compare field is found.

NOVLSHRT
specifies that DFSORT terminates if a short control field or compare field is found.

Notes:

1. VLSHRT is not used if an INREC, OUTREC, or SUM statement is specified, if you have an EFS01 or EFS02 routine, or if locale processing is used for SORT or MERGE fields. Note that none of these situations prevents the use of VLSCMP.
2. Unlike the OUTREC statement, the OUTREC parameter of the OUTFIL statement does not force NOVLSHRT. Thus, you can use VLSHRT with OUTFIL to eliminate records with the INCLUDE or OMIT parameter and reformat the remaining records with the OUTREC parameter. If a short OUTFIL OUTREC field is found, DFSORT terminates (even if VLSHRT is in effect) unless the VLFILL=byte parameter of OUTFIL is specified.
3. If VLSHRT is in effect and Blockset is selected:
   - DFSORT pads short SORT or MERGE control fields with binary zeros, thus making the order predictable for records with equal control fields of different lengths. The control fields are only padded temporarily for collation; they are not actually changed for output.

- Padding may increase the amount of work space required. If necessary, specify the file size to aid DFSORT's dynamic work space allocation and internal optimizations.

4. If VLSHRT is in effect and Blockset is not selected:
   - DFSORT terminates if the first byte of the first (major) SORT or MERGE control field is not included in the record.
   - DFSORT does not pad short SORT or MERGE 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 SORT or MERGE control fields.
   - EQUALS is not used.

   Note: You can use a SORTDIAG DD statement to force message ICE800I, which gives a code indicating why Blockset could not be used.
5. The way in which DFSORT processes short INCLUDE/OMIT compare fields depends on the settings for VLSCMP/NOVLSCMP and VLSHRT/NOVLSHRT. For details, see the discussion of the VLSCMP and NOVLSCMP options.

Default: Usually the installation default. See "Appendix B. Specification/Override of DFSORT Options" on page 549 for full override details.

Applicable Functions: See "Appendix B. Specification/Override of DFSORT Options" on page 549.

VSAMEMT or NVSAMEMT

```
►►──┬─VSAMEMT──┬──────────────────────────────────────────────────►◄
    └─NVSAMEMT─┘
```

Temporarily overrides the VSAMEMT installation option, which specifies whether DFSORT should accept an empty VSAM input data set.

VSAMEMT
specifies that DFSORT accepts an empty VSAM input data set and processes it as having zero records.

NVSAMEMT
specifies that DFSORT terminates if an empty VSAM input data set is found.

Default: Usually the installation default. See "Appendix B. Specification/Override of DFSORT Options" on page 549 for full override details.

Applicable Functions: See "Appendix B. Specification/Override of DFSORT Options" on page 549.

VSAMIO or NOVSAMIO

```
►►──┬─VSAMIO──┬──────────────────────────────────────────────────►◄
    └─NOVSAMIO─┘
```

Temporarily overrides the VSAMIO installation option, which specifies whether DFSORT should allow a VSAM data set defined with REUSE to be sorted in-place.

VSAMIO
specifies that DFSORT can use the same VSAM data set for input and output when all of the following conditions are met:

- The application is a sort.
- RESET is in effect.
- The VSAM data set was defined with REUSE.

These conditions ensure that the VSAM data set is processed as NEW for output and will contain the sorted input records, that is, it will be sorted in-place.

DFSORT terminates if the same VSAM data set is specified for input and output and any of the above conditions are not met.

NOVSAMIO
specifies that DFSORT terminates if the same VSAM data set is used for input and output.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

WRKREL or NOWRKREL

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

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

WRKREL
specifies that unused space is released.

NOWRKREL
specifies that unused space is not released.

Notes:

1. If you have dedicated certain volumes for SORTWKdd data sets, and you do not want unused temporary space to be released, you should specify NOWRKREL.
2. If WRKREL is in effect, DFSORT releases space for the SORTWKdd data sets just prior to termination. Space is released only for those SORTWKdd data sets that were used for the sort application.
3. RLS=0 can be used instead of NOWRKREL. RLS=n (n greater than 0) can be used instead of WRKREL.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

WRKSEC or NOWRKSEC

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

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

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

NOWRKSEC
specifies that automatic secondary allocation for temporary JCL SORTWKdd data sets is not used.

Note: SEC=0 can be used instead of NOWRKSEC. SEC=n (n greater than 0) can be used instead of WRKSEC.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Y2PAST

```
►►──Y2PAST=──┬─s─┬──────────────────────────────────────────────────►◄
             └─f─┘
```

Temporarily overrides the Y2PAST installation option, which specifies the sliding (s) or fixed (f) century window. The century window is used with DFSORT’s Y2 formats to correctly interpret two-digit year data values as four-digit year data values.

s specifies the number of years DFSORT is to subtract from the current year to set the beginning of the sliding century window. Since the Y2PAST value is subtracted from the current year, the century window slides as the current year changes. For example, Y2PAST=81 would set a century window of 1915-2014 in 1996 and 1916-2015 in 1997. s must be a value between 0 and 100.

f specifies the beginning of the fixed century window. For example, Y2PAST=1 would set a century window of 1962-2061. f must be a value between 1000 and 3000.

Note: CENTURY=value and CENTWIN=value can be used instead of Y2PAST=value.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

ZDPRINT or NZDPRINT

```
►►──┬─ZDPRINT──┬──────────────────────────────────────────────────────►◄
    └─NZDPRINT─┘
```

Temporarily overrides the ZDPRINT installation option, which specifies whether positive zoned-decimal (ZD) fields resulting from summing must be converted to printable numbers (that is, whether the zone of the last digit should be changed from a hexadecimal C to a hexadecimal F). See “SUM Control Statement” on page 274 for further details on the use of ZDPRINT and NZDPRINT.

ZDPRINT
means convert positive ZD summation results to printable numbers. For example, change hexadecimal F3F2C5 (prints as 32E) to F3F2F5 (prints as 325).

NZDPRINT
means do not convert positive ZD summation results to printable numbers.

Note: ZDPRINT=YES can be used instead of ZDPRINT. ZDPRINT=NO can be used instead of NZDPRINT.

Default: Usually the installation default. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Function: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Aliases for OPTION Statement Options

For compatibility reasons, the following OPTION statement options can be specified by using the aliases listed below. See the indicated OPTION statement options for complete details.

Table 23. Aliases for OPTION Statement Options

Alias	OPTION Statement Option
CAOUTREC	SOLRF
CENTURY=value	Y2PAST=value
CENTWIN=value	Y2PAST=value
CHKPT	CKPT
CORE=value	MAINSIZE=value
HIPRLIM=value	HIPRMAX=value
L5=value	AVGRLEN=value
NOSDB	SDB=NO
PRINT=value	MSGPRT=value
RLS=n	WRKREL
RLS=0	NOWRKREL
SDB	SDB=YES
SDB=SMALL	SDB=YES
SEC=n	WRKSEC
SEC=0	NOWRKSEC
SEQ=YES	VERIFY
SEQ=NO	NOVERIFY

Table 23. Aliases for OPTION Statement Options (continued)

Alias	OPTION Statement Option
ZDPRINT=YES	ZDPRINT
ZDPRINT=NO	NZDPRINT

Specifying DFSORT Options or COPY—Examples

Example 1

```
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 records that collate identically 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.

Example 2

```
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 the Blockset technique is used. 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. However, functions such as OUTFIL, which are supported only by the Blockset technique, cannot be used if the Checkpoint/Restart facility is used.

EQUALS
The sequence of records that collate identically 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
The record count is checked at the end of program processing.

Example 3

```
OPTION FILSZ=50,SKIPREC=5,DYNALLOC=3390
SORT  FIELDS=(1,2,CH,A),SKIPREC=1,SIZE=200,DYNALLOC=(3380,5)
```

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 3390.

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.

Example 4

```
OPTION NOBLKSET
```

NOBLKSET
DFSORT does not use the Blockset technique for a sort or merge.

Example 5

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

STOPAFT
DFSORT accepts 100 records before sorting or copying.

COBEXIT
COBOL E15 and E35 routines are run with either the VS COBOL II library or the Language Environment library.

Example 6

```
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.

Example 7

```
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, but must be coded as one contiguous statement without continuation lines.

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.

Example 8

```
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.

Example 9

```
SORT   FIELDS=(5,4,CH,A)
SUM FIELDS=(12,5,ZD,25,6,ZD)
OPTION ZDPRINT
```

ZDPRINT
The positive summed ZD values are printable because DFSORT uses an F sign for the last digit.

OUTFIL Control Statements

```
►►─OUTFIL──────────────────────────────────────────────────────────►

   ┌─,──────────────────────────────────────────────────────────┐
►──┴─FNAMES=──┬─ddname──────────────┬──────────────────────────┴──►◄
              │      ┌─,─────────┐  │
              └─(────┴─ddname─────┴──)─┘
  ├─FILES=────┬─suffix──────────────┬──────────────────────────┤
              │      ┌─,─────────┐  │
              └─(────┴─suffix─────┴──)─┘
  ├─STARTREC=n──────────────────────────────────────────────────┤
  ├─ENDREC=n────────────────────────────────────────────────────┤
  │ ┌─INCLUDE=──┬─(─logical expression─)─┬──────────────────────┤
  │ │           ├─ALL────────────────────┤                      │
  │ │           └─NONE───────────────────┘                      │
  │ ├─OMIT=─────┬─(─logical expression─)─┬──────────────────────┤
  │ │           ├─ALL────────────────────┤                      │
  │ │           └─NONE───────────────────┘                      │
  │ └─SAVE──────────────────────────────┘                       │
  ├─SPLIT───────────────────────────────────────────────────────┤
  │          ┌─,─────────┐                                      │
  ├─OUTREC=─(────┴─field─────┴──)──┬──────────────┬──┬──────────────┤
  │                                └─,CONVERT─────┘  └─,VLFILL=byte─┘
  ├─LINES=n─────────────────────────────────────────────────────┤
  │          ┌─,─────────┐                                      │
  ├─HEADER1=─(────┴─field─────┴──)──────────────────────────────┤
  │          ┌─,─────────┐                                      │
  ├─TRAILER1=─(────┴─field─────┴──)─────────────────────────────┤
  │          ┌─,─────────┐                                      │
  ├─HEADER2=─(────┴─field─────┴──)──────────────────────────────┤
  │          ┌─,─────────┐                                      │
  ├─TRAILER2=─(────┴─field─────┴──)─────────────────────────────┤
  │          ┌─,─────────┐                                      │
  ├─SECTIONS=─(────┴─field─────┴──)─────────────────────────────┤
  └─NODETAIL────────────────────────────────────────────────────┘
```

OUTFIL control statements allow you to create one or more output data sets for a sort, copy, or merge application from a single pass over one or more input data sets. You can use multiple OUTFIL statements, with each statement specifying the OUTFIL processing to be performed for one or more output data sets. OUTFIL processing begins after all other processing ends (that is, after processing for exits, options, and other control statements). OUTFIL statements support a wide variety of output data set tasks, including:

- Creation of multiple output data sets containing unedited or edited records from a single pass over one or more input data sets.

- Creation of multiple output data sets containing different ranges or subsets of records from a single pass over one or more input data sets. In addition, records that are not selected for any subset can be saved in a separate output data set.
- Conversion of variable-length record data sets to fixed-length record data sets.
- Insertion of blanks, zeros, strings and sequence numbers before, between, and after the input fields in the reformatted records.
- Sophisticated editing and conversion capabilities, such as hexadecimal display, conversion of numeric values from one format to another, and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.
- Transformation of two-digit year dates to various forms of four-digit year dates using a specified fixed or sliding century window.
- Selection of a character or hexadecimal string for output from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).
- Highly detailed three-level (report, page, and section) reports containing a variety of report elements you can specify (for example, current date, current time,page number, character strings, and blank lines) or derive from the input records (for example, character fields; edited numeric input fields; record counts; and edited totals, maximums, minimums, and averages for numeric input fields).
- Creation of multiple output records from each input record, with or without intervening blank output records.

The parameters of OUTFIL are grouped by primary purpose as follows:

- **FNAMES** and **FILES** specify theddnames of the **OUTFIL data sets** to be created. Each OUTFIL data set to be created must be specifically identified using FNAMES or FILES in an OUTFIL statement. By contrast, the **SORTOUT data set** is created by default if a DD statement for it is present. The term “SORTOUT data set” denotes the single non-OUTFIL output data set, but in fact, the SORTOUT ddname can be used for an OUTFIL data set either explicitly or by default.

  If SORTOUT is identified as an OUTFIL ddname, either explicitly(for example, via FILES=OUT) or by default (OUTFIL without FILES or FNAMES), the data set associated with the SORTOUT ddname will be processed as an OUTFIL data set rather than as the SORTOUT data set.

  OUTFIL data sets have characteristics and requirements similar to those for the SORTOUT data set, but there are differences in the way each is processed. The major differences are that an E39 exit routine is not entered for OUTFIL data sets, and that OUTFIL processing does not permit the use of the LRECL value to pad fixed-format OUTFIL records. (DFSORT will automatically determine and set an appropriate RECFM, LRECL, and BLKSIZE for each OUTFIL data set for which these attributes are not specified or available.)

  For a single DFSORT application, OUTFIL data sets can be intermixed with respect to VSAM and non-VSAM, tape and DASD, and so on. All of the data sets specified for a particular OUTFIL statement are processed in a similar way and thus are referred to as an **OUTFIL group**. (That is, you group OUTFIL data sets that use the same operands by specifying them on a single OUTFIL statement.) For example, the first OUTFIL statement might have an INCLUDE operand that applies to an OUTFIL group of one non-VSAM data set on DASD and another on

tape; a second OUTFIL statement might have OMIT and OUTREC operands that apply to an OUTFIL group of one non-VSAM data set on DASD and two VSAM data sets.

Records areprocessed for OUTFIL as they are for SORTOUT, after all other DFSORT processing is complete. Conceptually, you can think of an **OUTFIL input record** as being intercepted at the point between being passed from an E35 exit and written to SORTOUT, although neither an E35 exit nor SORTOUT need actually be specified with OUTFIL processing. With that in mind, see Figure 2 on page 8 for details on the processing that occurs prior to processing the OUTFIL input record. In particular:

– Records deleted by an E15 or E35 exit, an INCLUDE, OMIT or SUM statement, or the SKIPREC or STOPAFT parameter are not available for OUTFIL processing

– If records are reformatted by an E15 exit, an INREC or OUTREC statement, or an E35 exit, the resulting reformatted record is the OUTFIL input record to which OUTFIL fields must refer.

- **STARTREC** starts processing for an OUTFIL group at a specific OUTFIL input record. **ENDREC** ends processing for an OUTFIL group at a specific OUTFIL input record. Separately or together, STARTREC and ENDREC select a range of records to which subsequent OUTFIL processing will apply.
- **INCLUDE, OMIT**, and **SAVE** select the records to be included in the data sets of an OUTFIL group. INCLUDE and OMIT operate against the specified fields of each OUTFIL input record to select the output records for their OUTFIL group (all records are selected by default). SAVE selects the records that are not selected for any other OUTFIL group.

  Whereas the INCLUDE and OMIT statements apply to all input records, the INCLUDE and OMIT parameters apply only to the OUTFIL input records for their OUTFIL group. The INCLUDE and OMIT parameters have all of the logical expression capabilities of the INCLUDE and OMIT statements.
- **OUTREC** reformats the output records for an OUTFIL group. OUTREC enables you to rearrange, edit, and change the fields of the OUTFIL input records and to insert blanks, zeros, strings and sequence numbers.

  OUTREC also enables you to produce multiple reformatted output records from each input record, with or without intervening blank output records.

  OUTREC is used with **CONVERT** to change variable-length input records to fixed-length output records.

  VLFILL can be used to allow processing of variable-length input records which are too short to contain all specified OUTREC fields

  Whereas the OUTREC statement applies to all input records, the OUTREC parameter applies only to the OUTFIL input records for its OUTFIL group. In addition, the OUTREC parameter supports the forward slash (/) separator for creating blank records and new records, whereas the OUTREC statement does not.
- **LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS**, and **NODETAIL** indicate that a report is to be produced for an OUTFILgroup, and specify the details of the **report records** to be produced for the report. Reports can contain report records for a report header (first page), report trailer (last page), page header and page trailer (at the top and bottom of each page, respectively), and section headers and trailers (before and after each section, respectively).

  Data records for the report result from the inclusion of OUTFIL input records. All of the capabilities of the OUTREC parameter are available to create reformatted data records from the OUTFIL input records. Each set of sequential OUTFIL

input records, with the same binary value for a specified field, results in a corresponding set of data records that is treated as a section in the report.

The length for the data records must be equal to or greater than the maximum report record length. OUTFIL data sets used for reports must have or will be given ASA control character format ('A' as in, for example, RECFM=FBAor RECFM=VBA), and must allow an extra byte in the LRECL for the carriage control character that DFSORT will add to each report and data record. DFSORT uses these carriage control characters to control page ejects and the placement of the lines in your report according to your specifications. DFSORT uses appropriate carriage controls (for example, C'-' for triple space) in header and trailer records when possible, to reduce the number of report records written. DFSORT always uses the single space carriage control (C' ') in data records. Although these carriage control characters may not be shown when you view an OUTFIL data set (depending upon how you view it), they will be used if you print the report.

- **SPLIT** splits the output records in rotation among the data sets of an OUTFIL group. The first output record is written to the first OUTFIL data set in the group, the second output record is written to the second data set, and so on; when each OUTFIL data set has one record, the rotation starts again with the first OUTFIL data set.
- Figure 15 on page 186 illustrates the order in which OUTFIL records and parameters are processed.

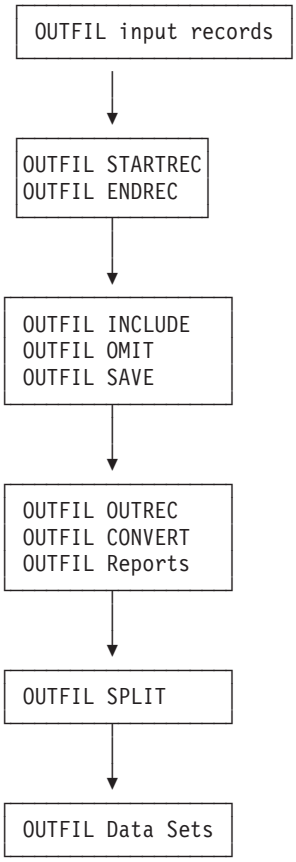


Figure 15. OUTFIL Processing Order

Notes:

1. DFSORT accepts but does not process the following OUTFIL operands:BLKSIZE=value, BUFLIM=value, BUFOFF=value, CARDS=value, CLOSE=value, DISK, ESDS, EXIT, FREEOUT, KSDS, LRECL=value, NOTPMK, OPEN=value, OUTPUT, PAGES=value, PRINT, PUNCH, REUSE, RRDS, SPAN, SYSLST, TAPE, and TOL.
2. Sample syntax is shown throughout this section. Complete OUTFIL statement examples are shown and explained under “OUTFIL Features—Examples” on page 239.

FNAMES

```
►►──FNAMES=──┬─ddname──────────────┬──────────────────────────────────►◄
             │   ┌──────────────┐  │
             │   ↓,             │  │
             └─(─────ddname─────┴─)─┘
```

Specifies ddnames associated with the OUTFIL data sets for this OUTFIL statement. The ddnames specified using the FNAMES and FILES parameters constitute the output data sets for this OUTFIL group to which all of the other parameters for this OUTFIL statement apply.

If FNAMES specifies the ddname in effect for the SORTOUT data set (that is, whichever is in effect among SORTOUT, name from SORTOUT=name, or ccccOUT from SORTDD=cccc), DFSORT will treat the data set associated with that ddname as an OUTFIL data set rather than as the SORTOUT data set.

ddname
specifies a 1- through 8-character ddname. A DD statement must be present for this ddname.

Sample Syntax:

```
OUTFIL FNAMES=(OUT1,OUT2,PRINTER,TAPE)
OUTFIL FNAMES=BACKUP
```

Default for FNAMES: If neither FNAMES nor FILES is specified for an OUTFIL statement, the default ddname is SORTOUT or ccccOUT if SORTDD=cccc is in effect.

FILES

```
►►─FILES=─┬─d──────────────┬──────────────────────────────────────►◄
          ├─dd─────────────┤
          ├─OUT────────────┤
          │   ┌─,──────┐   │
          │   ▼        │   │
          └─(──┬─d──┬──┴─)─┘
               ├─dd─┤
               └─OUT┘
```

Specifies suffixes for ddnames to be associated with the OUTFIL data sets for this OUTFIL statement. The ddnames specified using the FNAMES and FILES parameters constitute the output data sets for this OUTFIL group to which all of the other parameters for this OUTFIL statement apply.

If FILES specifies the ddname in effect for the SORTOUT data set (that is, whichever is in effect among SORTOUT, name from SORTOUT=name, or ccccOUT from SORTDD=cccc), DFSORT will treat the data set associated with that ddname as an OUTFIL data set rather than as the SORTOUT data set.

d specifies the 1-character suffix to be used to form the ddname SORTOFd or ccccOFd if SORTDD=cccc is in effect. A DD statement must be present for this ddname.

dd specifies the 2-character suffix to be used to form the ddname SORTOFdd or ccccOFdd if SORTDD=cccc is in effect. A DD statement must be present for this ddname.

OUT
specifies the suffix OUT is to be used to form the ddname SORTOUT or ccccOUT if SORTDD=cccc is in effect. A DD statement must be present for this ddname.

Sample Syntax:

```
OUTFIL FILES=(1,2,PR,TP)
OUTFIL FILES=OUT
```

Default for FILES: If neither FNAMES nor FILES is specified for an OUTFIL statement, the default ddname is SORTOUT or ccccOUT if SORTDD=cccc is in effect.

STARTREC

```
►►──STARTREC=n──────────────────────────────────────────────────────────►◄
```

Specifies the OUTFIL input record at which OUTFIL processing is to start for this OUTFIL group. OUTFIL input records before this starting record are not included in the data sets for this OUTFIL group.

n specifies the relative record number. The value for n starts at 1 (the first record) and is limited to 28 digits (15 significant digits).

Sample Syntax:

```
OUTFIL FNAMES=SKIP20,STARTREC=21
```

Default for STARTREC: 1.

ENDREC

```
►►──ENDREC=n────────────────────────────────────────────────────────────►◄
```

Specifies the OUTFIL input record at which OUTFIL processing is to end for this OUTFIL group. OUTFIL input records after this ending record are not included in the data sets for this OUTFIL group.

The ENDREC value must be equal to or greater than the STARTREC value if both are specified on the same OUTREC statement.

n specifies the relative record number. The value for n starts at 1 (the first record) and is limited to 28 digits (15 significant digits).

Sample Syntax:

```
OUTFIL FNAMES=TOP10,ENDREC=10
OUTFIL FNAMES=FRONT,ENDREC=500
OUTFIL FNAMES=MIDDLE,STARTREC=501,ENDREC=2205
OUTFIL FNAMES=BACK,STARTREC=2206
```

Default for ENDREC: The last OUTFIL input record.

INCLUDE

```
►►──INCLUDE=──┬─(logical expression)─┬──────────────────────────────────►◄
              ├─ALL──────────────────┤
              ├─(ALL)────────────────┤
              ├─NONE─────────────────┤
              └─(NONE)───────────────┘
```

Selects the records to be included in the data sets for this OUTFIL group.

The INCLUDE parameter operates in the same way as the INCLUDE statement, except that:

- The INCLUDE statement applies to all input records; the INCLUDE parameter applies only to the OUTFIL input records for its OUTFIL group.

- FORMAT=f can be specified with the INCLUDE statement, but not with the INCLUDE parameter.
- D2 format can be specified with the INCLUDE statement, but not with the INCLUDE parameter.

See “INCLUDE Control Statement” on page 93 for complete details.

logical expression
specifies one or more relational conditions logically combined based on fields in the OUTFIL input record. If the logical expression is true for a given record, the record is included in the data sets for this OUTFIL group.

ALL or (ALL)
specifies that all of the OUTFIL input records are to be included in the data sets for this OUTFIL group.

NONE or (NONE)
specifies that none of the OUTFIL input records are to be included in the data sets for this OUTFIL group.

Sample Syntax:

```
OUTFIL FNAMES=J69,INCLUDE=(5,3,CH,EQ,C'J69')
OUTFIL FNAMES=J82,INCLUDE=(5,3,CH,EQ,C'J82')
```

Default for INCLUDE: ALL.

OMIT

```
►►──OMIT=──┬─(logical expression)─┬──────────────────────────────────────►◄
           ├─ALL──────────────────┤
           ├─(ALL)────────────────┤
           ├─NONE─────────────────┤
           └─(NONE)───────────────┘
```

Selects the records to be omitted from the data sets for this OUTFIL group.

The OMIT parameter operates in the same way as the OMIT statement, except that:

- The OMIT statement applies to all input records; the OMIT parameter applies only to the OUTFIL input records for its OUTFIL group.
- FORMAT=f can be specified with the OMIT statement, but not with the OMIT parameter.
- The D2 format can be specified with the OMIT statement, but not with the OMIT parameter.

See “OMIT Control Statement” on page 130 and “INCLUDE Control Statement” on page 93 for complete details.

logical expression
specifies one or more relational conditions logically combined based on fields in the OUTFIL input record. If the logical expression is true for a given record, the record is omitted from the data sets for this OUTFIL group.

ALL or (ALL)
specifies that all of the OUTFIL input records are to be omitted from the data sets for this OUTFIL group.

NONE or (NONE)

specifies that none of the OUTFIL input records are to be omitted from the data sets for this OUTFIL group.

Sample Syntax:

```
OUTFIL FILES=01,OMIT=NONE
OUTFIL OMIT=(5,1,BI,EQ,B'110.....')
OUTFIL FNAMES=(OUT1,OUT2),
  OMIT=(7,2,CH,EQ,C'32',OR,18,3,CH,EQ,C'XYZ')
```

Default for OMIT: NONE.

SAVE

```
►►──SAVE──────────────────────────────────────────────────────────────────►◄
```

Specifies that OUTFIL input records not included by INCLUDE or OMIT for any other OUTFIL group are to be included in the data sets for this OUTFIL group. SAVE operates in a global fashion over all of the other OUTFIL statements for which SAVE is not specified, enabling you to keep any OUTFIL input records that would not be kept otherwise. SAVE will include the same records for each group for which it is specified.

Sample Syntax:

```
OUTFIL INCLUDE=(8,6,CH,EQ,C'ACCTNG'),FNAMES=GP1
OUTFIL INCLUDE=(8,6,CH,EQ,C'DVPMNT'),FNAMES=GP2
OUTFIL SAVE,FNAMES=NOT1OR2
```

Default for SAVE: None; must be specified.

SPLIT

```
►►──SPLIT─────────────────────────────────────────────────────────────────►◄
```

Splits the output records in rotation among the data sets of this OUTFIL group. Thus for an OUTFIL group with n data sets, the first OUTFIL data set in the group will receive records 1, 1+n, 1+2n, ..., the second data set will receive records 2, 2+n, 2+2n, ..., and so on for each data set in the group, until all of the output records have been written. As a result, the records will be split as evenly as possible among all of the data sets in the group.

The SPLIT parameter cannot be used with any of the report parameters (LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, and NODETAIL) since it doesn't make sense to split up the records of a report.

Sample Syntax:

```
OUTFIL FNAMES=(PIPE1,PIPE2,PIPE3,PIPE4),SPLIT
OUTFIL FNAMES=(TAPE1,TAPE2),SPLIT,
    INCLUDE=(8,2,ZD,EQ,27),OUTREC=(5X,1,75)
```

Default for SPLIT: None; must be specified.

OUTREC

```
                  ┌─,──────────────────────────────────┐
                  ▼                                    │
►►──OUTREC=─(──────┬──────┬──┬─s──────────────────────┬──)──────────────────────────►◄
                   └─c:───┘  ├─p,m──────────────────┤
                             │      └─,a──┘          │
                             ├─p──────────────────────┤
                             ├─p,m,HEX────────────────┤
                             ├─p,HEX──────────────────┤
                             ├─p,m,Y2x────────────────┤
                             ├─p,m,Y2x(c)─────────────┤
                             ├─p,m,Y2xP───────────────┤
                             ├─p,m,f──────────────────┤
                             │      └─,edit─┘         │
                             ├─p,m,f,to───────────────┤
                             ├─p,m,Y2x,edit───────────┤
                             ├─p,m,Y2x,to─────────────┤
                             ├─p,m,lookup─────────────┤
                             └─seqnum─────────────────┘
```

Specifies how the records in the data sets for this OUTFIL group are to be reformatted. OUTREC can define which parts of the OUTFIL input record are included in the reformatted OUTFIL output record, in what order the parts appear, how they are aligned, and how they are edited or changed. You can also insert separators and sequence numbers before, between, and after the input fields, and produce multiple reformatted output records for each input record, with or without intervening blank output records

You can use the OUTREC parameter in conjunction with the CONVERT parameter to convert variable-length record data sets to fixed-length record data sets.

You can use the VLFILL parameter to allow processing of variable-length input records which are too short to contain all specified OUTREC fields.

The OUTREC parameter can be used with any or all ofthe report parameters (LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, and NODETAIL) to produce reports. The report parameters specify the report records to be produced, while the OUTREC parameter specifies the reformatted data records to be produced. DFSORT uses ASA carriage control characters to control page ejects and the placement of the lines in your report, according to your specifications.

When you create an OUTFIL report, the length for the longest or only data record must be equal to or greater than the maximum report record length. You can use the OUTREC parameter to force a length for the data records that is longer than any report record; you can then either let DFSORT compute and set the LRECL, or ensure that the computed LRECL is equal to the existing or specified LRECL. Remember to allow an extra byte in the LRECL for the ASA carriage control character.

For example, if your data records are 40 bytes, but your longest report record is 60 bytes, you can use an OUTREC parameter such as:

```
OUTREC=(1,40,80:X)
```

DFSORT will then set the LRECL to 81 (1 byte for the ASA carriage control character plus 80 bytes for the length of the data records), and pad the data records with blanks on the right.

Note: The OUTREC statement applies to all input records, whereas the OUTREC parameter of the OUTFIL statement applies only to the OUTFIL input records for its OUTFIL group. The OUTREC parameter of the OUTFIL statement supports the slash (/) separator for creating blank records and new records; the OUTREC statement does not.

You can choose to include any or all of the following in your reformatted OUTFIL output records:

- Blanks, binary zeros, character strings, and hexadecimal strings
- Unedited input fields aligned on byte, halfword, fullword, and doubleword boundaries
- Hexadecimal representations of binary input fields
- Numeric input fields of various formats converted to different numeric formats, or to character format edited to contain signs, thousands separators, decimal points, leading zeros or no leading zeros, and so on.
- Two-digit year input dates of various formats converted to four-digit year dates in different numeric formats, or to character format edited to contain separators, leading zeros or no leading zeros, and so on.
- Sequence numbers in various formats.
- Character or hexadecimal strings from a lookup table.

The reformatted OUTFIL output record consists of the separation fields, edited and unedited input fields and sequence numbers you select, in the order in which you select them, aligned on the boundaries or in the columns you indicate.

c: specifies the position (column) for a separation field, input field or sequence number, relative to the start of the reformatted OUTFIL output record. Count the RDW (variable-length output records only) but not the carriage control character (for reports) when specifying c:. That is, 1: indicates the first byte of the data in fixed-length output records and 5: indicates the first byte of the data in variable-length output records.

Unused space preceding the specified column is padded with EBCDIC blanks. The following rules apply:

- c must be a number between 1 and 32752.
- 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 OUTFIL 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 OUTFIL input record.
- The colon (:) is treated like the comma (,) or semicolon (;) for continuation to another line.

See Table 16 on page 114 for examples of valid and invalid column alignment.

s specifies that a separation field is to appear in the reformatted OUTFIL output record, or that a new output record is to be started, with or without intervening blank output records. These separation elements (separation

fields, new record indicators and blank record indicators) can be specified before or after any input field. Consecutive separation elements may be specified. For variable-length records, separation elements must not be specified before the first input field (the record descriptor word) or after the variable part of the OUTFIL input record. Permissible values are nX, nZ, nC'xx...x', nX'yy...yy', /.../ and n/.

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

See Table 17 on page 114 for examples of valid and invalid blank separation.

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

See Table 18 on page 115 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 to appear in the reformatted OUTFIL output records. n can range 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'
```

See Table 19 on page 115 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 to appear in the reformatted OUTFIL output records. n can range 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 20 on page 116 for examples of valid and invalid hexadecimal string separation.

/.../ or n/
Blank records or a new record. A new output record is to be started with or without intervening blank output records. If /.../ or n/ is specified at the beginning or end of OUTREC, n blank output records are to be produced. If /.../ or n/ is specified in the middle of OUTREC, n-1 blank output records are to be produced (thus, / or 1/ indicates a new output record with no intervening blank output records).

At least one input field or separation field must be specified if you use /.../ or n/. For example, OUTREC=(//) is not allowed, whereas OUTREC=(//X) is allowed.

Either n/ (for example, 5/) or multiple /'s (for example, /////) can be used. n can range from 1 to 255. If n is omitted, 1 is used.

As an example, if you specify:

```
OUTFIL OUTREC=(2/,C'Field 2 contains ',4,3,/,
                  C'Field 1 contains ',1,3)
```

an input record containing:

```
111222
```

would produce the following four output records:

```
Blanks
Blanks
Field 2 contains 222
Field 1 contains 111
```

Note that **four** OUTFIL output records are produced for **each** OUTFIL input record.

p,m,a
specifies that an unedited input field is to appear in the reformatted OUTFIL output record.

p specifies the first byte of the input field relative to the beginning of the OUTFIL input record. 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 can extend beyond byte 32752. See “OUTFIL Statements Notes” on page 236 for special rules concerning variable-length records.

m specifies the length in bytes of the input field.

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

The permissible values of **a** are:

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

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

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

Alignment can be necessary if, for example, the data is 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.

p specifies the unedited variable part of the OUTFIL input record (that part beyond the minimum record length) is to appear in the reformatted OUTFIL output record as the last field. Note that if the reformatted OUTFIL record includes only the RDW and the variable part of the OUTFIL input record, “ null” records containing only an RDW may result.

A value must be specified for p that is less than or equal to the minimum OUTFIL input record length plus 1 byte.

p,m,HEX
specifies that the hexadecimal representation of an input field is to appear in the reformatted OUTFIL output record.

p See p under p,m,a.

m specifies the length in bytes of the input field. The value for m must be 1 to 16376.

HEX
requests hexadecimal representation of the input field. Each byte of the input field is replaced by its two-byte equivalent. For example, the characters AB would be replaced by C1C2.

p,HEX
specifies that the hexadecimal representation of the variable part of the OUTFIL input record (that part beyond the minimum record length) is to appear in the reformatted OUTFIL output record as the last field. Note that if the reformatted OUTFIL record includes only the RDW and the variable part of the OUTFIL input record, “null” records containing only an RDW may result.

p A value must be specified for p that is less than or equal to the minimum record length plus 1 byte.

HEX
requests hexadecimal representation of the variable part of the OUTFIL input record. Each byte of the input field is replaced by its two-byte equivalent. For example, the characters AB would be replaced by C1C2.

Sample Syntax:

```
Fixed input records:
 OUTFIL FNAMES=(OUT1,OUT2),
         OUTREC=(1:5,10,15:8C’0’,25:20,15,80:X)

Variable input records:
 OUTFIL OUTREC=(1,4,C’ RDW=’,1,3,HEX,C’ FIXED=’,
                5,20,HEX,C’ VARIABLE=’,21,HEX)
```

p,m,Y2x
specifies that the four-digit year CH date representation of a two-digit year input date field is to appear in the reformatted OUTFIL output record. Real dates are transformed using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2T transforms C'000000' to C'00000000').

p See p under p,m,a.

m specifies the length in bytes of the two-digit year date field.

Y2x
specifies the Y2 format. See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

Sample Syntax:

```
OUTFIL OUTREC=(21,3,Y2V,X,12,5,Y2W)
```

Table 24 shows the output produced for each type of date.

Table 24. p,m,Y2x Output

Type of Date	Fields (m,f)		Output for p,m,Y2x
yyx	3,Y2T	2,Y2U	C'yyyyx'
yyxx	4,Y2T	3,Y2V	C'yyyyxx'
yyxxx	5,Y2T	3,Y2U	C'yyyyxxx'
yyxxxx	6,Y2T	4,Y2V	C'yyyyxxxx'
xyy	3,Y2W	2,Y2X	C'xyyyy'
xxyy	4,Y2W	3,Y2Y	C'xxyyyy'
xxxyy	5,Y2W	3,Y2X	C'xxxyyyy'
xxxxyy	6,Y2W	4,Y2Y	C'xxxxyyyy'
yy	2,Y2C	2,Y2Z	C'yyyy'
yy	2,Y2S	2,Y2P	C'yyyy'
yy	1,Y2D	1,Y2B	C'yyyy'

p,m,Y2x(c)

specifies that the four-digit year CH date representation with separators of a two-digit year input date field is to appear in the reformatted OUTFIL output record. Real dates are transformed using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2T(/) transforms C'000000' to C'0000/00/00').

p See p under p,m,a.

m specifies the length in bytes of the two-digit year date field.

Y2x
specifies the Y2 format. See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

c specifies the separator character. c can be any character *except* a blank.

Sample Syntax:

```
OUTFIL OUTREC=(25,6,Y2T(-),X,14,2,Y2U(/))
```

Table 25 shows the output produced for each type of Y2x(c) date field when / is used for c.

Table 25. p,m,Y2x(c) Output

Type of Date	Fields (m,f)		Output for p,m,Y2x(/)
yyx	3,Y2T	2,Y2U	C'yyyy/x'
yyxx	4,Y2T	3,Y2V	C'yyyy/xx'
yyxxx	5,Y2T	3,Y2U	C'yyyy/xxx'
yyxxxx	6,Y2T	4,Y2V	C'yyyy/xx/xx'
xyy	3,Y2W	2,Y2X	C'x/yyyy'
xxyy	4,Y2W	3,Y2Y	C'xx/yyyy'
xxxyy	5,Y2W	3,Y2X	C'xxx/yyyy'
xxxxyy	6,Y2W	4,Y2Y	C'xx/xx/yyyy'

p,m,Y2xP

specifies that the four-digit year PD date representation of a two-digit year input date field is to appear in the reformatted OUTFIL output record. Real dates are transformed using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2TP transforms C'000000' to P'00000000'):

p See p under p,m,a.

m specifies the length in bytes of the two-digit year date field.

Y2xP

specifies the Y2 format. See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

Sample Syntax:

```
OUTFIL OUTREC=(11,3,Y2XP,X,21,4,Y2WP)
```

Table 26 shows the output produced for each type of date.

Table 26. p,m,Y2xP Output

Type of Date	Fields (m,f)		Output for p,m,Y2xP
yyx	3,Y2TP	2,Y2UP	P'yyyyx'
yyxx	4,Y2TP	3,Y2VP	P'yyyyxx'
yyxxx	5,Y2TP	3,Y2UP	P'yyyyxxx'
yyxxxx	6,Y2TP	4,Y2VP	P'yyyyxxxx'
xyy	3,Y2WP	2,Y2XP	P'xyyyy'
xxyy	4,Y2WP	3,Y2YP	P'xxyyyy'
xxxyy	5,Y2WP	3,Y2XP	P'xxxyyyy'
xxxxyy	6,Y2WP	4,Y2YP	P'xxxxyyyy'
yy	2,Y2PP		P'yyyy'
yy	1,Y2DP		X'yyyy'

p,m,f,edit

specifies that an edited numeric input field is to appear in the reformatted OUTFIL output record. You can edit BI, FI, PD, PD0, ZD, and CSF/FS fields using either pre-defined edit masks (M0-M25) or specific edit patterns you define. You can control the way the edited fields look with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on.

p See p under p,m,a.

m specifies the length in bytes of the numeric field. The length must include the sign, if the data is signed. See Table 27 on page 198 for permissible length values.

f specifies the format of the numeric field:

Table 27. Edit Field Formats and Lengths

Format Code	Length	Description
BI	1 to 4 bytes	Unsigned binary
FI	1 to 4 bytes	Signed fixed-point
PD	1 to 8 bytes	Signed packed decimal
PD0	2 to 8 bytes	Packed decimal with sign and first digit ignored
ZD	1 to 15 bytes	Signed zoned decimal
CSF or FS	1 to 16 bytes (15-digit limit)	Signed numeric with optional leading floating sign

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

For a CSF/FS format field:

- A maximum of 15 digits is allowed. If a CSF/FS value with 16 digits is found, the leftmost digit will be treated as a positive sign indicator.

For a ZD or PD format field:

- An invalid digit results in a data exception (0C7 ABEND) or incorrect numeric output; A-F are invalid digits. ICETOOL’s VERIFY or DISPLAY operator can be used to identify decimal values with invalid digits.
- A value is treated as positive if its sign is F, E, C, A, 8, 6, 4, 2, or 0.
- A value is treated as negative if its sign is D, B, 9, 7, 5, 3, or 1.

For a PD0 format field:

- The first digit is ignored.
- An invalid digit other than the first results in a data exception (0C7 ABEND) or incorrect numeric output; A-F are invalid digits.
- The sign is ignored and the value is treated as positive.

edit

```
►►──┬─Mn──────────────────────────┬──────────────────────────────────────────────►◄
    │ ┌─EDIT=──┬──────┬─(pattern)─┬─┐
    └─┴─EDxy=──┘      └─('pattern')─┘

►►──┬──────────────────────────────────────────┬──┬──────────────────────────┬──►◄
    │ ┌──────────────────────────────────────┐ │  │ ┌─LENGTH=──┬─n───────┬─┐ │
    └─┴─,──┬─SIGNS=──┬──(lp,ln,tp,tn)──────┬─┴─┘  └─┴──────────┘   └─(n)───┘─┘ │
           └─SIGNz=──┘                     └─┘
```

Specifies how the numeric field is to be edited for output. If an Mn, EDIT, or EDxy parameter is not specified, the numeric field is edited using the M0 edit mask.

Mn

specifies one of 26 pre-defined edit masks (M0-M25) for presenting numeric data. If these pre-defined edit masks are not suitable for presenting your numeric data, the EDIT parameter gives you the flexibility to define your own edit patterns.

The 26 pre-defined edit masks can be represented as follows:

Table 28. Edit Mask Patterns

Mask	Pattern	Examples	
		Value	Result
M0	IIIIIIIIIIIIITS	+01234	1234
		-00001	1-
M1	TTTTTTTTTTTTTTTS	-00123	00123-
		+00123	00123
M2	I,III,III,III,IIT.TTS	+123450	1,234.50
		-000020	0.20-
M3	I,III,III,III,IIT.TTCR	-001234	12.34CR
		+123456	1,234.56
M4	SI,III,III,III,IIT.TT	+0123456	+1,234.56
		-1234567	-12,345.67
M5	SI,III,III,III,IIT.TTS	-001234	(12.34)
		+123450	1,234.50
M6	III-TTT-TTTT	00123456	012-3456
		12345678	1-234-56788
M7	TTT-TT-TTTT	00123456	000-12-3456
		12345678	012-34-5678
M8	IT:TT:TT	030553	3:05:53
		121736	12:17:36
M9	IT/TT/TT	123094	12/30/94
		083194	8/31/94
M10	IIIIIIIIIIIIIT	01234	1234
		00000	0
M11	TTTTTTTTTTTTTTT	00010	00010
		01234	01234
M12	SIII,III,III,III,IIT	+1234567	1,234,567
		-0012345	-12,345
M13	SIII.III.III.III.IIT	+1234567	1.234.567
		-0012345	-12.345
M14	SIII III III III IITS	+1234567	1 234 567
		-0012345	(12 345)
M15	III III III III IITS	+1234567	1 234 567
		-0012345	12 345-
M16	SIII III III III IIT	+1234567	1 234 567
		-0012345	-12 345
M17	SIII’III’III’III’IIT	+1234567	1’234’567
		-0012345	-12’345
M18	SI,III,III,III,IIT.TT	+0123456	1,234.56
		-1234567	-12,345.67

Table 28. Edit Mask Patterns (continued)

Mask	Pattern	Examples	
		Value	Result
M19	SI.III.III.III.IIT,TT	+0123456	1.234,56
		-1234567	-12.345,67
M20	SI III III III IIT,TTS	+0123456	1 234,56
		-1234567	(12 345,67)
M21	I III III III IIT,TTS	+0123456	1 234,567
		-1234567	12 345,67-
M22	SI III III III IIT,TT	+0123456	1 234,56
		-1234567	-12 345,67
M23	SI'III'III'III'IIT.TT	+0123456	1'234.56
		-1234567	-12'345.67
M24	SI'III'III'III'IIT,TT	+0123456	1'234,56
		-1234567	-12'345,67
M25	SIIIIIIIIIIIIIT	+01234	1234
		-00001	-1

The elements used in the representation of the edit masks in Table 28 on page 199 are as follows:

- **I** indicates a leading insignificant digit. If zero, this digit will not be shown.
- **T** indicates a significant digit. If zero, this digit will be shown.
- **CR** (in M3) is printed to the right of the digits if the value is negative; otherwise, two blanks are printed to the right of the digits.
- **S** indicates a sign. If it appears as the first character in the pattern, it is a leading sign. If it appears as the last character in the pattern, it is a trailing sign. If S appears as both the first and last characters in a pattern (example: M5), the first character is a leading sign and the last character is a trailing sign. Four different sign values are used: leading positive sign (lp), leading negative sign (ln), trailing positive sign (tp) and trailing negative sign (tn). Their applicable values for the Mn edit masks are:

Table 29. Edit Mask Signs

Mask	lp	ln	tp	tn
M0	none	none	blank	-
M1	none	none	blank	-
M2	none	none	blank	-
M3	none	none	none	none
M4	+	-	none	none
M5	blank	(	blank	)
M6	none	none	none	none
M7	none	none	none	none
M8	none	none	none	none
M9	none	none	none	none
M10	none	none	none	none

Table 29. Edit Mask Signs (continued)

Mask	lp	ln	tp	tn
M11	none	none	none	none
M12	blank	-	none	none
M13	blank	-	none	none
M14	blank	(	blank	)
M15	none	none	blank	-
M16	blank	-	none	none
M17	blank	-	none	none
M18	blank	-	none	none
M19	blank	-	none	none
M20	blank	(	blank	)
M21	none	none	blank	-
M22	blank	-	none	none
M23	blank	-	none	none
M24	blank	-	none	none
M25	blank	-	none	none

- any other character (for example, /) will be printed as shown, subject to certain rules to be subsequently discussed.

The implied length of the edited output field depends on the number of digits and characters needed for the pattern of the particular edit mask used. The LENGTH parameter can be used to change the implied length of the edited output field.

The number of digits needed depends on the format and length of the numeric field as follows:

Table 30. Digits Needed for Numeric Fields

Format	Input Length	Digits Needed
ZD	m	m
PD	m	2m-1
PD0	m	2m-2
BI, FI	1	3
BI, FI	2	5
BI, FI	3	8
BI, FI	4	10
CSF or FS	16	15
CSF or FS	m (less than 16)	m

The length of the output field can be represented as follows for each pattern, where d is the number of digits needed, as shown in Table 30, and the result is rounded down to the nearest integer:

Table 31. Edit Mask Output Field Lengths

Mask	Output Field Length	Example	
		Input (f,m)	Output Length
M0	d + 1	ZD,3	4
M1	d + 1	PD,5	10
M2	d + 1 + d/3	BI,4	14
M3	d + 2 + d/3	ZD,6	10
M4	d + 1 + d/3	PD,8	21
M5	d + 2 + d/3	FI,3	12
M6	12	ZD,10	12
M7	11	PD,5	11
M8	8	ZD,6	8
M9	8	PD,4	8
M10	d	BI,1	3
M11	d	PD,5	9
M12	d + 1 + (d - 1)/3	PD,3	7
M13	d + 1 + (d - 1)/3	FS,5	7
M14	d + 2 + (d - 1)/3	ZD,5	8
M15	d + 1 + (d - 1)/3	FI,3	11
M16	d + 1 + (d - 1)/3	ZD,6	8
M17	d + 1 + (d - 1)/3	FI,4	14
M18	d + 1 + d/3	BI,4	14
M19	d + 1 + d/3	PD,8	21
M20	d + 2 + d/3	FI,3	12
M21	d + 1 + d/3	ZD,3	5
M22	d + 1 + d/3	BI,2	7
M23	d + 1 + d/3	PD,6	15
M24	d + 1 + d/3	ZD,9	13
M25	d + 1	CSF,16	16

To illustrate conceptually how DFSORT produces the edited output from the numeric value, consider the following example:

```
OUTFIL OUTREC=(5,7,ZD,M5)

with ZD values of C'0123456'(+0123456)
and C'000302J' (-0003021)
```

As shown in the preceding tables, it is determined that:

- The general pattern for M5 is: SI,III,...,IIT.TTS
- The signs to be used are blank for leading positive sign, C'(' for leading negative sign, blank for trailing positive sign and C')' for trailing negative sign
- The number of digits needed is 7
- The length of the output field is 11 (7 + 2 + 7/3)

- The specific pattern for the output field is thus: C'SII,IIT.TTS'

The digits of C'0123456' are mapped to the pattern, resulting in C'S01,234.56S'. Since the value is positive, the leading sign is replaced with blank and the trailing sign is replaced with blank, resulting in C' 01,234.56 '. Finally, all digits before the first non-zero digit (1 in this case), are replaced with blanks, resulting in the final output of C'   1,234.56 '.

The digits of C'000302J' are mapped to the pattern, resulting in C'S00,030.21S'. Since the value is negative, the leading sign is replaced with C'(' and the trailing sign is replaced with C')' resulting in C'(00,030.21)'. All digits before the first non-zero digit (3 in this case), are replaced with blanks, resulting in C'(30.21)'. Finally, the leading sign is ″floated″ to the right, next to the first non-zero digit, resulting in the final output of C'   (30.21)'.

To state the rules in more general terms, the steps DFSORT takes conceptually to produce the edited output from the numeric value are as follows:

- Determine the specific pattern and its length, using the preceding tables.
- Map the digits of the numeric value to the pattern.
- If the value is positive, replace the leading and trailing signs (if any) with the characters for positive values shown in Table 29 on page 200. Otherwise, replace the leading and trailing signs (if any) with the characters for negative values shown in that same table.
- Replace all digits before the first non-zero (I) or significant digit (T) with blanks.
- Float the leading sign (if any) to the right, next to the first non-zero (I) or significant digit (T).

The following additional rule applies to edit masks:

- The specific pattern is determined from the general pattern by including signs, the rightmost digits needed as determined from the input format and length, and any characters in between those rightmost digits. This may unintentionally truncate significant digits (T). As an example, if you specify 5,1,ZD,M4, the length of the output field will be 2 (1 + 1 + 1/3). The general pattern for M4 is SI,III,...,IIT.TT, but the specific pattern will be ST (the leading sign and the rightmost digit).

EDIT

specifies an edit pattern for presenting numeric data. If the pre-defined edit masks (M0-M25) are not suitable for presenting your numeric data, EDIT gives you the flexibility to define your own edit patterns. The elements you use to specify the pattern are the same as those used for the edit masks: I, T, S, and printable characters. However, S will not be recognized as a sign indicator unless the SIGNS parameter is also specified.

pattern

specifies the edit pattern to be used. Not enclosing the pattern in single apostrophes restricts you from specifying the following characters in the pattern: blank, apostrophe, unbalanced left or right parentheses, and hexadecimal digits 20, 21, and 22. For example, EDIT=((IIT.TT)) is valid, whereas EDIT=(C)ITT.TT), EDIT=(I / T) and EDIT=(S'II.T) are not.

The maximum number of digits (I's and T's) you specify in the pattern must not exceed 15. The maximum length of the pattern must not exceed 22 characters.

'pattern'
specifies the edit pattern to be used. Enclosing the pattern in single apostrophes allows you to specify any character in the pattern except hexadecimal digits 20, 21, or 22. If you want to include a single apostrophe in the pattern, you must specify it as two single apostrophes, which will be counted as a single character in the pattern. As examples, EDIT=('C)ITT.TT'), EDIT=('I / T'), and EDIT=('S'II.T') are all valid.

The maximum number of digits (I's and T's) you specify in the pattern must not exceed 15. The maximum length of the pattern must not exceed 22 characters.

The implied length of the edited output field is the same as the length of the pattern. The LENGTH parameter can be used to change the implied length of the edited output field.

To illustrate conceptually how DFSORT produces the edited output from the numeric value, consider the following example:

```
OUTFIL OUTREC=(1,5,ZD,EDIT=(**I/ITTTCR))

with ZD values of C'01230'(+1230)
and C'0004J' (-41)
```

The digits of C'01230' are mapped to the pattern, resulting in C'**0/1230CR'. Since the value is positive, the characters (C'CR') to the right of the last digit are replaced with blanks, resulting in C'**0/1230 '. All digits before the first non-zero digit (1 in this case) are replaced with blanks, resulting in C'** /1230 '. Finally, all characters before the first digit in the pattern are floated to the right, next to the first non-zero digit, resulting in C' **1230 '.

The digits of C'0004J' are mapped to the pattern, resulting in C'**0/0041CR'. Since the value is negative, the characters (C'CR') to the right of the last digit are kept. All digits before the first T digit are replaced with blanks, resulting in C'** / 041CR'. Finally, all characters before the first digit in the pattern are floated to the right, next to the first non-zero digit, resulting in C' **041CR'.

In general terms, the steps DFSORT takes conceptually to produce the edited output from the numeric value are as follows:

- Map the digits of the numeric value to the pattern, padding on the left with zeros, if necessary.
- If the value is positive, replace the leading and trailing signs (if any) with the characters for positive values specified by the SIGNS parameter and replace any characters between the last digit and the trailing sign (if any) with blanks. Otherwise, replace the leading and trailing signs (if any) with the characters for negative values specified by the SIGNS parameter and keep any characters between the last digit and the trailing sign (if any).
- Replace all digits before the first non-zero (I) or significant digit (T) with blanks.
- Float all characters (if any) before the first digit in the pattern to the right, next to the first non-zero (I) or significant digit (T).

The following additional rules apply to edit patterns:

- An insignificant digit (I) after a significant digit (T) is treated as a significant digit.

- If SIGNS is specified, an S in the first or last character of the pattern is treated as a sign; an S anywhere else in the pattern is treated as the letter S. If SIGNS is not specified, an S anywhere in the pattern is treated as the letter S.
- If the pattern contains fewer digits than the value, the leftmost digits of the value will be lost, intentionally or unintentionally. As an example, if you specify 5,5,ZD,EDIT=(IIT) for a value of C'12345', the result will be C'345'. As another example, if you specify 1,6,ZD,EDIT=($IIT.T) for a value of C'100345', the result will be C' $34.5'.

EDxy
specifies an edit pattern for presenting numeric data. EDxy is a special variation of EDIT that allows other characters to be substituted for I and T in the pattern. For example, if you use EDAB instead of EDIT, you must use A in the pattern instead of I and use B instead of T to represent digits. x and y must not be the same character. If SIGNS is specified, x and y must not be S. If SIGNz is specified, x and y must not be the same character as z. You can select x and y from: A-Z, #, $, @, and 0-9.

SIGNS
specifies the sign values to be used when editing numeric values according to the edit mask (Mn) or pattern (EDIT or EDxy). You can specify any or all of the four sign values. Any value not specified must be represented by a comma. Blank will be used for any sign value you do not specify. As examples, SIGNS=(+,-) specifies + for lp, - for ln, blank for tp, and blank for tn; SIGNS=(,,+,-) specifies blank for lp, blank for ln, + for tp, and - for tn.

lp specifies the value for the leading positive sign. If an S is specified as the first character of the edit mask or pattern and the value is positive, the lp value will be used as the leading sign.

ln specifies the value for the leading negative sign. If an S is specified as the first character of the edit mask or pattern and the value is negative, the ln value will be used as the leading sign.

tp specifies the value for the trailing positive sign. If an S is specified as the last character of the edit mask or pattern and the value is positive, the tp value will be used as the trailing sign.

tn specifies the value for the trailing negative sign. If an S is specified as the last character of the edit mask or pattern and the value is negative, the tn value will be used as the trailing sign.

If you want to use any of the following characters as sign values, you must enclose them in single apostrophes: comma, blank, or unbalanced left or right parentheses. A single apostrophe must be specified as four single apostrophes (that is, two single apostrophes enclosed in single apostrophes).

A semicolon cannot be substituted for a comma as the delimiter between sign characters.

SIGNz
specifies the sign values to be used when editing numeric values according to the edit pattern (EDIT or EDxy). SIGNz is a special variation of SIGNS which allows another character to be substituted for S in the pattern. For example, if you use SIGNX instead of SIGNS, you must use X in the pattern instead of S to identify a sign. If EDIT is specified, z must not be I or T. If EDxy is specified, z must not be the same character as either x or y. You can select z from: A-Z, #, $, @, and 0-9.

LENGTH

specifies the length of the edited output field. If the implied length of the edited output field produced using an edit mask or edit pattern is not suitable for presenting your numeric data, LENGTH can be used to make the edited output field shorter or longer.

n specifies the length of the edited output field. The value for n must be between 1 and 22.

LENGTH does not change the pattern used, only the length of the resulting edited output field. For example, as discussed previously for Mn, if you specify:

```
OUTFIL OUTREC=(5,1,ZD,M4)
```

the pattern will be C'ST' rather than C'ST.TT' because the digit length is 1. Specifying:

```
OUTFIL OUTREC=(5,1,ZD,M4,LENGTH=5)
```

will change the pattern to C' ST', not to C'ST.TT'.

If you specify a value for n that is shorter than the implied length, truncation will occur on the left after editing. For example, if you specify:

```
OUTFIL OUTREC=(1,5,ZD,EDIT=($IIT.TT),LENGTH=5)
```

with a value of C'12345', editing according to the specified $IIT.TT pattern will produce C'$123.45', but the specified length of 5 will truncate this to C'23.45'.

If you specify a value for n that is longer than the implied length, padding on the left with blanks will occur after editing. For example, if you specify:

```
OUTFIL OUTREC=(1,5,ZD,EDIT=($IIT.TT),LENGTH=10)
```

with a value of C'12345', editing according to the specified $IIT.TT pattern will produce C'$123.45', but the specified length of 10 will pad this to C' $123.45'.

Sample Syntax:

```
OUTFIL FNAMES=OUT1,OUTREC=(5:21,8,ZD,M19,25:46,5,ZD,M13)
OUTFIL FILES=1,OUTREC=(5,2,BI,C' * ',18,2,BI,80:X),
   ENDREC=2000,OMIT=(5,2,BI,EQ,18,2,BI)
OUTFIL FILES=(2,3),
  OUTREC=(11:35,6,FS,SIGNS=(,,+,-),LENGTH=10,
          31:8,4,PD,EDIT=(**II,IIT.TTXS),SIGNS=(,,+,-))
```

p,m,f,to

specifies that a converted numeric input field is to appear in the reformatted OUTFIL output record. You can convert BI, FI, PD, PD0, ZD or CSF/FS fields to BI, FI, PD, ZD or CSF/FS fields.

p See p under p,m,f,edit.

m See m under p,m,f,edit.

f See f under p,m,f,edit.

to

```
►►──┬─fo──────────────┬──┬──────────────────────────────────────►◄
    └─TO=──┬─fo─────┬─┘  └─,LENGTH=──┬─n────┬─┘
           └─(fo)───┘                └─(n)──┘
```

specifies how the numeric field is to be converted for output.

fo specifies the format for the output field which can be BI, FI, PD, ZD or CSF/FS. Any one of these output field formats (fo) can be used with any one of the input field formats (f).

If you do not specify the LENGTH parameter, DFSORT will determine the implied length of the output field from the length (m) and format (f) of the input field and the format (fo) of the output field. The implied length of the output field can be represented as follows for each output format, where d is the number of digits needed for the input field as shown in Table 30 on page 201, and the result is rounded down to the nearest integer:

Table 32. To Output Field Lengths

Output Format	Output Length	Example Input (f,m)	Example Output Length
BI	4	FS,9	4
FI	4	ZD,7	4
PD	d/2 + 1	BI,4	6
ZD	d	PD,4	7
CSF or FS	d + 1	FI,3	9

For ZD output, F is used as the positive sign and D is used as the negative sign.

For PD output, C is used as the positive sign and D is used as the negative sign.

For CSF/FS output, blank is used as the positive sign, - is used as the negative sign and leading zeros are suppressed.

For ZD, PD and CSF/FS output, the maximum output value is 999999999999999 (15 digits) and the minimum output value is -999999999999999 (15 digits) which correspond to the maximum and minimum input values, respectively.

For BI output:

- An input value greater than 4294967295 (X’FFFFFFFF’) produces an output value of 4294967295 (X’FFFFFFFF’)
- An input value less than zero produces an absolute output value. For example, an input value of P’-5000’ produces a BI output value of 5000 (X’1388’).

For FI output, an input value greater than 2147483647 (X’7FFFFFFF’) produces an output value of 2147483647 (X’7FFFFFFF’), and an input value less than -2147483648 (X’80000000’) produces an output value of -2147483648 (X’80000000’).

fo, TO=fo and TO=(fo) are interchangeable except that:

- fo must be specified before the LENGTH parameter whereas TO can be specified before or after the LENGTH parameter.
- TO=fo or TO=(fo) should be used after a symbol rather than fo to prevent the misinterpretation of fo as f. See the discussion of OUTFIL OUTREC in Chapter 7 for details.

LENGTH

specifies the length of the converted output field. If the implied length of the output field is not suitable, LENGTH can be used to make the output field shorter or longer.

n specifies the length of the converted output field. The value for n must be between 1 and 22.

If you specify a value for n that is shorter than the implied length, truncation on the left will occur after conversion. For example, if you specify:

```
OUTFIL OUTREC=(1,8,ZD,TO=PD,LENGTH=3)
```

with values of ZL8'-12345678' (X'F1F2F3F4F5F6F7D8') and ZL8'58' (X'F0F0F0F0F0F0F5F8'), conversion with the implied length (5) will produce PL5'-12345678' (X'012345678D') and PL5'58' (X'0000000058C'). The specified length of 3 will then result in truncation to PL3'-45678' (X'45678D') and PL3'58' (X'00058C').

If you specify a value for n that is longer than the implied length, padding on the left will occur after conversion using:

- Blanks for CSF/FS output values
- Character zeros for ZD output values
- Binary zeros for PD and BI output values
- Binary zeros for positive FI output values
- Binary ones for negative FI output values

For example, if you specify:

```
OUTFIL OUTREC=(1,4,ZD,TO=FI,LENGTH=6)
```

with values of ZL4'-1234' (X'F1F2F3D4') and ZL4'58' (X'F0F0F5F8'), conversion with the implied length (4) will produce FL4'-1234' (X'FFFFFB2E') and FL4'58' (X'000004D2'). The specified length of 6 will then result in padding to FL6'-1234' (X'FFFFFFFFFB2E') and FL6'58' (X'00000000003A').

Sample Syntax:

```
OUTFIL OUTREC=(21,5,ZD,TO=PD,X,8,4,ZD,FI,LENGTH=2)
```

p,m,Y2x,edit

specifies that an edited four-digit year CH date representation of a two-digit year input date field is to appear in the reformatted OUTFIL output record. The two-digit year date field you specify is transformed to a four-digit year ZD date field as shown in Table 24 on page 196, and then edited according to the edit parameters you specify.

For example, if you specify:

```
OUTFIL OUTREC=(28,3,Y2V,EDIT=(TTTT-T-T))
```

the PL3'yyxx' (X'0yyxxC') date value will be transformed to a Z'yyyyxx' date value and then edited to C'yyyy-x-x'.

See p,m,Y2x and p,m,f,edit for related details.

p,m,Y2x,to
specifies that a converted four-digit year date representation of a two-digit year input date field is to appear in the reformatted OUTFIL output record. The two-digit year date field you specify is transformed to a four-digit year ZD date field as shown in Table 24 on page 196, and then converted according to the to parameters you specify.

For example, if you specify:

```
OUTFIL OUTREC=(5,4,Y2W,TO=PD,LENGTH=6)
```

the C'xxyy' date value will be transformed to a Z'xxyyyy' date value and then converted to PL6'xxyyyy' (X'00000xxyyyyC').

See p,m,Y2x and p,m,f,to for related details.

p,m,lookup
specifies that a character or hexadecimal string from a lookup table is to appear in the reformatted OUTFIL output record. You can use p,m,lookup to select a specified character or hexadecimal string based on matching an input value against character, hexadecimal, or bit constants.

p See p under p,m,a.

m specifies the length in bytes of the input field to be compared to the find-constants. The value for m must be 1 to 64 if character or hexadecimal find-constants are used, or 1 if bit find-constants are used.

lookup
Specifies how the input field is to be changed to the output field, using a lookup table.

CHANGE

```
                  .
                  ┌──────────┐
                  ▼          │
►►─CHANGE=(-v──────,find,set──┴──)──┬──────────────────────────────────────────────────►◄
                                    └─,NOMATCH=(──┬─set──┬──)──┘
                                                  └─q,n─┘
```

specifies a list of change pairs, each consisting of a find-constant to be compared to the input field value and a set-constant to use as the output field when a match occurs.

v specifies the length in bytes of the output field to be inserted in the reformatted OUTFIL output record. The value for v must be between 1 and 64.

find
specifies a find-constant to be compared to the input field value. If the input field value matches the find-constant, the corresponding set-constant is used for the output field. The find-constants can be either character and hexadecimal string constants or bit constants:

- Character string constants (C'xx...x') and hexadecimal string constants (X'yy...yy') can be 1 to m bytes and can be intermixed with each other,

but not with bit constants. See “INCLUDE Control Statement” on page 93 for details of coding character and hexadecimal string constants.

If the string is less than m bytes, it will be padded on the right to a length of m bytes, using blanks (X’40’) for a character string constant or zeros (X’00’) for a hexadecimal string constant.

- Bit constants (B’bbbbbbbb’) must be 1 byte and cannot be intermixed with character or string constants. See “INCLUDE Control Statement” on page 93 for details of coding bit constants.

set
specifies a set-constant to be used as the output field if the corresponding find-constant matches the input field value. The set-constants can be character string constants (C'xx...x') or hexadecimal string constants (X'yy...yy') of 1 to v bytes and can be intermixed. See “INCLUDE Control Statement” on page 93 for details of coding character and hexadecimal string constants.

If the string is less than v bytes, it will be padded on the right to a length of v bytes, using blanks (X'40') for a character string constant or zeros (X'00') for a hexadecimal string constant.

For bit constants, because of the specification of bits to be ignored, more than one find-constant can match an input field value; the set-constant for the first match found will be used as the output field. For example, if you specify:

```
OUTFIL OUTREC=(5,1,
    CHANGE=(2,B'11......',C'A',B'1.......',C'B'))
```

input field value X'C0' (B'11000000') matches both bit constants, but C'A' will be used for the set-constant because its find-constant is the first match.

NOMATCH
specifies the action to be taken if an input field value does not match any of the find-constants. If you do not specify NOMATCH, and no match is found for any input value, DFSORT will terminate processing.

If you specify NOMATCH, it must follow CHANGE.

set
specifies a set-constant to be used as the output field if no match is found. See set under CHANGE for details.

q specifies the position of an input field to be used as the output field if no match is found. See p under p,m,a for details.

n specifies the length of an input field to be used as the output field if no match is found. The value for n must be 1 to v. If n is less than v, the input field will be padded on the right to a length of v bytes, using blanks (X'40').

Sample Syntax:

```
OUTFIL FILES=1,
 OUTREC=(11,1,
         CHANGE=(6,
           C'R',C'READ',
           C'U',C'UPDATE',
           X'FF',C'EMPTY',
           C'A',C'ALTER'),
         NOMATCH=(11,6),
         4X,
```

```
21,1,
CHANGE=(10,
  B'.1......',C'VSAM',
  B'.0......',C'NON-VSAM'))
```

seqnum

```
►►──SEQNUM,n,fs──┬──────────────────────┬──────────────────────────────────►◄
                 └─,START=──┬─j─────┬──┘
                            └─(j)───┘

►►──┬──────────────────────┬──────────────────────────────────────────────►◄
    └─,INCR=──┬─i─────┬──┘
              └─(i)───┘
```

specifies that a sequence number is to appear in the reformatted OUTFIL output record. The sequence numbers are assigned in the order in which the records are received for OUTFIL OUTREC processing.You can create BI, PD, ZD or CSF/FS sequence numbers and control their lengths, starting values and increment values.

n specifies the length of the sequence number. The value for n must be between 1 and 16.

fs specifies the format for the sequence number which can be BI, PD, ZD or CSF/FS.

For a ZD format sequence number, F is used as the sign.

For a PD format sequence number, C is used as the sign.

For a CSF/FS format sequence number, blank is used as the sign and leading zeros are suppressed.

For a PD, ZD or CSF/FS format sequence number, the maximum value DFSORT can create is limited to the lesser of 15 decimal digits or the output field length (n). If a sequence number overflows this limit, it will be truncated to the lesser of 15 decimal digits or the output field length, and then subsequently incremented as usual.

For a BI format sequence number, the maximum value DFSORT can create is limited to the lesser of 8 bytes of ones (X'FFFFFFFFFFFFFFFF') or the number of ones that will fit in the specified output field length (n). If a sequence number overflows this limit, it will be truncated to the lesser of 8 bytes or the output field length, and then subsequently incremented as usual.

START
specifies the starting value for the sequence number.

j specifies the starting value. The value for j must be between 0 and 100000000000. The default for j is 1.

INCR
specifies the increment value for the sequence number.

i specifies the increment value. The value for i must be between 1 and 10000000. The default for i is 1.

Sample Syntax:

```
OUTFIL FNAMES=O1,OUTREC=(SEQNUM,6,ZD,START=1000,INCR=50,
         X,22,8,X,13,5)
OUTFIL FNAMES=O2,OUTREC=(1,12,SEQNUM,4,BI)
```

Default for OUTREC: None; must be specified.

CONVERT

```
►►──CONVERT──────────────────────────────────────────────────────────►◄
```

Specifies that variable-length OUTFIL input records are to be converted to fixed-length OUTFIL output records for this OUTFIL group. You must use the OUTREC parameter to define the reformatted records; all OUTREC parameters are allowed except p and p,HEX. Remember that the data for the variable-length input records starts at position 5 (after the RDW), while the data for the fixed-length output records starts at position 1 (no RDW).

By default, CONVERT automatically uses VLFILL=X’40’ (blank fill byte) to allow processing of variable-length input records which are too short to contain all specified OUTREC fields. You can specify VLFILL=byte to change the fill byte.

All OUTFIL data sets for which CONVERT is used must have or will be given fixed-length record formats.

If CONVERT is specified for fixed-length input records, it will not be used.

Sample Syntax:

```
OUTFIL FNAMES=VTOF,CONVERT,OUTREC=(1:5,14,35:32,8,50:22,6)
```

Default for CONVERT: None; must be specified.

VLFILL

```
►►──VLFILL=byte──────────────────────────────────────────────────────►◄
```

Allows DFSORT to continue processing if a variable-length OUTFIL input record is found to be too short to contain all specified OUTFIL OUTREC fields for this OUTFIL group. Without VLFILL=byte, a short record causes DFSORT to issue message ICE218A and terminate. With VLFILL=byte, missing bytes in OUTFIL OUTREC fields are replaced with fill bytes so the filled fields can be processed.

If VLFILL=byte is specified for fixed-length input records, it will not be used.

byte
specifies the fill byte. Permissible values are C'x' and X'yy'.

C'x'
Character byte: The value x must be one EBCDIC character. If you want to use an apostrophe as the fill byte, you must specify it as C''''.

X'yy'
Hexadecimal byte: The value yy must be one pair of hexadecimal digits (00-FF).

Sample Syntax:

```
OUTFIL FNAMES=VTOF,CONVERT,OUTREC=(1,20,2X,35,10),VLFILL=C'*'
OUTFIL FNAMES=OUT1,VLFILL=X'FF',OUTREC=(1,4,15,5,52)
```

Default for VLFILL: VLFILL=X'40' (blank fill byte) if CONVERT is specified. Otherwise, none; must be specified.

LINES

```
►►──LINES=n──────────────────────────────────────────────────────────►◄
```

Specifies the number of lines per page to be used for the reports produced for this OUTFIL group. DFSORT uses ASA carriage control characters to control page ejects and the placement of the lines in your report, according to your specifications.

n specifies the number of lines per page. The value for n must be between 1 and 255. However, n—or the default for n if LINES is not specified—must be greater than or equal to the number of lines needed for each of the following:

- The HEADER1 lines
- The TRAILER1 lines
- The sum of all lines for HEADER2, TRAILER2, HEADER3s, TRAILER3s, and the data lines and blank lines produced from an input record.

Sample Syntax:

```
OUTFIL FNAMES=RPT1,LINES=50
```

Default for LINES: None; must be specified, unless HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, or NODETAIL is specified, in which case the default for LINES is 60.

HEADER1

```
                ┌─,──────┐
►►──HEADER1=(──┴──┬────┬──┴──┬─r──────────────────────────┬──)──────────────►◄
                  └─c:─┘     ├─p,m────────────────────────┤
                             ├─┬─DATE──────────────────┬─┤
                             │ ├─&DATE─────────────────┤ │
                             │ └─┬─DATE=──┬──(abcd)──┘ │
                             │   └─&DATE=─┘             │
                             ├─┬─TIME──────────────────┬─┤
                             │ ├─&TIME─────────────────┤ │
                             │ └─┬─TIME=──┬──(abc)───┘ │
                             │   └─&TIME=─┘             │
                             └─┬─PAGE──────────────────┬─┘
                               └─&PAGE─────────────────┘
```

Specifies the report header to be used for the reports produced for this OUTFIL group.The report header appears by itself as the first page of the report. DFSORT uses ASA carriage control characters to control page ejects and the placement of the lines in your report, according to your specifications.

You can choose to include any or all of the following report elements in your report header:

- Blanks and character strings
- Unedited input fields from the first OUTFIL input record
- Current date
- Current time
- Page number.

The report header consists of the elements you select, in the order in which you specify them, and in the columns or lines you specify.

c: specifies the column in which the first position of the associated report element is to appear, relative to the start of the data in the report record. Ignore the RDW (variable-length report records only) and carriage control character when specifying c:. That is, 1: indicates the first byte of the data in the report record for both fixed-length and variable-length report records.

Unused space preceding the specified column is padded with EBCDIC blanks. The following rules apply:

- c must be a number between 1 and 32752.
- c: must be followed by a report element, but must not precede / or n/.
- c must not overlap the previous report element in the report record.
- The colon (:) is treated like the comma (,) or semicolon (;) for continuation to another line.

r specifies that blanks or a character string are to appear in the report record, or that a new report record is to be started in the header, with or without intervening blank lines. These report elements can be specified before or after any other report elements. Consecutive character strings or blank lines can be specified. Permissible values are nX, n'xx...x', nC'xx...x', /.../ and n/.

nX Blanks. n bytes of EBCDIC blanks (X'40') are to appear in the report record. n can range from 1 to 4095. If n is omitted, 1 is used.

n'xx...x'
Character string. n repetitions of the character string constant ('xx...x') are to appear in the report record. n can range from 1 to 4095. If n is omitted, 1 is used. x can be any EBCDIC character. You can specify 1 to 256 characters.

nC'xx...x' can be used instead of n'xx...x'.

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

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

/.../ or n/
Blank lines or a new line. A new report record is to be started in the header with or without intervening blank lines. If /.../ or n/ is specified at the beginning or end of the header, n blank lines are to appear in the header. If /.../ or n/ is specified in the middle of the header, n-1 blank lines are to appear in the header (thus, / or 1/ indicates a new line with no intervening blank lines).

Either n/ (for example, 5/) or multiple /'s (for example, /////) can be used. n can range from 1 to 255. If n is omitted, 1 is used.

As an example, if you specify:

```
OUTFIL HEADER1=(2/,'First line of text',/,
                   'Second line of text',2/,
                   'Third line of text',2/)
```

the report header appears as follows when printed:

```
blank line
blank line
First line of text
Second line of text
blank line
Third line of text
blank line
blank line
```

p,m
specifies that an unedited input field, from the first OUTFIL input record for which a data record appears in the report, is to appear in the report record.

p specifies the first byte of the input field relative to the beginning of the OUTFIL input record. 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 can extend beyond byte 32752. See “OUTFIL Statements Notes” on page 236 for special rules concerning variable-length records.

m specifies the length in bytes of the input field. The value for m must be between 1 and 256.

DATE
specifies that the current date is to appear in the report record in the form ’mm/dd/yy’, where mm represents the month (01-12), dd represents the day (01-31), and yy represents the last two digits of the year (for example, 95).

&DATE
&DATE can be used instead of DATE.

DATE=(abcd)
specifies that the current date is to appear in the report record in the form ’adbdc’, where a, b, and c indicate the order in which the month, day, and year are to appear and whether the year is to appear as two or four digits, and d is the character to be used to separate the month, day and year.

For a, b, and c, use M to represent the month (01-12), D to represent the day (01-31), Y to represent the last two digits of the year (for example, 95), or 4 to represent the four digits of the year (for example, 1995). M, D, and Y or 4 can each be specified only once. Examples: DATE=(DMY.) would produce a date of the form ’dd.mm.yy’ which on March 29, 1995, would appear as ’29.03.95’. DATE=(4MD-) would produce a date of the form ’yyyy-mm-dd’ which on March 29, 1995, would appear as ’1995-03-29’.

a, b, c, and d must be specified.

&DATE=(abcd)
&DATE=(abcd) can be used instead of DATE=(abcd).

TIME
specifies that the current time is to appear in the report record in the form ’hh:mm:ss’, where hh represents the hour (00-23), mm represents the minutes (00-59), and ss represents the seconds (00-59).

&TIME
&TIME can be used instead of TIME.

TIME=(abc)
specifies that the current time is to appear in the report record in the form 'hhcmmcss' (24-hour time) or 'hhcmmcss xx' (12-hour time).

If ab is 24, the time is to appear in the form 'hhcmmcss' (24-hour time) where hh represents the hour (00-23), mm represents the minutes (00-59), ss represents the seconds (00-59), and c is the character used to separate the hours, minutes, and seconds. Example: TIME=(24.) would produce a time of the form 'hh.mm.ss' which at 08:25:13 pm would appear as '20.25.13'.

If ab is 12, the time is to appear in the form 'hhcmmcss xx' (12-hour time) where hh represents the hour (01-12), mm represents the minutes (00-59), ss represents the seconds (00-59), xx is am or pm, and c is the character used to separate the hours, minutes, and seconds. Example: TIME=(12.) would produce a time of the form 'hh.mm.ss xx' which at 08:25:13 pm would appear as '08.25.13 pm'.

ab and c must be specified.

&TIME=(abc)
&TIME=(abc) can be used instead of TIME=(abc).

PAGE
specifies that the page number is to appear in the report record. The page number for the report header appears as ' 1'.

If HEADER1 is specified with PAGE, PAGE for the report header (first page) will be ' 1' and PAGE for the next page (second page) will be ' 2'. If HEADER1 is specified without PAGE, PAGE for the page after the report header (second page) will be ' 1' (typical of a report with a cover sheet).

&PAGE
&PAGE can be used instead of PAGE.

Sample Syntax:

```
OUTFIL FNAMES=(RPT1,RPT2),
   HEADER1=(30:'January Report',4/,
            28:'Prepared on ',DATE,//,
            32:'at ',TIME,//,
            28:'using DFSORT'S OUTFIL',5/,
            10:'Department: ',12,8,50:'Page:',PAGE)
```

Default for HEADER1: None; must be specified.

TRAILER1

```
                  ┌──,──────────────────────────────────────────────────────┐
                  ▼                                                         │
►►──TRAILER1=─(──┬──────┬──┬─r──────────────────────────────────────────────┴──)──────►◄
                 └─c:───┘  ├─p,m──────────────────────────────────────────────┤
                           ├─┬─DATE──────────────────────────────────────────┤
                           │ ├─&DATE─────────────────────────────────────────┤
                           │ └─┬─────────┬─(abcd)─────────────────────────────┤
                           │   ├─DATE=───┤                                   │
                           │   └─&DATE=──┘                                   │
                           ├─┬─TIME──────────────────────────────────────────┤
                           │ ├─&TIME─────────────────────────────────────────┤
                           │ └─┬─────────┬─(abc)───────────────────────────┤
                           │   ├─TIME=───┤                                   │
                           │   └─&TIME=──┘                                   │
                           ├─┬─PAGE──┬──────────────────────────────────────┤
                           │ └─&PAGE─┘                                       │
                           ├───┬─COUNT─────────────────────────────────────┤
                           │   ├─COUNT15───────────────────────────────────┤
                           │   └─COUNT=(edit)──────────────────────────────┤
                           ├───┬─SUBCOUNT──────────────────────────────────┤
                           │   ├─SUBCOUNT15────────────────────────────────┤
                           │   └─SUBCOUNT=(edit)───────────────────────────┤
                           ├───┬─TOTAL=──┬──(p,m,f──┬──────────┬──)────────┤
                           │   └─TOT=────┘           └─,edit────┘           │
                           ├─MIN=──(p,m,f──┬──────────┬──)──────────────────┤
                           │               └─,edit────┘                     │
                           ├─MAX=──(p,m,f──┬──────────┬──)──────────────────┤
                           │               └─,edit────┘                     │
                           ├─AVG=──(p,m,───┬──────────┬──)──────────────────┤
                           │               └─,edit────┘                     │
                           ├───┬─SUBTOTAL=───┬──(p,m,f──┬──────────┬──)────┤
                           │   ├─SUBTOT=─────┤          └─,edit────┘        │
                           │   └─SUB=────────┘                              │
                           ├─SUBMIN=──(p,m,f──┬──────────┬──)──────────────┤
                           │                  └─,edit────┘                  │
                           ├─SUBMAX=──(p,m,f──┬──────────┬──)──────────────┤
                           │                  └─,edit────┘                  │
                           └─SUBAVG=──(p,m,f──┬──────────┬──)──────────────┘
                                              └─,edit────┘
```

Specifies the report trailer to be used for the reports produced for this OUTFIL group. The report trailer appears by itself as the last page of the report. DFSORT uses ASA carriage control characters to control page ejects and the placement of the lines in your report, according to your specifications.

You can choose to include any or all of the following report elements in your report trailer:

- Blanks and character strings
- Unedited input fields from the last OUTFIL input record
- Current date
- Current time
- Page number
- Any or all of the following statistics:
  - Count of data records in the report
  - Total, minimum, maximum, or average for each specified ZD, PD, BI, FI, or CSF/FS numeric input field in the data records of the report, edited to contain signs, decimal points, leading zeros or no leading zeros, and so on.

The report trailer consists of the elements you select, in the order in which you specify them, and in the columns or lines you specify.

c: See c: under HEADER1.

r specifies that blanks or a character string are to appear in the report record, or that a new report record is to be started in the trailer, with or without intervening blank lines. These report elements can be specified before or after any other report elements. Consecutive character strings or blank lines can be specified. Permissible values are nX, n'xx...x', nC'xx...x', /.../, and n/.

nX Blanks. See nX under r for HEADER1.

n'xx...x'
Character string. See n'xx...x' under r for HEADER1. nC'xx...x' can be used instead of n'xx...x'

/.../ or n/
Blank lines or a new line. A new report record is to be started in the trailer, with or without intervening blank lines. If /.../ or n/ is specified at the beginning or end of the header, n blank lines are to appear in the trailer. If /.../ or n/ is specified in the middle of the trailer, n-1 blank lines are to appear in the trailer (thus, / or 1/ indicates a new line with no intervening blank lines).

Either n/ (for example, 5/) or multiple /'s (for example, /////) can be used. n can range from 1 to 255. If n is omitted, 1 is used.

p,m
specifies that an unedited input field, from the last OUTFIL input record for which a data record appears in the report, is to appear in the report record.

p See p under HEADER1.

m See m under HEADER1.

DATE
See DATE under HEADER1.

&DATE
&DATE can be used instead of DATE. See &DATE under HEADER1.

DATE=(abcd)
See DATE=(abcd) under HEADER1.

&DATE=(abcd)
&DATE=(abcd) can be used instead of DATE=(abcd). See &DATE=(abcd) under HEADER1.

TIME
See TIME under HEADER1.

&TIME
&TIME can be used instead of TIME. See &TIME under HEADER1.

TIME=(abc)
See TIME=(abc) under HEADER1.

&TIME=(abc)
&TIME=(abc) can be used instead of TIME=(abc). See &TIME=(abc) under HEADER1.

PAGE
specifies that the current page number is to appear in the report record.

The page number for the trailer appears as 6 digits, right-justified, with leading zeros suppressed. For example, if the page is numbered 12, it appears as ' 12'.

&PAGE
&PAGE can be used instead of PAGE.

COUNT
specifies that the count of data records in the report is to appear in the report record as 8 digits, right-justified, with leading zeros suppressed. For example, if there are 6810 input records in the report, the count appears as ' 6810'.

COUNT counts input records, not data records. However, unless slash (/) is used in OUTREC to produce multiple records, the count will also represent the number of data records.

COUNT15
same as COUNT except that the count appears as 15 digits.

COUNT=(edit)
same as COUNT except that the 15–digit count appears edited as specified. See p,m,f,edit under OUTREC for further details on the edit fields you can use.

SUBCOUNT
specifies that the running count of input records in the report is to appear in the report record as 8 digits, right-justified, with leading zeros suppressed.

For TRAILER1, the running count is the same as the count, so SUBCOUNT produces the same value as COUNT.

SUBCOUNT counts input records, not data records. However, unless slash (/) is used in OUTREC to produce multiple records, the running count will also represent the number of data records.

SUBCOUNT15
same as SUBCOUNT except that the running count appears as 15 digits.

SUBCOUNT=(edit)
same as SUBCOUNT except that the 15–digit running count appears edited as specified. See p,m,f,edit under OUTREC for further details on the edit fields you can use.

TOTAL
specifies that an edited total, for the values of a numeric input field in all data records of the report, is to appear in the report record.

TOT can be used instead of TOTAL.

p,m,f,edit
specifies the numeric input field for which the total is to be produced and how the output field (that is, the total) is to be edited.

See p,m,f,edit under OUTREC for further details. However, note PD0 is not allowed for TOTAL and that for TOTAL, the number of digits needed with Mn edit masks is the maximum for that format type rather than the actual length of the field, as follows:

Table 33. Digits Needed for TOTAL Fields

Format	Digits Needed
ZD	15
PD	15
BI	10
FI	10
CSF or FS	15

MIN
specifies that an edited minimum, for the values of a numeric input field in all data records of the report, is to appear in the report record.

p,m,f,edit
specifies the numeric input field for which the minimum is to be produced and how the output field (that is, the minimum) is to be edited.

See p,m,f,edit under OUTREC for further details. However, note that PD0 is not allowed for MIN.

MAX
specifies that an edited maximum, for the values of a numeric input field in all data records of the report, is to appear in the report record.

p,m,f,edit
specifies the numeric input field for which the maximum is to be produced and how the output field (that is, the maximum) is to be edited.

See p,m,f,edit under OUTREC for further details. However, note that PD0 is not allowed for MAX.

AVG
specifies that an edited average, for the values of a numeric input field in all data records of the report, is to appear in the report record. The average (or mean) is calculated by dividing the total by the count and rounding down to the nearest integer. For example:

```
+2305 / 152 = +15
-2305 / 152 = -15
```

p,m,f,edit
specifies the numeric input field for which the average is to be produced and how the output field (that is, the average) is to be edited.

See p,m,f,edit under OUTREC for further details. However, note that PD0 is not allowed for AVG.

SUBTOTAL
specifies that an edited running total, for the values of a numeric input field in all data records of the report, is to appear in the report record.

SUBTOT or SUB can be used instead of SUBTOTAL.

For TRAILER1, the running total is the same as the total, so SUBTOTAL produces the same value as TOTAL.

p,m,f,edit
specifies the numeric input field for which the running total is to be produced and how the output field (that is, the running total) is to be edited.

See p,m,f,edit under TOTAL for further details.

SUBMIN
specifies that an edited running minimum, for the values of a numeric input field in all data records of the report, is to appear in the report record.

For TRAILER1, the running minimum is the same as the minimum, so SUBMIN produces the same value as MIN.

p,m,f,edit
specifies the numeric input field for which the running minimum is to be produced and how the output field (that is, the running minimum) is to be edited.

See p,m,f,edit under OUTREC for further details. However, note that PD0 is not allowed for SUBMIN.

SUBMAX
specifies that an edited running maximum, for the values of a numeric input field in all data records of the report, is to appear in the report record.

For TRAILER1, the running maximum is the same as the maximum, so SUBMAX produces the same value as MAX.

p,m,f,edit
specifies the numeric input field for which the running maximum is to be produced and how the output field (that is, the running maximum) is to be edited.

See p,m,f,edit under OUTREC for further details. However, note that PD0 is not allowed for SUBMAX.

SUBAVG
specifies that an edited running average, for the values of a numeric input field in all data records of the report, is to appear in the report record.

For TRAILER1, the running average is the same as the average, so SUBAVG produces the same value as AVG.

p,m,f,edit
specifies the numeric input field for which the running average is to be produced and how the output field (that is, the running average) is to be edited.

See p,m,f,edit under OUTREC for further details. However, note that PD0 is not allowed for SUBAVG.

Sample Syntax:

```
OUTFIL FNAMES=RPT,
  TRAILER1=(5/,
   10:'Summary of Report for Division Revenues',3/,
   10:'Number of divisions reporting: ',COUNT,2/,
   10:'Total revenue: ',TOTAL=(25,5,PD,M5),2/,
```

```
10:'Lowest revenue: ',MIN=(25,5,PD,M5),2/,
10:'Highest revenue: ',MAX=(25,5,PD,M5),2/,
10:'Average revenue: ',AVG=(25,5,PD,M5))
```

Default for TRAILER1: None; must be specified.

HEADER2

```
                  ┌─,──────────────────────────────────────┐
                  ▼                                        │
►►─HEADER2=─(──────┬──────┬──────────┬─r──────────────────┬──────────────────)──────────────►◄
                   └─c:───┘          ├─p,m────────────────┤
                                     ├─┬─DATE─────────────┬┤
                                     │ ├─&DATE────────────┤│
                                     │ └─┬─DATE=──┬─(abcd)─┘│
                                     │   └─&DATE=─┘         │
                                     ├─┬─TIME─────────────┬┤
                                     │ ├─&TIME────────────┤│
                                     │ └─┬─TIME=──┬─(abc)──┘│
                                     │   └─&TIME=─┘         │
                                     └─┬─PAGE─────────────┬┘
                                       └─&PAGE────────────┘
```

Specifies the page header to be used for the reports produced for this OUTFIL group. The page header appears at the top of each page of the report, except for the report header page (if any) and report trailer page (if any). DFSORT uses ASA carriage control characters to control pageejects and the placement of the lines in your report, according to your specifications.

You can choose to include any or all of the following report elements in your page header:

- Blanks and character strings
- Unedited input fields from the first OUTFIL input record for which a data record appears on the page
- Current date
- Current time
- Page number.

The page header consists of the elements you select, in the order in which you specify them, and in the columns or lines you specify.

c: See c: under HEADER1.

r See r under HEADER1.

p,m
specifies that an unedited input field, from the first OUTFIL input record for which a data record appears on the page, is to appear in the report record. See p,m under HEADER1 for further details.

DATE
See DATE under HEADER1.

&DATE
&DATE can be used instead of DATE. See &DATE under HEADER1.

DATE=(abcd)
See DATE=(abcd) under HEADER1.

&DATE=(abcd)
&DATE=(abcd) can be used instead of DATE=(abcd). See &DATE=(abcd) under HEADER1.

TIME
See TIME under HEADER1.

&TIME
&TIME can be used instead of TIME. See &TIME under HEADER1.

TIME=(abc)
See TIME=(abc) under HEADER1.

&TIME=(abc)
&TIME=(abc) can be used instead of TIME=(abc). See &TIME=(abc) under HEADER1.

PAGE
specifies that the current page number is to appear in the OUTFIL report record. The page number for the header appears as 6 digits, right-justified, with leading zeros suppressed. For example, if the page is numbered 3, it appears as ' 3'.

&PAGE
&PAGE can be used instead of PAGE.

If HEADER1 is specified with PAGE and HEADER2 is specified with PAGE, the page number for the first page header will be ' 2'. If HEADER1 is not specified or is specified without PAGE and HEADER2 is specified with PAGE, the page number for the first page header will be ' 1'.

Sample Syntax:

```
OUTFIL FNAMES=STATUS,
  HEADER2=(5:'Page ',PAGE,' of Status Report for ',DATE=(MD4/),
           ' at ',TIME=(12:),2/,
           10:'Item ',20:'Status      ',35:'Count',/,
           10:'-----',20:'------------',35:'-----'),
   OUTREC=(10:6,5,
           20:14,1,CHANGE=(12,
                           C'S',C'Ship',
                           C'H',C'Hold',
                           C'T',C'Transfer'),
                   NOMATCH=(C'*Check Code*'),
           36:39,4,ZD,M10,
           132:X)
```

Default for HEADER2: None; must be specified.

TRAILER2

```
                  ┌──,──────────────────────────────────────────────────┐
►►──TRAILER2=─(──┴──┬──────┬──┬─r──────────────────────────────────────┴──)──────►◄
                    └─c:───┘  ├─p,m──────────────────────────────────────┤
                              ├─┬─DATE──────────────────────────────────┤
                              │ ├─&DATE─────────────────────────────────┤
                              │ └─┬─DATE=──┬──(abcd)──┬──────────────────┤
                              │   └─&DATE=─┘           │                  │
                              ├─┬─TIME──────────────────────────────────┤
                              │ ├─&TIME─────────────────────────────────┤
                              │ └─┬─TIME=──┬──(abc)───┬──────────────────┤
                              │   └─&TIME=─┘           │                  │
                              ├─┬─PAGE──┬───────────────────────────────┤
                              │ └─&PAGE─┘                                │
                              ├───┬─COUNT──────────────────────────────┤
                              │   ├─COUNT15────────────────────────────┤
                              │   └─COUNT=(edit)───────────────────────┤
                              ├───┬─SUBCOUNT───────────────────────────┤
                              │   ├─SUBCOUNT15─────────────────────────┤
                              │   └─SUBCOUNT=(edit)────────────────────┤
                              ├─┬─TOTAL=─┬──(p,m,f─┬──────────┬──)─────┤
                              │ └─TOT=───┘          └─,edit────┘        │
                              ├─MIN=──(p,m,f─┬──────────┬──)────────────┤
                              │              └─,edit────┘               │
                              ├─MAX=──(p,m,f─┬──────────┬──)────────────┤
                              │              └─,edit────┘               │
                              ├─AVG=──(p,m,f─┬──────────┬──)────────────┤
                              │              └─,edit────┘               │
                              ├───┬─SUBTOTAL=──┬──(p,m,f─┬──────────┬──)─┤
                              │   ├─SUBTOT=────┤         └─,edit────┘    │
                              │   └─SUB=───────┘                         │
                              ├─SUBMIN=──(p,m,f─┬──────────┬──)─────────┤
                              │                 └─,edit────┘            │
                              ├─SUBMAX=──(p,m,f─┬──────────┬──)─────────┤
                              │                 └─,edit────┘            │
                              └─SUBAVG=──(p,m,f─┬──────────┬──)─────────┘
                                                └─,edit────┘
```

Specifies the page trailer to be used for the reports produced for this OUTFIL group. The page trailer appears at the very bottom of each page of the report (as specified or defaulted by the LINES value), except for the report header page (if any) and report trailer page (if any). DFSORT uses ASA carriage control characters to control pageejects and the placement of the lines in your report, according to your specifications.

You can choose to include any or all of the following report elements in your page trailer:

- Blanks and character strings
- Unedited input fields from the last OUTFIL input record for which a data record appears on the page
- Current date
- Current time
- Page number
- Any or all of the following statistics:
  - Count of data records on the page

- Total, minimum, maximum, or average for each specified ZD, PD, BI, FI, or CSF/FS numeric input field in the data records on the page, edited to contain signs, decimal points, leading zeros or no leading zeros, and so on
- Running total, minimum, maximum, or average for each specified ZD, PD, BI, FI, or CSF/FS numeric input field in the data records up to this point, edited to contain signs, decimal points, leading zeros or no leading zeros, and so on.

The page trailer consists of the elements you select, in the order in which you specify them, and in the columns or lines you specify.

c: See c: under HEADER1.

r See r under TRAILER1.

p,m
specifies that an unedited input field, from the last OUTFIL input record for which a data record appears on the page, is to appear in the report record. See p,m under TRAILER1 for further details.

DATE
See DATE under HEADER1.

&DATE
&DATE can be used instead of DATE. See &DATE under HEADER1.

DATE=(abcd)
See DATE=(abcd) under HEADER1.

&DATE=(abcd)
&DATE=(abcd) can be used instead of DATE=(abcd). See &DATE=(abcd) under HEADER1.

TIME
See TIME under HEADER1.

&TIME
&TIME can be used instead of TIME. See &TIME under HEADER1.

TIME=(abc)
See TIME=(abc) under HEADER1.

&TIME=(abc)
&TIME=(abc) can be used instead of TIME=(abc). See &TIME=(abc) under HEADER1.

PAGE
See PAGE under TRAILER1.

&PAGE
&PAGE can be used instead of PAGE. See &PAGE under TRAILER1.

COUNT
specifies that the count of data records on the page is to appear in the report record as 8 digits, right-justified, with leading zeros suppressed. For example, if page 1 has 40 input records, page 2 has 40 input records, and page 3 has 26 input records, COUNT will show ' 40' for page 1, ' 40' for page 2, and ' 26' for page 3.

COUNT counts input records, not data records. However, unless slash (/) is used to produce multiple records, the count will also represent the number of data records.

COUNT15
same as COUNT except that the count appears as 15 digits.

COUNT=(edit)
same as COUNT except that the 15–digit count appears edited as specified. See p,m,f,edit under OUTREC for further details on the edit fields you can use.

SUBCOUNT
specifies that the count of input records up to this point in the report is to appear in the report record as 8 digits, right-justified, with leading zeros suppressed. The running count accumulates the count for all pages up to and including the current page. For example, if page 1 has 40 input records, page 2 has 40 input records, and page 3 has 26 input records, SUBCOUNT will show ' 40' for page 1, ' 80' for page 2, and ' 106' for page 3.

SUBCOUNT counts input records, not data records. However, unless slash (/) is used to produce multiple records, the running count will also represent the number of data records.

SUBCOUNT15
same as SUBCOUNT except that the running count appears as 15 digits.

SUBCOUNT=(edit)
same as SUBCOUNT except that the 15–digit running count appears edited as specified. See p,m,f,edit under OUTREC for further details on the edit fields you can use.

TOTAL
specifies that an edited total, for the values of a numeric input field in the data records on the page, is to appear in the report record.

TOT can be used instead of TOTAL.

p,m,f,edit
See p,m,f,edit under TOTAL for TRAILER1.

MIN
specifies that an edited minimum, for the values of a numeric input field in the data records on the page, is to appear in the report record.

p,m,f,edit
See p,m,f,edit under MIN for TRAILER1.

MAX
specifies that an edited maximum, for the values of a numeric input field in the data records on the page, is to appear in the report record.

p,m,f,edit
See p,m,f,edit under MAX for TRAILER1.

AVG
specifies that an edited average, for the values of a numeric input field in the data records on the page, is to appear in the report record.

p,m,f,edit
See p,m,f,edit under AVG for TRAILER1.

SUBTOTAL
specifies that an edited running total, for the values of a numeric input field in the data records up to this point in the report, is to appear in the report

record. The running total accumulates the total for all pages up to and including the current page. For example, if the total for a selected numeric field is +200 for page 1, -250 for page 2, and +90 for page 3, SUBTOTAL will be +200 for page 1, -50 for page 2, and +40 for page 3.

SUBTOT or SUB can be used instead of SUBTOTAL.

p,m,f,edit
See p,m,f,edit under SUBTOTAL for TRAILER1.

SUBMIN
specifies that an edited running minimum, for the values of a numeric input field in the data records up to this point in the report, is to appear in the report record. The running minimum selects the minimum from all pages up to and including the current page. For example, if the minimum for a selected numeric field is +200 for page 1, -250 for page 2, and +90 for page 3, SUBMIN will be +200 for page 1, -250 for page 2, and -250 for page 3.

p,m,f,edit
See p,m,f,edit under SUBMIN for TRAILER1.

SUBMAX
specifies that an edited running maximum, for the values of a numeric input field in the data records up to this point in the report, is to appear in the report record. The running maximum selects the maximum from all pages up to and including the current page. For example, if the maximum for a selected numeric field is -100 for page 1, +250 for page 2, and +90 for page 3, SUBMAX will be -100 for page 1, +250 for page 2, and +250 for page 3.

p,m,f,edit
See p,m,f,edit under SUBMAX for TRAILER1.

SUBAVG
specifies that an edited running average, for the values of a numeric input field in the data records up to this point in the report, is to appear in the report record. The running average computes the average for all pages up to and including the current page. For example, if the count of data records and total for a selected numeric field are 60 and +2205 for page 1, respectively, 60 and -6252 for page 2, respectively, and 23 and -320 for page 3, respectively, SUBAVG will be +36 for page 1, -33 for page 2, and -30 for page 3.

p,m,f,edit
See p,m,f,edit under SUBAVG for TRAILER1.

Sample Syntax:

```
OUTFIL FNAMES=STATS,
   STARTREC=3,
    OUTREC=(20:23,3,PD,M16,
            30:40,3,PD,M16,
            80:X),
   TRAILER2=(/,2:'Average on page:',
             20:AVG=(23,3,PD,M16),
             30:AVG=(40,3,PD,M16),/,
             2:'Average so far:',
             20:SUBAVG=(23,3,PD,M16),
             30:SUBAVG=(40,3,PD,M16))
```

Default for TRAILER2: None; must be specified.

SECTIONS

```
►►──SECTIONS=──────────────────────────────────────────────────────►

      ┌─,──────────────────────────────────────────────────────┐
      │   ┌─,──────────────────────────────────────────────────┐│
►─(───┴───┴─p,m──┬─SKIP=──┬─P──┬──────────────────────────────┬──┴──)──────────►◄
                 │        ├─L──┤                              │
                 │        └─nL─┘                              │
                 │          ┌─,──────┐                        │
                 ├─HEADER3=─┴─(──┴─field──┴─)──┬──────────────┤
                 │                             └─,PAGEHEAD────┘
                 │          ┌─,──────┐                        │
                 └─TRAILER3=─┴─(──┴─field──┴─)──────────────┘
```

Specifies the section break processing to be used for the reports produced for this OUTFIL group. A section break field divides the report into sets of sequential OUTFIL input records with the same binary value for that field, which result in corresponding sets of data records (that is, sections) in the report. A break is said to occur when the binary value changes. Of course, since a break can occur in any record, the data records of a section can be split across pages in your report.

For each section break field you specify, you can choose to include any or all of the following:

- A page eject between sections.
- One or more blank lines to appear between sections on the same page.
- A section header to appear before the first data record of each section and optionally, at the top of each page. When a page header and section header are both to appear at the top of a page, the section header will follow the page header.
- A section trailer to appear after the last data record of each section. When a page trailer and section trailer are both to appear at the bottom of a page, the page trailer will follow the section trailer.

DFSORT uses ASA carriage control characters to control page ejects and the placement of the lines in your report, according to your specifications.

If multiple section break fields are used, they are processed in first-to-last order, in the same way they would be sorted by these fields. In fact, the input data set is generally sorted by the section break fields, to group the records with the same section break values together for the report. This sorting can be done by the same application that produces the report or by a previous application.

A break in section break field 1 results in a break in section break fields 2 through n. A break in section break 2 results in a break in section break fields 3 through n, and so on. The section headers appear before each section in first-to-last order, whereas the section trailers appear in last-to-first order. For example, if section break fields represented by B1 with header H3A and trailer T3A, B2 with header H3B and trailer T3B, and B3 with header H3C and trailer T3C are specified in order, the following can appear:

```
H3A (header for B1=1 section)
  H3B (header for B2=1 section)
    H3C (header for B3=1 section)
      data records for B1=1, B2=1, B3=1 (new B1, B2, and B3 section)
    T3C (trailer for B3=1 section)
    H3C (header for B3=2 section)
      data records for B1=1, B2=1, B3=2 (new B3 section)
    T3C (trailer for B3=2 section)
  T3B (trailer for B2=1 section)
  H3B (header for B2=2 section)
    H3C (header for B3=1 section)
      data records for B1=1, B2=2, B3=1 (new B2 and B3 section)
    T3C (trailer for B3=1 section)
  T3B (trailer for B2=2 section)
T3A (trailer for B1=1 section)
H3A (header for B1=2 section)
  H3B (header for B2=2 section)
    H3C (header for B3=0 section)
      data records for B1=2, B2=2, B3=0 (new B1, B2, and B3 section)
    T3C (trailer for B3=0 section)
    H3C (header for B3=1 section)
      data records for B1=2, B2=2, B3=1 (new B3 section)
    T3C (trailer for B3=1 section)
  T3B (trailer for B2=2 section)
T3A (trailer for B1=2 section)
```

p,m

specifies a section break field in the OUTFIL input records to be used to divide the report into sections. Each set of sequential OUTFIL input records, with the same binary value for the section break field, results in a corresponding set of data records. Each such set of data records is treated as a section in the report. A break is said to occur when the binary value changes.

p See p under HEADER1.

m See m under HEADER1.

SKIP

```
►►──SKIP=──┬─P──┬──────────────────────────────────────────────────►◄
           ├─L──┤
           └─nL─┘
```

Specifies, for reports produced for this OUTFIL group, that either:

- Each section for the associated section break field is to appear on a new page, or
- One or more blank lines is to appear after each section associated with this section break field, when it is followed by another section on the same page.

Thus, you can use SKIP to specify how sections will be separated from each other.

P specifies that each section is to appear on a new page.

L specifies that one blank line is to appear between sections on the same page. L is the same as 1L.

nL specifies that n blank lines are to appear between sections on the same page. You can specify from 1 to 255 for n.

Sample Syntax:

```
OUTFIL FNAMES=(PRINT,TAPE),
   SECTIONS=(10,20,SKIP=P,
             42,10,SKIP=3L)
```

HEADER3

```
                  ┌──,──────────────────────────────────┐
                  ▼                                     │
►►─HEADER3=─(──────┬──────┬──────r──────────────────────┴──)──────────────────►◄
                   └─c:───┘      │
                                 ├─p,m──────────────────┤
                                 │ ┌─DATE─────────────┐ │
                                 │ ├─&DATE────────────┤ │
                                 │ └──┬─DATE=──┬─(abcd)─┘ │
                                 │    └─&DATE=─┘         │
                                 │ ┌─TIME─────────────┐ │
                                 │ ├─&TIME────────────┤ │
                                 │ └──┬─TIME=──┬─(abc)──┘ │
                                 │    └─&TIME=─┘         │
                                 └─┬─PAGE───┬────────────┘
                                   └─&PAGE──┘
```

Specifies the section header to be used with the associated section break field for the reports produced for this OUTFIL group. The section header appears before the first data record of each section. DFSORT uses ASA carriage control characters to control page ejects and the placement of the lines in your report, according to your specifications.

You can choose to include any or all of the following report elements in your section header:

- Blanks and character strings
- Unedited input fields from the first OUTFIL input record for which a data record appears in the section
- Current date
- Current time
- Page number.

The section header consists of the elements you select, in the order in which you specify them, and in the columns or lines you specify.

c: See c: under HEADER1.

r See r under HEADER1.

p,m
specifies that an unedited input field, from the first OUTFIL input record for which a data record appears in the section, is to appear in the report record. See p,m under HEADER1 for further details.

DATE
See DATE under HEADER1.

&DATE
&DATE can be used instead of DATE. See &DATE under HEADER1.

DATE=(abcd)
See DATE=(abcd) under HEADER1.

&DATE=(abcd)
&DATE=(abcd) can be used instead of DATE=(abcd). See &DATE=(abcd) under HEADER1.

TIME
See TIME under HEADER1.

&TIME
&TIME can be used instead of TIME. See &TIME under HEADER1.

TIME=(abc)
See TIME=(abc) under HEADER1.

&TIME=(abc)
&TIME=(abc) can be used instead of TIME=(abc). See &TIME=(abc) under HEADER1.

PAGE
specifies that the current page number is to appear in the OUTFIL report record. The page number for the header appears as 6 digits, right-justified, with leading zeros suppressed. For example, if the page is numbered 3, it appears as ' 3'.

&PAGE
&PAGE can be used instead of PAGE.

Sample Syntax:

```
OUTFIL FNAMES=STATUS1,
         HEADER2=(10:'Status Report for all departments',5X,
                        '- ',&PAGE,' -'),
         SECTIONS=(10,8,
          HEADER3=(2/,10:'Report for department ',10,8,' on ',&DATE,2/,
                   10:'   Number',25:'Average Time',/,
                   10:'Completed',25:'   (in days)',/,
                   10:'---------',25:'------------')),
            OUTREC=(10:21,5,ZD,M10,LENGTH=9,
                       25:38,4,ZD,EDIT=(III.T),LENGTH=12,
                       132:X)
```

PAGEHEAD

```
►►──PAGEHEAD──────────────────────────────────────────────────────────►◄
```

Specifies that the section header to be used with the associated section break field is to appear at the top of each page of the report, except for the report header page (if any) and report trailer page (if any), as well as before each section. If you do not specify PAGEHEAD, the section header appears only before each section; so if a section is split between pages, the section header appears only in the middle of the page. PAGEHEAD can be used when you want HEADER3 to be used as a page header as well as a section header.

If PAGEHEAD is specified for a section break field for which HEADER3 is not also specified, PAGEHEAD will not be used.

Sample Syntax:

```
OUTFIL FNAMES=STATUS2,
   HEADER2=(10:'Status Report for all departments',5X,
                '- ',&PAGE,' -'),
   SECTIONS=(10,8,
    HEADER3=(2/,10:'Report for department ',10,8,' on ',&DATE,2/,
             10:'   Number',25:'Average Time',/,
             10:'Completed',25:'   (in days)',/,
             10:'---------',25:'------------'),
```

```
   PAGEHEAD,SKIP=P),
OUTREC=(10:21,5,ZD,M10,LENGTH=9,
        25:38,4,ZD,EDIT=(III.T),LENGTH=12,
        132:X)
```

TRAILER3

```
                  ┌─,──────────────────────────────────────────────────┐
                  ▼                                                    │
►►─TRAILER3=─(─┬──────┬──┬─r──────────────────────────────────────────┬─)──────►◄
               └─c:───┘  ├─p,m────────────────────────────────────────┤
                         ├─┬─DATE──────────────────────────────────┬──┤
                         │ ├─&DATE─────────────────────────────────┤  │
                         │ └─┬─DATE=──┬─(abcd)─┘                   │  │
                         │   └─&DATE=─┘                            │  │
                         ├─┬─TIME──────────────────────────────────┬──┤
                         │ ├─&TIME─────────────────────────────────┤  │
                         │ └─┬─TIME=──┬─(abc)─┘                    │  │
                         │   └─&TIME=─┘                            │  │
                         ├─┬─PAGE──┬──────────────────────────────┤  │
                         │ └─&PAGE─┘                               │  │
                         ├─┬─COUNT──────────────────────────────┬──┤  │
                         │ ├─COUNT15──────────────────────────┤  │  │
                         │ └─COUNT=(edit)─┘                     │  │  │
                         ├─┬─SUBCOUNT───────────────────────────┬──┤  │
                         │ ├─SUBCOUNT15───────────────────────┤  │  │
                         │ └─SUBCOUNT=(edit)─┘                  │  │  │
                         ├─┬─TOTAL=──┬──(p,m,f─┬──────┬─)──────┤  │  │
                         │ └─TOT=────┘          └─,edit─┘        │  │  │
                         ├─MIN=──(p,m,f─┬──────┬─)──────────────┤  │  │
                         │              └─,edit─┘                │  │  │
                         ├─MAX=──(p,m,f─┬──────┬─)──────────────┤  │  │
                         │              └─,edit─┘                │  │  │
                         ├─AVG=──(p,m,f─┬──────┬─)──────────────┤  │  │
                         │              └─,edit─┘                │  │  │
                         ├─┬─SUBTOTAL=──┬──(p,m,f─┬──────┬─)────┤  │  │
                         │ ├─SUBTOT=────┤          └─,edit─┘      │  │  │
                         │ └─SUB=───────┘                         │  │  │
                         ├─SUBMIN=──(p,m,f─┬──────┬─)───────────┤  │  │
                         │                 └─,edit─┘             │  │  │
                         ├─SUBMAX=──(p,m,f─┬──────┬─)───────────┤  │  │
                         │                 └─,edit─┘             │  │  │
                         └─SUBAVG=──(p,m,f─┬──────┬─)───────────┘  │  │
                                           └─,edit─┘                │  │
```

Specifies the section trailer to be used with the associated section break field for the reports produced for this OUTFIL group. The section trailer appears after the last data record of each section. DFSORT uses ASA carriage control characters to control page ejects and the placement of the lines in your report, according to your specifications.

You can choose to include any or all of the following report elements in your section trailer:

- Blanks and character strings
- Unedited input fields from the last OUTFIL input record for which a data record appears in the section
- Current date
- Current time

- Page number
- Any or all of the following statistics:
  - Count of data records in the section
  - Total, minimum, maximum, or average for each specified ZD, PD, BI, FI, or CSF/FS numeric input field in the data records in the section, edited to contain signs, decimal points, leading zeros or no leading zeros, and so on
  - Running total, minimum, maximum, or average for each specified ZD, PD, BI, FI, or CSF/FS numeric input field in the data records up to this point, edited to contain signs, decimal points, leading zeros or no leading zeros, and so on.

The section trailer consists of the elements you select, in the order in which you specify them, and in the columns or lines you specify.

c: See c: under HEADER1.

r See r under TRAILER1.

p,m
specifies that an unedited input field, from the last OUTFIL input record for which a data record appears in the section, is to appear in the report record. See p,m under TRAILER1 for further details.

DATE
See DATE under HEADER1.

&DATE
&DATE can be used instead of DATE. See &DATE under HEADER1.

DATE=(abcd)
See DATE=(abcd) under HEADER1.

&DATE=(abcd)
&DATE=(abcd) can be used instead of DATE=(abcd). See &DATE=(abcd) under HEADER1.

TIME
See TIME under HEADER1.

&TIME
&TIME can be used instead of TIME. See &TIME under HEADER1.

TIME=(abc)
See TIME=(abc) under HEADER1.

&TIME=(abc)
&TIME=(abc) can be used instead of TIME=(abc). See &TIME=(abc) under HEADER1.

PAGE
See PAGE under TRAILER1.

&PAGE
&PAGE can be used instead of PAGE. See &PAGE under TRAILER1.

COUNT
specifies that the count of data records in the section is to appear in the report record as 8 digits, right-justified, with leading zeros suppressed. For example, if the first section has 40 input records, the second section has 40 input records, and the third section has 26 input records,

COUNT will show ' 40' for the first section, ' 40' for the second section, and ' 26' for the third section.

COUNT counts input records, not data records. However, unless slash (/) is used to produce multiple records, the count will also represent the number of data records.

COUNT15
same as COUNT except that the count appears as 15 digits.

COUNT=(edit)
same as COUNT except that the 15–digit count appears editied as specified. See p,m,f,edit under OUTREC for further details on the edit fields you can use.

SUBCOUNT
specifies that the running count of input records up to this point in the report is to appear in the report record 8 digits, right-justified, with leading zeros suppressed. The running count accumulates the count for all sections up to and including the current section. For example, if the first section has 40 input records, the second section has 40 input records, and the third section has 26 input records, SUBCOUNT will show ' 40' for the first section, ' 80' for the second section, and ' 106' for the third section.

SUBCOUNT counts input records, not data records. However, unless slash (/) is used to produce multiple records, the count will also represent the number of data records.

SUBCOUNT15
same as SUBCOUNT except that the running count appears as 15 digits.

SUBCOUNT=(edit)
same as SUBCOUNT except that the 15–digit running count appears editied as specified. See p,m,f,edit under OUTREC for further details on the edit fields you can use.

TOTAL
specifies that an edited total, for the values of a numeric input field in the data records in the section, is to appear in the report record.

TOT can be used instead of TOTAL.

p,m,f,edit
See p,m,f,edit under TOTAL for TRAILER1.

MIN
specifies that an edited minimum, for the values of a numeric input field in the data records in the section, is to appear in the report record.

p,m,f,edit
See p,m,f,edit under MIN for TRAILER1.

MAX
specifies that an edited maximum, for the values of a numeric input field in the data records in the section, is to appear in the report record.

p,m,f,edit
See p,m,f,edit under MAX for TRAILER1.

AVG
specifies that an edited average, for the values of a numeric input field in the data records in the section, is to appear in the report record.

p,m,f,edit
See p,m,f,edit under AVG for TRAILER1.

SUBTOTAL
specifies that an edited running total, for the values of a numeric input field in the data records up to this point in the report, is to appear in the report record. The running total accumulates the total for all sections up to and including the current section. For example, if the total for a selected numeric field is +200 for the first section, -250 for the second section and +90 for the third section, SUBTOTAL will be +200 for the first section, -50 for the second section and +40 for the third section.

SUBTOT or SUB can be used instead of SUBTOTAL.

p,m,f,edit
See p,m,f,edit under SUBTOTAL for TRAILER1.

SUBMIN
specifies that an edited running minimum, for the values of a numeric input field in the data records up to this point in the report, is to appear in the report record. The running minimum selects the minimum from all sections up to and including the current section. For example, if the minimum for a selected numeric field is +200 for the first section, -250 for the second section and +90 for the third section, SUBMIN will be +200 for the first section, -250 for the second section and -250 for the third section.

p,m,f,edit
See p,m,f,edit under SUBMIN for TRAILER1.

SUBMAX
specifies that an edited running maximum, for the values of a numeric input field in the data records up to this point in the report, is to appear in the report record. The running maximum selects the maximum from all sections up to and including the current section. For example, if the maximum for a selected numeric field is -100 for the first section, +250 for the second section and +90 for the third section, SUBMAX will be -100 for the first section, +250 for the second section and +250 for the third section.

p,m,f,edit
See p,m,f,edit under SUBMAX for TRAILER1.

SUBAVG
specifies that an edited running average, for the values of a numeric input field in the data records up to this point in the report, is to appear in the report record. The running average computes the average for all sections up to and including the current section. For example, if the count of data records and total for a selected numeric field are 60 and +2205 for the first section, respectively, 60 and -6252 for the second section, respectively, and 23 and -320 for the third section, respectively, SUBAVG will be +36 for the first section, -33 for the second section and -30 for the third section.

p,m,f,edit
See p,m,f,edit under SUBAVG for TRAILER1.

Sample Syntax:

```
OUTFIL FNAMES=SECRPT,
  INCLUDE=(11,4,CH,EQ,C'SSD'),
  SECTIONS=(3,5,SKIP=P,
   HEADER3=(2:'Department: ',3,5,4X,'Date: ',&DATE,2/),
   TRAILER3=(2/,2:'The average for ',3,5,' is ',
                  AVG=(40,3,PD,M12),/,
                2:'The overall average so far is ',
                  SUBAVG=(40,3,PD,M12)),
  45,8,SKIP=3L,
   HEADER3=(4:'Week Ending ',45,8,2/,
               4:'Item Number',20:'Completed',/,
               4:'-',20:'-'),
   TRAILER3=(4:'The item count for week ending ',45,8,
      ' is ',COUNT=(EDIT=(II,IIT)))),
   OUTREC=(11:16,4,22:40,3,PD,M12,120:X)
```

Default for SECTIONS: None; must be specified.

NODETAIL

```
►►──NODETAIL──────────────────────────────────────────────────────────►◄
```

Specifies that data records are not to be output for the reports produced for this OUTFIL group. With NODETAIL, the data records are completely processed with respect to input fields, statistics, counts, sections breaks, and so on, but are not written to the OUTFIL data set and are not included in line counts for determining the end of a page. You can use NODETAIL to summarize the data records without actually showing them.

Sample Syntax:

```
OUTFIL FNAMES=SUMMARY,NODETAIL,
   HEADER2=(' Date: ',DATE=(DMY.),4X,'Page: ',PAGE,2/,
             10:'Division',25:'    Total Revenue',/,
             10:'--------',25:'-----------------'),
   SECTIONS=(3,5,
    TRAILER3=(10:3,5,
              25:TOTAL=(25,4,FI,M19,
                 LENGTH=17))),
    TRAILER1=(5/,10:'Summary of Revenue ',4/,
                 12:'Number of divisions reporting is ',
                     COUNT,/,
                 12:'Total revenue is ',
                    TOTAL=(25,4,FI,M19))
```

Default for NODETAIL: None; must be specified.

Default for OUTFIL Statements: None; must be specified. Multiple OUTFIL statements can be specified in the same and different sources; override is at the ddname level.

Applicable Functions for OUTFIL Statements: Sort, merge, and copy.

OUTFIL Statements Notes

- OUTFIL processing is supported for sort, merge, and copy applications, but only by the Blockset technique.
- The ODMAXBF value in effect specifies the maximum buffer spaceto be used for each OUTFIL data set. The ODMAXBF value can be specified as an installation or run-time parameter, or in an ICEIEXIT routine. The default value of 2M is

recommended for the ODMAXBF option in effect. Lowering ODMAXBF can cause performance degradation for the application,but might be necessary if you consider the amount of storage used for OUTFIL processing to be a problem. Raising ODMAXBF can improve EXCPs for the application, but can also increase the amount of storage needed.

- The storage used for OUTFIL processing will be adjusted automatically according to the total storage available, the storage needed for non-OUTFIL processing, and the number of OUTFIL data sets and their attributes (for example, block size). OUTFIL processing will be subject to the ODMAXBF limit in effect and the system storage limits(for example, IEFUSI), but not to the DFSORT storage limits (that is, SIZE, MAXLIM, and TMAXLIM). DFSORT attempts to use storage above 16MB virtual for OUTFIL processing whenever possible.
- The VSAMBSP option applies to SORTOUT data sets, but not to OUTFIL data sets. The NOBLKSET option will be ignored if OUTFIL data sets are being processed. An E39 exit routine is entered for the SORTOUT data set, but not for OUTFIL data sets.
- For fixed-format OUTFIL data sets: DFSORT will determine each OUTFIL data set LRECL based on the length of the OUTREC records for the group, or the length of the OUTFIL input records (if OUTREC is not specified for the group). For VSAM data sets, the maximum record size defined in the cluster is equivalent to the LRECL.

  If an OUTFIL data set LRECL is not specified or available, DFSORT will set it to the determined LRECL. If an OUTFIL data set LRECL is specified or available, it must not be less than the determined LRECL, or more than the determined LRECL if the OUTREC parameter is specified. In other words, the LRECL value cannot be used to pad the output records, or to truncate the records produced by OUTREC parameter processing.

  In general, OUTREC processing should be used to pad or truncate the records, and the LRECL should either not be specified or set to the length of the reformatted records.
- For variable-format OUTFIL data sets: DFSORT will determine each OUTFIL data set maximum LRECL based on the length of the OUTREC records for the group, or the length of the OUTFIL input records (if OUTREC is not specified for the group). If an OUTFIL data set maximum LRECL is not specified or available, DFSORT will set it to the determined maximum LRECL. For VSAM data sets, the maximum record size defined in the cluster is four bytes more than the maximum LRECL.
- When you create an OUTFIL report, the length for the longest or onlydata record must be equal to or greater than the maximum report record length. You can use the OUTREC parameter to force a length for the data records that is longer than any report record; you can then either let DFSORT compute and set the LRECL, or ensure that the computed LRECL is equal to the existing or specified LRECL. Remember to allow an extra byte in the LRECL for the ASA carriage control character.

  For example, if your data records are 40 bytes, but your longest report record is 60 bytes, you can use an OUTREC parameter such as:

  ```
  OUTREC=(1,40,80:X)
  ```

  DFSORT will then set the LRECL to 81 (1 byte for the ASA carriage control character plus 80 bytes for the length of the data records), and pad the data records with blanks on the right.

  System errors can result if you print an OUTFIL report containing records longer than your printer can handle.

- DFSORT uses appropriate carriage controls (for example, C'-' for triple space) in header and trailer records when possible to reduce the number of report records written. DFSORT always uses the single space carriage control (C' ') in data records. Although these carriage control characters may not be shown when you view an OUTFIL data set (depending on how you view it), they will be used if you print the report. If you are creating a report for viewing and want blank lines to appear in headers and trailers, specify a line of blanks instead of using n/. For example, instead of specifying:

  ```
  OUTFIL FNAMES=RPT,
   HEADER2=(2/,'start of header',2/,'next line')
  ```

  which will result in blank lines for the printer, but not for viewing, specify:

  ```
  OUTFIL FNAMES=RPT,
   HEADER2=(X,/,X,/,'start of header',/,X,/,'next line')
  ```

- When defining variable-length OUTFIL output or data records with OUTREC, you must explicitly specify the 4-byte RDW at the beginning of each record. When using / in OUTREC, you must explicitly specify the 4-byte RDW at the beginning of each new output or data record.

  When defining variable-length OUTFIL header or trailer records, you must not specify the 4-byte RDW at the beginning of the record.

- For variable-length OUTFIL records, if the variable part of the record is specified as the last field of an INREC or OUTREC statement, then the variable part of the record must be specified as the last field of all OUTFIL OUTREC records. If the variable part of the record is not specified as the last field of an INREC or OUTREC statement, then the variable part of the record must not be specified for any OUTFIL OUTREC record. If INREC and OUTREC statements are not specified, then the variable part of the record can be specified or not specified independently for OUTFIL OUTREC records.
- If there are no OUTFIL input records for an OUTFIL group, the headers and trailers appear without any data records. Blanks will be used for any specified unedited input fields, and zero values will be used for any specified statistics fields.
- If a variable-length OUTFIL input record is too short to contain a specified unedited input field for a report header or trailer, blanks will be used for the missing bytes. If a variable-length OUTFIL input record is too short to contain a specified section break field or statistics field, zeros will be used for the missing bytes, intentionally or unintentionally.
- If a variable-length OUTFIL input record is too short to contain an OUTFIL INCLUDE or OMIT compare field, the action DFSORT takes depends on the settings for VLSCMP/NOVLSCMP and VLSHRT/NOVLSHRT. For details, see the discussion of the VLSCMP and NOVLSCMP options in “OPTION Control Statement” on page 134.
- If a variable-length OUTFIL input record is too short to contain an OUTFIL OUTREC field, DFSORT will terminate unless the VLFILL=byte parameter is specified.
- If a variable-length OUTFIL output data record is longer than the LRECL of its OUTFIL data set, the action DFSORT takes depends on the settings for VLLONG/NOVLLONG. For details, see the discussion of the VLLONG and NOVLLONG options in “OPTION Control Statement” on page 134. Note that VLLONG can be used to truncate long OUTFIL data records, but has no effect on long OUTFIL header or trailer records.
- If a page number overflows 6 digits (PAGE, &PAGE), a count or running count overflows 15 digits (COUNT, COUNT15, SUBCOUNT, SUBCOUNT15, AVG,

SUBAVG), or a total or running total overflows 15 digits (TOTAL, SUBTOTAL, AVG, SUBAVG), the overflowing value will be truncated to the number of digits allowed, intentionally or unintentionally.

- Multiple OUTFIL statements can be specified in the same and different sources. If a ddname occurs more than once in the same source, the ddname is associated with the first OUTFIL group in which it appears. For example, if the following is specified in SYSIN:

  ```
  OUTFIL FNAMES=(OUT1,OUT2),INCLUDE=(1,1,CH,EQ,C'A')
  OUTFIL FNAMES=(OUT3,OUT1),SAVE
  ```

  OUT1 and OUT2 are processed as part of the first OUTFIL group, that is, with INCLUDE. OUT3 is processed as part of the second OUTFIL group, that is, with SAVE; but OUT1 is not because it is a duplicate ddname.

  If a ddname occurs in more than one source, the ddname is associated with the highest source OUTFIL group in which it appears. For example, if the following is specified in DFSPARM:

  ```
  OUTFIL FNAMES=(OUT1,OUT2),INCLUDE=(1,1,CH,EQ,C'A')
  ```

  and the following is specified in SYSIN:

  ```
  OUTFIL FNAMES=(OUT3,OUT1),SAVE
  ```

  OUT1 and OUT2 are processed as part of the DFSPARM OUTFIL group, that is, with INCLUDE. OUT3 is processed as part of the SYSIN OUTFIL group, that is, with SAVE; but OUT1 is not because it is an overridden ddname.
- OUTFIL statements cannot be passed to or returned from an EFS program. The D2 format cannot be specified in the INCLUDE or OMIT parameter of an OUTFIL statement.
- If SZERO is in effect, -0 compares as less than +0 when numeric fields and constants are used. If NOSZERO is in effect, -0 compares as equal to +0 when numeric fields and constants are used.

  Note: OPTION SZERO or OPTION NOSZERO is ignored for OUTFIL INCLUDE=(...) or OUTFIL OMIT=(...) if the OPTION statement is ″found″ **before** the OUTFIL statement in the same source. For example:

  ```
  //SYSIN DD *
   OPTION NOSZERO,COPY
   OUTFIL INCLUDE=(...)
  /*
  ```

OUTFIL Features—Examples

Example 1

```
OPTION COPY
OUTFIL INCLUDE=(15,6,CH,EQ,C'MSG005'),FNAMES=M005
OUTFIL INCLUDE=(15,6,CH,EQ,C'MSG022'),FNAMES=M022
OUTFIL INCLUDE=(15,6,CH,EQ,C'MSG028'),FNAMES=M028
OUTFIL INCLUDE=(15,6,CH,EQ,C'MSG115'),FNAMES=M115
OUTFIL SAVE,FNAMES=UNKNOWN
```

This example illustrates how records can be distributed to different OUTFIL data sets based on criteria you specify:

- Input records with MSG005 in bytes 15 through 20 will be written to the OUTFIL data set associated with ddname M005.

- Input records with MSG022 in bytes 15 through 20 will be written to the OUTFIL data set associated with ddname M022.
- Input records with MSG028 in bytes 15 through 20 will be written to the OUTFIL data set associated with ddname M028.
- Input records with MSG115 in bytes 15 through 20 will be written to the OUTFIL data set associated with ddname M115.
- Input records with anything else in bytes 15 through 20 will be written to the OUTFIL data set associated with ddname UNKNOWN

Example 2

```
SORT FIELDS=(18,5,ZD,D)
OUTFIL FNAMES=(V,VBU1,VBU2)
OUTFIL FNAMES=(F,FBU1),
  CONVERT,OUTREC=(11,3,X,18,5,X,X'0000000F')
```

This example illustrates how multiple sorted output data sets can be created and how a variable-length record data set can be converted to a fixed-length record data set:

- The first OUTFIL statement writes the variable-length input records to the variable-length OUTFIL data sets associated with ddnames V, VBU1, and VBU2.
- The second OUTFIL statement reformats the variable-length input records to fixed-length output records and writes them to the fixed-length OUTFIL data sets associated with ddnames F and FBU1. CONVERT is used to indicate that a variable-length data set is to be converted to a fixed-length data set; OUTREC is used to describe how the variable-length input records are to be reformatted as fixed-length output records.

Example 3

```
SORT FIELDS=(15,6,ZD,A)
OUTFIL FNAMES=USA,
   HEADER2=(5:'Parts Completion Report for USA',2/,
            5:'Printed on ',DATE,
              ' at ',TIME=(12:),3/,
            5:'Part  ',20:'Completed',35:'   Value ($)',/,
            5:'------',20:'---------',35:'------------'),
   OUTREC=(5:15,6,ZD,M11,
           20:3,4,ZD,M12,LENGTH=9,
           35:38,8,ZD,M18,LENGTH=12,
           132:X)
OUTFIL FNAMES=FRANCE,
   HEADER2=(5:'Parts Completion Report for France',2/,
            5:'Printed on ',DATE=(DM4/),
              ' at ',TIME,3/,
            5:'Part  ',20:'Completed',35:'   Value (F)',/,
            5:'------',20:'---------',35:'------------'),
   OUTREC=(5:15,6,ZD,M11,
           20:3,4,ZD,M16,LENGTH=9,
           35:38,8,ZD,M22,LENGTH=12,
           132:X)
OUTFIL FNAMES=DENMARK,
   HEADER2=(5:'Parts Completion Report for Denmark',2/,
            5:'Printed on ',DATE=(DMY-),
              ' at ',TIME=(24.),3/,
            5:'Part  ',20:'Completed',35:'  Value (kr)',/,
            5:'------',20:'---------',35:'------------'),
   OUTREC=(5:15,6,ZD,M11,
           20:3,4,ZD,M13,LENGTH=9,
           35:38,8,ZD,M19,LENGTH=12,
           132:X)
```

This example illustrates how reports for three different countries can be produced from sorted fixed-length input records. The reports differ only in the way that date, time, and numeric formats are specified:

1. The first OUTFIL statement produces a report that has the date, time, and numeric formats commonly used in the United States.
2. The second OUTFIL statement produces a report that has the date, time, and numeric formats commonly used in France.
3. The third OUTFIL statement produces a report that has the date, time, and numeric formats commonly used in Denmark.

Of course, any one of the three reports can be produced by itself using a single OUTFIL statement instead of three OUTFIL statements. (This may be necessary if you are sorting character data according to a specified locale for that country.)

The FNAMES parameter specifies the ddname (USA, FRANCE, DENMARK) associated with the fixed-length data set for that report.

The HEADER2 parameter specifies the page header to appear at the top of each page for that report, which will consist of:

- A line of text identifying the report. Note that all English text in the report can be replaced by text in the language of that country.
- A blank line (2/).
- A line of text showing the date and time. Note that variations of the DATE, DATE=(abcd), TIME, and TIME=(abc) operands are used to specify the date and time in the format commonly used in that country.

- Two blank lines (3/).
- Two lines of text showing headings for the columns of data. Note that the appropriate currency symbol can be included in the text.

The OUTREC parameter specifies the three columns of data to appear for each input record as follows:

- A 6-byte edited numeric value produced by transforming the ZD value in bytes 15 through 20 according to the pattern specified by M11. M11 is a pattern for showing integers with leading zeros.
- A 9-byte (LENGTH=9) edited numeric value produced by transforming the ZD value in bytes 3 through 6 according to the pattern for integer values with thousands separators commonly used in that country. M12 uses a comma for the thousands separator. M16 uses a blank for the thousands separator. M13 uses a period for the thousands separator.
- A 12-byte (LENGTH=12) edited numeric value produced by transforming the ZD value in bytes 38 through 45 according to the pattern for decimal values with thousands separators and decimal separators commonly used in that country. M18 uses a comma for the thousands separator and a period for the decimal separator. M22 uses a blank for the thousands separator and a comma for the decimal separator. M19 uses a period for the thousands separator and a comma for the decimal separator.

Table 28 on page 199 shows the 26 pre-defined edit masks (M0-M25) you can choose from.

132:X is used at the end of the OUTREC parameter to ensure that the data records are longer than the report records. This will result in an LRECL of 132 for the fixed-length OUTFIL data sets (1 byte for the ASA control character and 131 bytes for the data).

The three reports might look as follows:

```
Parts Completion Report for USA

Printed on 03/25/95 at 01:56:20 pm


Part           Completed         Value ($)
------         ---------      ------------
000310               562          8,317.53
001184             1,234         23,456.78
029633                35            642.10
192199             3,150        121,934.65
821356               233          2,212.34
```

```
Parts Completion Report for France

Printed on 25/03/1995 at 13:56:20


Part           Completed         Value (F)
------         ---------      ------------
000310               562          8 317,53
001184             1 234         23 456,78
029633                35            642,10
192199             3 150        121 934,65
821356               233          2 212,34
```

```
Parts Completion Report for Denmark

Printed on 25-03-95 at 13.56.20


Part           Completed        Value (kr)
------         ---------      ------------
000310               562          8.317,53
001184             1.234         23.456,78
029633                35            642,10
192199             3.150        121.934,65
821356               233          2.212,34
```

Example 4

```
SORT FIELDS=(3,10,A,16,13,A),FORMAT=CH
OUTFIL FNAMES=WEST,
  INCLUDE=(42,6,CH,EQ,C'West'),
  HEADER1=(5/,18:'    Western Region',3/,
              18:'Profit and Loss Report',3/,
              18:'     for  ',&DATE,3/,
              18:'      Page',&PAGE),
  OUTREC=(6:16,13,24:31,10,ZD,M5,LENGTH=20,75:X),
  SECTIONS=(3,10,SKIP=P,
    HEADER3=(2:'Division:  ',3,10,5X,'Page:',&PAGE,2/,
             6:'Branch Office',24:'       Profit/(Loss)',/,
             6:'-------------',24:'--------------------'),
   TRAILER3=(6:'=============',24:'====================',/,
             6:'Total',24:TOTAL=(31,10,ZD,M5,LENGTH=20),/,
             6:'Lowest',24:MIN=(31,10,ZD,M5,LENGTH=20),/,
             6:'Highest',24:MAX=(31,10,ZD,M5,LENGTH=20),/,
             6:'Average',24:AVG=(31,10,ZD,M5,LENGTH=20),/,
             3/,2:'Average for all Branch Offices so far:',
                X,SUBAVG=(31,10,ZD,M5))),
    TRAILER1=(8:'Page ',&PAGE,5X,'Date:  ',&DATE,5/,
              8:'Total Number of Branch Offices Reporting:  ',
                COUNT,2/,
              8:'Summary of Profit/(Loss) for all',
                ' Western Division Branch Offices',2/,
              12:'Total:',
                  22:TOTAL=(31,10,ZD,M5,LENGTH=20),/,
              12:'Lowest:',
                  22:MIN=(31,10,ZD,M5,LENGTH=20),/,
              12:'Highest:',
                  22:MAX=(31,10,ZD,M5,LENGTH=20),/,
              12:'Average:',
                  22:AVG=(31,10,ZD,M5,LENGTH=20))
```

This example illustrates how a report can be produced with a header and trailer page and sections of columns of data, from a sorted subset of fixed-length input records.

The FNAMES parameter specifies the ddname (WEST) associated with the fixed-length data set for the report.

The INCLUDE parameter specifies the records to be selected for the report.

The HEADER1 parameter specifies the report header to appear as the first page of the report, which will consist of five blank lines (5/) followed by four lines of text, each separated by 2 blank lines (3/). The last two lines of text will show the date (&DATE) and page number (&PAGE), respectively.

The OUTREC parameter specifies the two columns of data to appear for each selected input record as follows:

- The character string from bytes 16 through 28 of the input record.
- A 20-byte (LENGTH=20) edited numeric value produced by transforming the ZD value in bytes 31 through 40 according to the pattern specified by M5.

The SECTIONS parameter specifies the section break field (3,10), page ejects between sections (SKIP=P), the header (HEADER3) to appear before each section and the trailer (TRAILER3) to appear after each section. The section header will consist of a line of text showing the page number, a blank line (2/) and two lines of text showing the headings for the columns of data. The section trailer will consist of a line of text separating the data from the trailer, lines of text showing the total (TOTAL), minimum (MIN), maximum (MAX) and average (AVG) for the data in the section as edited numeric values, two blank lines, and a line of text showing the running average (SUBAVG) for all of the data records in the report up to this point.

The TRAILER1 parameter specifies the report trailer to appear as the last page of the report, which will consist of a line of text showing the page and date, four blank lines (5/), a text line showing the count of data records in the report, a blank line, a line of text, a blank line, and lines of text showing the total, minimum maximum and average for all of the data in the report as edited numeric values.

75:X is used at the end of the OUTREC parameter to ensure that the data records are longer than the report records. This will result in an LRECL of 76 for the fixed-length OUTFIL data set (1 byte for the ASA control character and 75 bytes for the data).

The report might look as follows:

```
                    Western Region

                Profit and Loss Report

                     for  05/11/95

                      Page     1
```

```
Division:  Chips          Page:     2

      Branch Office            Profit/(Loss)
      -------------     --------------------
      Gilroy                     554,843.42
      Los Angeles                (22,340.14)
      Morgan Hill                987,322.32
      Oakland                    234,124.32
      San Francisco              (32,434.31)
      San Jose                 1,232,133.35
      San Martin                 889,022.03
      =============     ====================
      Total                    3,842,670.99
      Lowest                     (32,434.31)
      Highest                  1,232,133.35
      Average                    548,952.99


  Average for all Branch Offices so far:      548,952.99
```

```
Division:  Ice Cream      Page:     3

      Branch Office            Profit/(Loss)
      -------------     --------------------
      Marin                      542,341.23
      Napa                       857,342.83
      San Francisco              922,312.45
      San Jose                       (234.55)
      San Martin               1,003,467.30
      =============     ====================
      Total                    3,325,229.26
      Lowest                         (234.55)
      Highest                  1,003,467.30
      Average                    665,045.85


  Average for all Branch Offices so far:      597,325.02
```

```
Division:  Pretzels       Page:     4

      Branch Office            Profit/(Loss)
      -------------     --------------------
      Marin                    5,343,323.44
      Morgan Hill                843,843.40
      Napa                     5,312,348.56
      San Francisco            5,412,300.05
      San Jose                 1,234,885.34
      San Martin                  (2,343.82)
      =============     ====================
      Total                   18,144,356.97
      Lowest                      (2,343.82)
      Highest                  5,412,300.05
      Average                  3,024,059.49


  Average for all Branch Offices so far:    1,406,236.51
```

```
 Page      5     Date:  05/11/95



Total Number of Branch Offices Reporting:        18

Summary of Profit/(Loss) for all Western Division Branch Offices

    Total:          25,312,257.22
    Lowest:            (32,434.31)
    Highest:         5,412,300.05
    Average:         1,406,236.51
```

Example 5

```
SORT FIELDS=(6,5,CH,A)
OUTFIL FNAMES=STATUS,
  HEADER2=(1:C'PAGE ',&PAGE,C' OF STATUS REPORT FOR ',&DATE,2/,
           6:C'ITEM ',16:C'STATUS      ',31:C'PARTS',/,
           6:C'-----',16:C'------------',31:C'-----'),
   OUTREC=(1,4,
           10:6,5,
           20:14,1,CHANGE=(12,
                           C'1',C'SHIP',
                           C'2',C'HOLD',
                           C'3',C'TRANSFER'),
                   NOMATCH=(C'*CHECK CODE*'),
           37:39,1,BI,M10,
           120:X)
```

This example illustrates how a report can be produced with a page header and columns of data from sorted variable-length input records, using a lookup table.

The FNAMES parameter specifies the ddname (STATUS) associated with the variable-length data set for the report.

The HEADER2 parameter specifies the page header to appear at the top of each page, which will consist of a line of text showing the page number (&PAGE) and date (&DATE), a blank line (2/), and two lines of text showing headings for the columns of data.

The OUTREC parameter specifies the RDW and three columns of data to appear for each input record as follows (remember that byte 5 is the first byte of data for variable-length records):

- The character string from bytes 6 through 10 of the input record
- A character string produced by finding a match for byte 14 of the input record in the table defined by CHANGE (lookup and change). NOMATCH indicates the character string to be used if byte 14 does not match any of the entries in the CHANGE table.
- An edited numeric value produced by transforming the BI value in byte 39 according to the pattern specified by M10.

With variable-length input records, you must account for the RDW when specifying the c: values for OUTREC, but not for headers or trailers. The 1: used for the first line of HEADER2 causes it to start in the first data byte (by contrast, 5: must be used to specify the first OUTREC data byte for variable-length records). Also, since 6: is used for the ITEM heading, 10: must be used for the ITEM data to get the heading and data to line up in columns.

120:X is used at the end of the OUTREC parameter to ensure that the data records are longer than the report records. This will result in a maximum LRECL of 121 for the variable-length OUTFIL data set (1 byte for the ASA control character and a maximum of 120 bytes for the data).

The first page of the printed report might start as follows:

```
PAGE      1 OF STATUS REPORT FOR 05/12/95

     ITEM      STATUS         PARTS
     -----     ------------   -----
     00082     HOLD              36
     00123     SHIP             106
     00300     *CHECK CODE*      95
     10321     TRANSFER          18
     12140     SHIP             120
```

Example 6

```
OPTION COPY
OUTFIL FNAMES=(PIPE1,PIPE2,PIPE3,PIPE4,PIPE5),SPLIT
```

This example illustrates how output records can be split as evenlyas possible among a set of SmartBatch pipes. The first record will be written to the writer associated with PIPE1, the second to PIPE2, the third to PIPE3, the fourth to PIPE4, the fifth to PIPE5, the sixth to PIPE1, and so on until all of the records have been written.

Of course, the records can be written to data sets as well as pipes.

Example 7

```
OPTION COPY
OUTFIL FNAMES=RANGE1,ENDREC=1000000
OUTFIL FNAMES=RANGE2,STARTREC=1000001,ENDREC=2000000
OUTFIL FNAMES=RANGE3,STARTREC=2000001,ENDREC=3000000
OUTFIL FNAMES=RANGE4,STARTREC=3000001,ENDREC=4000000
OUTFIL FNAMES=(RANGE5,EXTRA),STARTREC=4000001
```

This example illustrates how specific ranges of output records can be written to different output data sets. A typical application might be database partitioning.

The first 1 million records will be written to the data set associated with RANGE1, the second million to RANGE2, the third million to RANGE3, and the fourth million to RANGE4. The remaining records will be written to both the data set associated with RANGE5 and the data set associated with EXTRA (SAVE or STARTREC=4000001 will accomplish the same purpose in this case).

Note that the INCLUDE, OMIT, and SAVE parameters of OUTFIL can also be used to select records to be written to different output data sets, based on criteria you specify.

Example 8

```
OPTION COPY,Y2PAST
OUTFIL FNAMES=Y4,
       OUTREC=(1,19,
               21,2,PD0,M11,C'/',  transform mm
               22,2,PD0,M11,C'/',  transform dd
               20,2,Y2P,           transform yy to yyyy
               24,57)
```

This example illustrates how to transform an existing data set with a packed decimal date field of the form P'yymmdd' (X'0yymmddC') in bytes 20-23 into a new data set with a character date field of the form C'mm/dd/yyyy' in bytes 20-29. yy represents the two-digit year, yyyy represents the four-digit year, mm represents the month, dd represents the day, and C represents a positive sign.

The input data set has an LRECL of 80 and the Y4 data set will have an LRECL of 86.

The Y2PAST=26 option sets the century window to be used to transform two-digit years into four-digit years. If the current year is 1996, the century window will be 1970 to 2069. Using this century window, the input and output fields might be as follows:

```
Input Field (HEX)              Output Field (CH)
   20                             20
    |                              |
    0020505F                       05/05/2002
    0950823C                       08/23/1995
    0980316C                       03/16/1998
    0000316F                       03/16/2000
```

Example 9

```
OPTION COPY,Y2PAST=1996
OUTFIL FNAMES=SPCL,
       OUTREC=(1,14, copy positions 1-14
          15,6,Y2T,  transform yy to yyyy - allow blanks
          21,20)     copy positions 21 - 40
```

This example illustrates how to transform an existing data set with a character date field of the form C'yymmdd' and blank special indicators in bytes 15-20, into a new date set with a character date field of the form C'yyyymmdd' and blank special indicators in bytes 15-22.

The input data set has an LRECL of 40 and the SPCL data set will have an LRECL of 42.

The Y2PAST=1996 option sets the century window to 1996-2095. The century window will be used to transform the two-digit years into four-digit years, but will not be used for the special blank indicators.

The input records might be as follows:

```
MORGAN HILL                  CA
SAN JOSE            960512   CA
BOCA RATON          000628   FL
DENVER              951115   CO
```

The output records would be as follows:

```
MORGAN HILL                     CA
SAN JOSE            19960512    CA
BOCA RATON          20000628    FL
DENVER              20951115    CO
```

Example 10

```
OPTION COPY
OUTFIL FNAMES=ALL,OUTREC=(C'US ',1,10,C' is in ',11,15,/,
                          C'WW ',1,10,C' is in ',26,20,2/)
OUTFIL FNAMES=(US,WW),SPLIT,
                       OUTREC=(1,10,C' is in ',11,15,/,
                               1,10,C' is in ',26,20)
```

This example illustrates how multiple OUTFIL output and blank records can be produced from each OUTFIL input record. The input data set has an LRECL of 50 and contains the following three records:

```
Finance   San Francisco  Buenos Aires
Research  New York       Amsterdam
Marketing Los Angeles    Mexico City
```

The first OUTFIL statement creates the data set associated with ddname ALL. This data set will have an LRECL of 40 (the length of the longest output record; the one that includes the 26,20 input field). Each input record will result in two data records followed by two blank records as follows:

ALL data set

```
US Finance    is in San Francisco
WW Finance    is in Buenos Aires


US Research   is in New York
WW Research   is in Amsterdam


US Marketing  is in Los Angeles
WW Marketing  is in Mexico City
```

The second OUTFIL statement creates the two data sets associated with ddnames US and WW. These data sets will have an LRECL of 37 (the length of the longest output record; the one that includes the 26,20 input field). Each input record will result in two data records. SPLIT will cause the first data record to be written to the US data set and the second data record to be written to the WW data set. Thus, each input record will create one record in each OUTFIL data set as follows:

US data set

```
Finance    is in San Francisco
Research   is in New York
Marketing  is in Los Angeles
```

WW data set

```
Finance    is in Buenos Aires
Research   is in Amsterdam
Marketing  is in Mexico City
```

Example 11

```
SORT FIELDS=(6,3,CH,D)
OUTFIL FNAMES=SET60,OUTREC=(1,60),VLFILL=C' '
OUTFIL FNAMES=VTOF,CONVERT,OUTREC=(5,20,5X,28,20),VLFILL=C'*'
```

This example illustrates how variable-length records that are too short to contain all OUTFIL OUTREC fields can be processed successfully.

The input data set has RECFM=VB and LRECL=80. The records in this data set have lengths that vary from 15 bytes to 75 bytes.

The first OUTFIL statement creates the data set associated with ddname SET60. This data set will have RECFM=VB and LRECL=60. Every record in this data set will have a length of 60. The 1,60 field truncates records longer than 60 bytes to 60 bytes. Because VLFILL=C' ' is specified, the 1,60 field pads records shorter than 60 bytes to 60 bytes using a blank (C' ') as the fill byte.

Note: Without VLFILL=byte, this OUTFIL statement would terminate with an ICE218A message because some of the input records are too short to contain the OUTREC field.

The second OUTFIL statement creates the data set associated with ddname VTOF. This data set will have RECFM=FB and LRECL=45. CONVERT changes the variable-length input records to fixed-length output records according to the fields specified by OUTREC. VLFILL=C'*' allows short input records to be processed. Each missing byte in an OUTFIL OUTREC field is replaced with an asterisk (C'*') fill byte.

Note: VLFILL=C'*' overrides the default of VLFILL=X'40' for CONVERT.

Example 12

```
OPTION COPY
  OUTFIL OUTREC=(SEQNUM,4,BI,Z,8,5,ZD,TO=PD,Z,
                 31,2,PD,TO=FI,LENGTH=2)
```

This example illustrates how a sequence number can be generated and how values in one numeric format can be converted to another numeric format.

The input data set has an LRECL of 50 and the SORTOUT data set will have an LRECL of 11.

The OUTFIL statement creates output records with the following fields:

- A binary sequence number in bytes 1-4 that starts at 1 and increments by 1.
- X’00’ in byte 5.
- A PD field in bytes 6-8 containing the converted ZD field from input bytes 8-12
- X’00’ in position 9.
- An FI field in bytes 10-11 containing the converted PD field from input bytes 31-32.

OUTREC Control Statement

```
                        ┌──,──────────────────────────────┐
                        ▼                                 │
►►──OUTREC──FIELDS=──(──┬──────┬──┬──s──────────────────┬──)──────────────────►◄
                        └──c:──┘  ├──p,m──┬──────┬──────┤
                                  │       └──,a──┘      │
                                  ├──p──────────────────┤
                                  ├──p,m,HEX────────────┤
                                  ├──p,HEX──────────────┤
                                  ├──p,m,Y2x────────────┤
                                  ├──p,m,Y2x(c)─────────┤
                                  ├──p,m,Y2xP───────────┤
                                  ├──p,m,f──┬──────┬────┤
                                  │         └──,edit──┘ │
                                  ├──p,m,f,to───────────┤
                                  ├──p,m,Y2x,edit───────┤
                                  ├──p,m,Y2x,to─────────┤
                                  ├──p,m,lookup─────────┤
                                  └──seqnum─────────────┘
```

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 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.

The OUTREC control statement also provides for:

- Insertion of blanks, zeros, strings and sequence numbers before, between, and after the input fields in the reformatted records.
- Sophisticated editing and conversion capabilities, such as hexadecimal display, conversion of numeric values from one format to another, and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.
- Transformation of various forms of two-digit year dates to various forms of four-digit year dates using a specified fixed or sliding century window.
- Selection of a character or hexadecimal string from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).

For information concerning the interaction of INREC and OUTREC, see “INREC Statement Notes” on page 119 and “OUTREC Statement Notes” on page 256.

The OUTREC statement differs from the OUTREC parameter of the OUTFIL statement in the following ways:

- The OUTREC statement applies to all input records; the OUTREC parameter applies only to the OUTFIL input records for its OUTFIL group.

- The OUTREC parameter supports the slash (/) separator for creating blank records and new records, whereas the OUTREC statement does not.

See “OUTFIL Control Statements” on page 182 for complete details on the OUTFIL OUTREC parameter.

FIELDS

```
                  ┌─,──────────────────────────────┐
                  ▼                                │
►►─FIELDS=─(──────┬──────┬──────┬──────────────────┬─)──────────────────────────►◄
                  └─c:───┘      ├─s─────────────────┤
                                ├─p,m──┬────┬────────┤
                                │      └─,a─┘        │
                                ├─p─────────────────┤
                                ├─p,m,HEX───────────┤
                                ├─p,HEX─────────────┤
                                ├─p,m,Y2x───────────┤
                                ├─p,m,Y2x(c)────────┤
                                ├─p,m,Y2xP──────────┤
                                ├─p,m,f──┬────┬─────┤
                                │        └─,edit─┘  │
                                ├─p,m,f,to──────────┤
                                ├─p,m,Y2x,edit──────┤
                                ├─p,m,Y2x,to────────┤
                                ├─p,m,lookup────────┤
                                └─seqnum────────────┘
```

Specifies the order and alignment of the separation fields, unedited and edited input fields and sequence numbers in the reformatted output records.

c: specifies the position (column) for a separation field, input field or sequence number, 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 32752.
- 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 16 on page 114 for examples of valid and invalid column alignment.

s specifies that a separation field is to appear in the reformatted output record. 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, nZ, nC'xx...x', and nX'yy...yy'.

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

See Table 17 on page 114 for examples of valid and invalid blank separation.

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

See Table 18 on page 115 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 to appear in the reformatted output records. n can range 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'
```

See Table 19 on page 115 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 to appear in the reformatted output records. n can range 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 20 on page 116 for examples of valid and invalid hexadecimal string separation.

p,m,a
specifies that an unedited input field is to appear in the reformatted output record.

p specifies the first byte of the input field relative to the beginning of the input record.[11] 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 32752. See "OUTREC Statement Notes" on page 256 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 "OUTREC Statement Notes" on page 256 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 of **a** are:

H Halfword aligned. The displacement (p-1) of the field from the

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

beginning of the reformatted input record, in bytes, is a multiple of 2 (that is, position 1, 3, 5, and so forth).

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

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

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

p specifies that the unedited part of the input record (that part beyond the minimum record length) is to appear in the reformatted output record as the last field. Note that if the reformatted output record includes only the RDW and the variable part of the input record, “null” records containing only an RDW may result.

A value must be specified for p that is less than or equal to the minimum record length (RECORD statement L4 value) plus 1 byte.

p,m,HEX
specifies that the hexadecimal representation of an input field is to appear in the reformatted output record.

See p,m,HEX under OUTFIL OUTREC for details.

p,HEX
specifies that the hexadecimal representation of the variable part of the input record (that part beyond the minimum record length) is to appear in the reformatted output record as the last field. Note that if the reformatted output record includes only the RDW and the variable part of the input record, ″null″ records containing only an RDW may result.

See p,HEX under OUTFIL OUTREC for details.

Sample Syntax:

```
Fixed input records
  OUTREC FIELDS=(1:5,10,15:8C'0',25:20,15,80:X)

Variable input records
  OUTREC FIELDS=(1,4,C' RDW=',1,4,HEX,C' FIXED=',
                5,20,HEX,C' VARIABLE=',21,HEX)
```

p,m,Y2x
specifies that the four-digit CH date representation of a two-digit year input date field is to appear in the reformatted output record. Real dates are transformed using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2T transforms C’000000’ to C’00000000’).

See p,m,Y2x under OUTFIL OUTREC for details.

Sample Syntax:

```
OUTREC FIELDS=(21,3,Y2V,X,12,5,Y2W)
```

p,m,Y2x(c)
specifies that the four-digit CH date representation with separators of a two-digit year input date field is to appear in the reformatted output record. Real dates are transformed using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2T(/) transforms C'000000' to C'0000/00/00').

See p,m,Y2x(c) under OUTFIL OUTREC for details.

Sample Syntax:

```
OUTREC FIELDS=(25,6,Y2T(-),X,14,2,Y2U(/))
```

p,m,Y2xP
specifies that the four-digit PD date representation of a two-digit year input date field is to appear in the reformatted output record. Real dates are transformed using the century window established by the Y2PAST option in effect. The century window is not used for special indicators; they are just expanded appropriately (for example, p,6,Y2TP transforms C'000000' to P'00000000').

See p,m,Y2xP under OUTFIL OUTREC for details.

Sample Syntax:

```
OUTREC FIELDS=(11,3,Y2XP,X,21,4,Y2WP)
```

p,m,f,edit
specifies that an edited numeric input field is to appear in the reformatted output record. You can edit BI, FI, PD, PD0, ZD, and CSF/FS fields using either pre-defined edit masks (M0-M25) or specific edit patterns you define. You can control the way the edited fields look with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on.

See p,m,f,edit under OUTFIL OUTREC for details.

Sample Syntax:

```
OUTREC FIELDS=(5:21,8,ZD,M19,X,46,5,ZD,M13,
               31:35,6,FS,SIGNS=(,,+,-),LENGTH=10,
               51:8,4,PD,EDIT=(**II,IIT.TTXS),SIGNS=(,,+,-))
```

p,m,f,to
specifies that a converted numeric input field is to appear in the reformatted output record. You can convert BI, FI, PD, PD0, ZD or CSF/FS fields to BI, FI, PD, ZD or CSF/FS fields.

See p,m,f,to under OUTFIL OUTREC for details.

Sample Syntax:

```
OUTREC FIELDS=(21,5,ZD,TO=PD,X,8,4,ZD,FI,LENGTH=2)
```

p,m,Y2x,edit
specifies that an edited four-digit year CH date representation of a two-digit year input date field is to appear in the reformatted output record.

See p,m,Y2x,edit under OUTFIL OUTREC for details.

p,m,Y2x,to
specifies that a converted four-digit year date representation of a two-digit year input date field is to appear in the reformatted output record.

See p,m,Y2x,to under OUTFIL OUTREC for details.

p,m,lookup
specifies that a character or hexadecimal string from a lookup table is to appear in the reformatted output record. You can use p,m,lookup to select a specified character or hexadecimal string based on matching an input value against character, hexadecimal, or bit constants.

See p,m,lookup under OUTFIL OUTREC for details.

Sample Syntax:

```
OUTREC FIELDS=(11,1,
               CHANGE=(6,
                 C'R',C'READ',
                 C'U',C'UPDATE',
                 X'FF',C'EMPTY',
                 C'A',C'ALTER'),
               NOMATCH=(11,6),
               4X,
               21,1,
               CHANGE=(10,
                 B'.1......',C'VSAM',
                 B'.0......',C'NON-VSAM'))
```

seqnum
specifies that a sequence number is to appear in the reformatted output record. The sequence numbers are assigned in the order in which the records are received for OUTREC processing. You can create BI, PD, ZD or CSF/FS sequence numbers and control their lengths, starting values and increment values.

See seqnum under OUTFIL OUTREC for details.

Sample Syntax:

```
OUTREC FIELDS=(SEQNUM,6,ZD,START=1000,INCR=50,1,60)
```

Default: None; must be specified. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

OUTREC Statement Notes

- If input records are reformatted by INREC or E15, OUTREC must refer to fields in the appropriate reformatted record (see “INREC Statement Notes” on page 119).
- When you specify OUTREC, you must be aware of the change in record size and layout of the resulting reformatted output records.
- If the SORTOUT LRECL is specified or available, DFSORT will use it even if it does not match the reformatted OUTREC record length; this can cause padding or truncation of the reformatted OUTREC records, or termination. If the SORTOUT LRECL is not specified or available, DFSORT can automatically use the reformatted OUTREC record length as the SORTOUT LRECL, when

appropriate. See the discussion of the SOLRF and NOSOLRF options in "OPTION Control Statement" on page 134 for details.

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 14 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 (RECORD statement L4 value) 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 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.
- When OUTREC is specified, your E35 user exit routine must refer to fields in the reformatted output record.
- DFSORT issues a message and terminates processing 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.
- If SZERO is in effect, -0 is treated as negative and +0 is treated as positive for edited or converted input fields. If NOSZERO is in effect, -0 and +0 are treated as positive for edited or converted input fields.

Reformatting the Output Record—Examples

See "Reformatting Records Before Processing — Examples" on page 120. Example 1, Example 3, and Example 4 show applications in which both INREC and OUTREC statements are used in the same application.

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.

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, DFSORT issues a message and terminates processing.

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.

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. SOLRF (the IBM-supplied default) is in effect, so unless the SORTOUT LRECL is specified or available, it will automatically be set to the reformatted record length of 103. The reformatted records look as follows:

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'

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. 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)

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. The reformatted records look as follows:

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

Example 7

```
OPTION COPY,Y2PAST=1985
  OUTREC FIELDS=(SEQNUM,8,ZD,START=1000,INCR=100,
                 11:8,4,PD,M12,
                 31:15,4,Y2V(/),
                 51:2,1,CHANGE=(3,
                          X'01',C'L92',X'02',C'M72',X'03',C'J42'),
                        NOMATCH=(C'???'))
```

This example illustrates how a sequence number can be generated, how numeric and date values can be edited, and how a lookup table can be used.

The reformatted output records look as follows:

Position	Contents
1-8	A zoned decimal sequence number that starts at 1000 and increments by 100.
11–20	A CH field containing the PD field from input positions 8 through 11 edited according to the M12 edit mask.

31-40	A C'yyyy/mm/dd' date field containing the P'yymmdd' date field from input positions 15-18 transformed according to the specified century window of 1985-2084.
51–53	A CH field containing C'L92', C'M72', C'J42' or C'???' as determined by using a lookup table for the input field in position 2.

RECORD Control Statement

```
►►─RECORD──┬─TYPE=─┬─F─┬──────────────────────────────────────────────────────►◄
           │       │ V │                                                      │
           │       └ D ┘                                                      │
           │                                                                  │
           ├─┬──────────┬──LENGTH=(─┬─L1──────────────────────────────────┬─)──┤
           │ └TYPE=F,───┘           │                                     │   │
           │                        ├─┬────┬─,─L2──────────────────────────┤   │
           │                        │ └L1──┘                              │   │
           │                        └─┬────┬─,─┬────┬─,─L3──────────────────┘   │
           │                          └L1──┘   └L2──┘                          │
           │                                                                  │
           └─┬──────────────┬──LENGTH=(─┬─L1──────────────────────────────┬─)──┘
             └TYPE=─┬─V─┬─,─┘           │                                 │
                    └ D ┘                ├─┬────┬─,─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──┘
```

The RECORD control statement can be used to specify the type and lengths of the records being processed, and the minimum and average record lengths for a variable-length sort.

The RECORD control statement is required when:

- A user exit changes record lengths.
- A user exit supplies all of the input records.
- A Conventional merge or tape work data set sort uses VSAM input.

TYPE

```
►►──TYPE=x──────────────────────────────────────────────────────────────────►◄
```

Can be used to specify the record type when input is VSAM, or an E15 or E32 exit supplies all of the input records. The record type can be:

- **Fixed-length (F).** The records are processed without an RDW, so the data starts in position 1. Control statement positions should be specified accordingly.

  An RRDS can always be processed as fixed-length. A KSDS, ESDS or VRRDS used for input should only be processed as fixed-length if all of its records have a length equal to the maximum record size defined for the

cluster. Otherwise, input records which are shorter than the maximum record size are padded with bytes that may or may not be zeros (that is, ″garbage″ bytes).

- **Variable-length (V).** The records are processed with an RDW in positions 1-4, so the data starts in position 5. Control statement positions should be specified accordingly.

  An RRDS, KSDS, ESDS or VRRDS can always be processed as variable-length. For VSAM input, DFSORT reads each record and prepends an RDW to it. For VSAM output, DFSORT removes the RDW before writing each record.

TYPE is only required for a Conventional merge or tape work data set sort that uses VSAM input or an E15 or E32 exit that supplies all of the input records.

If input is non-VSAM, DFSORT determines the record type from the RECFM of the input data set and ignores TYPE.

If input is VSAM, or an E15 or E32 exit supplies all of the input records, DFSORT determines or assigns the record type as follows, using the information in the order listed:

1. F or V from RECORD TYPE if specified.
2. F or V from SORTOUT RECFM if available.
3. V if OUTFIL CONVERT or VLFILL specified.
4. F or V from OUTFIL RECFM if available.
5. V if SORTIN is VSAM and SORTOUT is VSAM; otherwise F.

   Notes:

   a. If the selected record type is not what you want DFSORT to use, specify RECORD TYPE=F or RECORD TYPE=V as appropriate.
   b. For a Conventional merge or tape work data set sort, you must specify RECORD TYPE=F or RECORD TYPE=V as appropriate.

x can be one of the following:

F fixed-length record processing.

Note: FB can be used instead of F.

V variable-length record processing.

Note: VB can be used instead of V.

D ISCII/ASCII variable-length record processing.

Note: DB can be used instead of D.

Default: F or V as described above. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

LENGTH

```
►►─LENGTH=─(──────────────────────────────────────────────────────►
►─┬─L1──────────────────────────────────────────────────────────┬►
  │ ┌─L1─┐ ,─L2──────────────────────────────────────────────────│
  │ ┌─L1─┐ , ┌─L2─┐ ,─L3──────────────────────────────────────────│
  │ ┌─L1─┐ , ┌─L2─┐ , ┌─L3─┐ ,─L4──────────────────────────────────│
  │ ┌─L1─┐ , ┌─L2─┐ , ┌─L3─┐ , ┌─L4─┐ ,─L5──────────────────────────│
  │ ┌─L1─┐ , ┌─L2─┐ , ┌─L3─┐ , ┌─L4─┐ , ┌─L5─┐ ,─L6──────────────│
  │ ┌─L1─┐ , ┌─L2─┐ , ┌─L3─┐ , ┌─L4─┐ , ┌─L5─┐ , ┌─L6─┐ ,─L7──────│
►─)──────────────────────────────────────────────────────────────►◄
```

Can be used to specify various record lengths. L1 through L3 apply to fixed-length and variable-length record processing. L4 and L5 apply to variable-length record processing. L6 and L7 are accepted, but not used.

LENGTH is required only if:

- A user exit changes record lengths.
- A user exit supplies all of the input records.

L1 Input record length. For variable-length records, maximum input record length.

Notes:
1. L1 is ignored if the input record length is available from SORTIN.
2. L1 is required if there is no SORTIN or SORTINnn data set, unless L2 is specified.

Default: The SORTIN or SORTINnn record length. For VSAM data sets, the maximum record size (RECSZ value).

L2 Record length after E15. For variable-length records, maximum record length after E15.

Notes:
1. L2 is ignored if E15 is not used.
2. An accurate value for L2 must be specified if E15 changes the record length.
3. L2 must be at least 18 bytes if tape work data sets are used.
4. L2 is ignored if there is no SORTIN or SORTINnn data set, unless L1 is not specified.

Default: L1.

L3 Output record length. For variable—length records, maximum output record length.

Note: L3 is ignored if the record length (LRECL or VSAM RECSZ) is available from SORTOUT, or if NOSOLRF is in effect and E35, INREC, OUTREC, and OUTFIL are *not* used.

Default: One of the following, in the order listed:

1. SORTOUT record length if available
2. OUTREC record length if SOLRF is in effect
3. INREC record length if SOLRF is in effect
4. L2 if specified providing an E15 is used
5. SORTIN or SORTINnn record length if available
6. L1

L4 Minimum record length.

Notes:

1. L4 is not used if the Blockset technique is selected
2. L4 is only used for variable-length record sort applications.
3. Specifying L4 may improve performance, but if L4 is too large, DFSORT could fail with message ICE015A.

Default: The minimum length needed to contain all control fields. This number must be at least 18 bytes if the maximum input record length is greater than 18 bytes; otherwise, DFSORT sets L4 to 18 bytes.

L5 Average record length.

Notes:

1. L5 is not used if the Blockset technique is selected
2. L5 is overridden by the AVGRLEN parameter if both are specified
3. L5 is only used for variable-length sorts.

Default: None; optional.

L6, L7
Record lengths that are accepted but are reserved for future use.

Notes:

1. You can drop values from the right. For example, LENGTH=(80,70,70,70).
2. You can omit values from the middle or left, provided you indicate their omission by a comma or semicolon. For example, LENGTH=(,,,30,80).
3. Parentheses are optional when L1 alone is specified. If any of L2 through L7 is specified, with or without L1, parentheses are required.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Describing the Record Format and Length—Examples

Example 1

```
MODS E15=(INEX,1000,EXIT),E35=(OUTEX,2000,EXIT)
 RECORD LENGTH=(,175,180)
```

This example illustrates how the RECORD statement can be used to indicate that E15 and E35 exits change the record length. The record type (F) and input record length (200) are obtained automatically from the RECFM and LRECL of the input data set, respectively.

LENGTH
L2 specifies that the E15 exit passes back 175 byte records. L3 specifies that the E35 exit passes back 180 byte records.

Example 2

```
MODS E15=(E15ONLY,1000,EXIT)
 RECORD TYPE=V,LENGTH=60
```

This example illustrates how the RECORD statement can be used to set the record type and maximum input record length when an E15 exit supplies all of the input as variable-length records.

TYPE
V specifies that the E15 exit inserts variable-length records, that is, the inserted records contain an RDW in positions 1-4 and the data starts in position 5.

LENGTH
L1 specifies that the E15 exit inserts records with a maximum length of 60 bytes.

SORT Control Statement

```
                                  ,
                                  ▼
►►──SORT──FIELDS=──┬──(──────p,m,f,s──────)──────────────────────────────►
                   │                  ,
                   │                  ▼
                   ├──(──────p,m,s──────)──,──FORMAT=f──────────────────┤
                   │  ┌─COPY────┐
                   └──┤         ├──────────────────────────────────────┘
                      └─(COPY)──┘

►──┬──────────────────────────────────────────────────────────────────────►◄
   │                  ,
   │                  ▼
   └──,──┬──CKPT──────────────────────────────────────────────────────┬──┘
         │──DYNALLOC──┬──────────────────────────────────────────────┤
         │            └──=──┬──d──────────────────────────────────┘
         │                  ├──(d)──────────────────────────────┤
         │                  ├──(,n)─────────────────────────────┤
         │                  ├──(d,n)────────────────────────────┤
         │                  ├──OFF──────────────────────────────┤
         │                  └──(OFF)────────────────────────────┘
         │──┬──EQUALS──────────────────────────────────────────────┤
         │  └──NOEQUALS────────────────────────────────────────────┘
         │──FILSZ=──┬──x──┬──────────────────────────────────────┤
         │          ├──Ex─┤                                        │
         │          └──Ux─┘                                        │
         │──SIZE=───┬──y──┬──────────────────────────────────────┤
         │          ├──Ey─┤                                        │
         │          └──Uy─┘                                        │
         │──SKIPREC=z───────────────────────────────────────────────┤
         │──STOPAFT=n───────────────────────────────────────────────┤
         └──Y2PAST=──┬──s──┬──────────────────────────────────────┘
                     └──f──┘
```

The SORT control statement must be used when a sorting application is 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. User labels will not be copied to the output data sets.

The way in which DFSORT processes short SORT control fields depends on the setting for VLSHRT/NOVLSHRT. A short field is one where the variable-length record is too short to contain the entire field, that is, the field extends beyond the record. For details about sorting short records, see the discussion of the VLSHRT and NOVLSHRT options in “OPTION Control Statement” on page 134.

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 B. Specification/Override of DFSORT Options” on page 549.

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

DFSORT accepts but does not process the following SORT operands: WORK=value and ORDER=value.

DFSORT’s collating behavior can be modified according to your cultural environment. The cultural environment is established by selecting the active locale. The active locale’s collating rules affect SORT processing as follows:

- DFSORT produces sorted records for output according to the collating rules defined in the active locale. This provides sorting for single- or multi-byte character data, based on defined collating rules that retain the cultural and local characteristics of a language.

If locale processing is to be used, the active locale will only be used to process character (CH) control fields.

For more information on locale processing, see “Cultural Environment Considerations” on page 6 or LOCALE in “OPTION Control Statement” on page 134.

FIELDS

Requires four facts about each control field in the input records: the position of

```
                  ┌─,──────────┐
►►─FIELDS=(──────┴─p,m,f,s────┴──)──────────────────────────────────────►◄
```

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. However, INREC and OUTREC can be used to rearrange the records such that fields beyond the first 4092 bytes can be sorted as illustrated by “Example 4” on page 122.

Control fields must not extend beyond the shortest record to be sorted unless VLSHRT is in effect. The collected control fields (comprising the control word) must not exceed 4092 bytes (or 4088 bytes when EQUALS is in effect). 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 in the discussion of the

FORMAT parameter later in this section) 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. [12]

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 shown in Figure 16 on page 267:

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

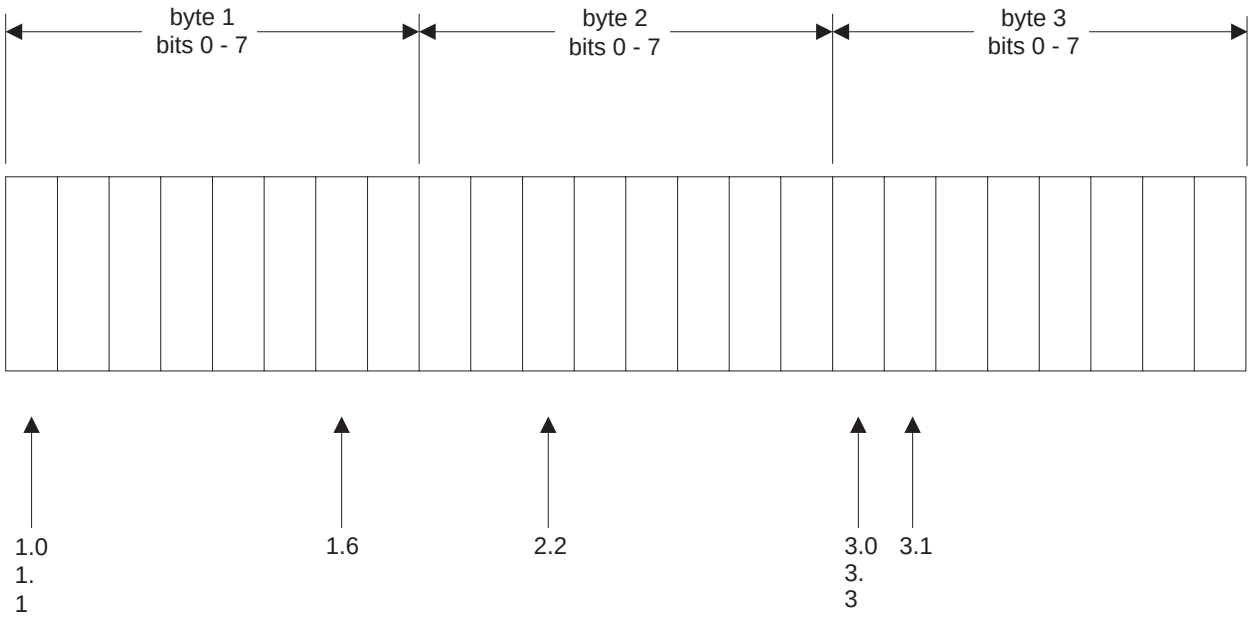


Figure 16. Examples of Notation for Binary Fields

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 34.

Table 34. Control Field Formats and Lengths

Format	Length	Description
CH	1 to 4092 bytes	Character [13]
AQ	1 to 256 bytes	Character with alternate collating sequence
ZD	1 to 32 bytes	Signed zoned decimal
PD	1 to 32 bytes	Signed packed decimal

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

Table 34. Control Field Formats and Lengths (continued)

Format	Length	Description
PD0	2 to 8 bytes	Packed decimal with sign and first digit ignored
FI	1 to 256 bytes	Signed fixed-point
BI	1 bit to 4092 bytes	Unsigned binary
FL	1 to 256 bytes	Signed floating-point
AC	1 to 256 bytes	ISCII/ASCII character
CSF or FS	1 to 16 bytes	Signed numeric with optional leading floating sign
CSL or LS	2 to 256 bytes	Signed numeric with leading separate sign
CST or TS	2 to 256 bytes	Signed numeric with trailing separate sign
CLO or OL	1 to 256 bytes	Signed numeric with leading overpunch sign
CTO or OT	1 to 256 bytes	Signed numeric with trailing overpunch sign
ASL	2 to 256 bytes	Signed ISCII/ASCII numeric with leading separate sign
AST	2 to 256 bytes	Signed ISCII/ASCII numeric with trailing separate sign
D1	1 to 4092 bytes	User-defined data type (requires an EFS program)
Y2T	3 to 6 bytes	Character or zoned yyx...x full date format with special indicators
Y2U	2 or 3 bytes	Packed decimal yyx and yyxxx full date format with special indicators
Y2V	3 or 4 bytes	Packed decimal yyxx and yyxxxx full date format with special indicators
Y2W	3 to 6 bytes	Character or zoned x...xyy full date format with special indicators
Y2X	2 or 3 bytes	Packed decimal xyy and xxxyy full date format with special indicators
Y2Y	3 or 4 bytes	Packed decimal xxyy and xxxxyy full date format with special indicators
Y2C or Y2Z	2 bytes	Two-digit character or zoned-decimal year data
Y2P	2 bytes	Two-digit packed-decimal year data
Y2D	1 byte	Two-digit decimal year data
Y2S	2 bytes	Two-digit character or zoned-decimal year data with special indicators

Table 34. Control Field Formats and Lengths (continued)

Format	Length	Description
Y2B	1 byte	Two-digit binary year data

Note: See "Appendix C. Data Format Descriptions" on page 581 for detailed format descriptions.

CSF/FS, Y2 and PD0 format fields can only be used if Blockset is selected.

For Y2 format fields, real dates are collated using the century window established by the Y2PAST option in effect, but the century window is not used for special indicators. Thus the Y2 formats will collate real dates and special indicators as follows:

- Y2T and Y2W:

  Ascending:
  BI zeros, blanks, CH/ZD zeros, lower century dates (for example, 19yy), upper century dates (for example, 20yy), CH/ZD nines, BI ones.

  Descending:
  BI ones, CH/ZD nines, upper century dates (for example, 20yy), lower century dates (for example, 19yy), CH/ZD zeros, blanks, BI zeros.
- Y2U, Y2V, Y2X and Y2Y:

  Ascending:
  PD zeros, lower century dates (for example, 19yy), upper century dates (for example, 20yy), PD nines.

  Descending:
  PD nines, upper century dates (for example, 20yy), lower century dates (for example, 19yy), PD zeros.
- Y2C, Y2Z, Y2P, Y2D and Y2B:

  Ascending:
  Lower century years (for example, 19yy), upper century years (for example, 20yy).

  Descending:
  Upper century years (for example, 20yy), lower century years (for example, 19yy).
- Y2S:

  Ascending:
  BI zeros, blanks, lower century years (for example, 19yy), upper century years (for example, 20yy), BI ones.

  Descending:
  BI ones, upper century years (for example, 20yy), lower century years (for example, 19yy), blanks, BI zeros.

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 an E15 or E61 user exit to do this during processing. If you use E61, specify the E option for the value of s in the FIELDS operand for each control field you are going to modify with this user exit.

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 an E61 user exit to modify control fields before the program sorts them. After an E61 user exit modifies the control fields, DFSORT collates the records in ascending order using the formats specified. [14]

For information on how to add a user exit, see “Chapter 4. Using Your Own User Exit Routines” on page 279.

Default: None; must be specified. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

FORMAT

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

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

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

Default: None; must be specified if not included in FIELDS parameter. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Note: If format values are specified in both FORMAT and FIELDS, DFSORT issues an information message, uses the format values from FIELDS (f must be specified for each control field), and does not use the format values from FORMAT.

FIELDS=COPY or FIELDS=(COPY)

14. 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.

```
►►──FIELDS=──┬─COPY─────┬──────────────────────────────────────────►◄
             └─(COPY)───┘
```

See the discussion of the COPY option discussed in “OPTION Control Statement” on page 134.

CKPT

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

See the discussion of this option discussed in “OPTION Control Statement” on page 134.

DYNALLOC

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

See the discussion of this option in “OPTION Control Statement” on page 134.

EQUALS or NOEQUALS

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

See the discussion of these options in “OPTION Control Statement” on page 134.

FILSZ or SIZE

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

See the discussion of these options in “OPTION Control Statement” on page 134.

SKIPREC

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

See the discussion of this option in “OPTION Control Statement” on page 134.

STOPAFT

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

See the discussion of this option in “OPTION Control Statement” on page 134.

Y2PAST

```
►►──Y2PAST=──┬─s─┬──────────────────────────────────────────────────────►◄
             └─f─┘
```

See the discussion of this option in “OPTION Control Statement” on page 134.

Note: CENTURY=value and CENTWIN=value can be used instead of Y2PAST=value.

SORT/MERGE Statement Notes

- If records are reformatted by INREC (SORT and MERGE) or E15 (SORT), FIELDS must refer to fields in the appropriate reformatted records.
- If SZERO is in effect, −0 collates before +0 in ascending order and after +0 in descending order when numeric fields are sorted or merged. If NOSZERO is in effect, −0 collates equally with +0 when numeric fields are sorted or merged. However, SZERO is always used for a conventional merge or tape work data set sort application.

Specifying a SORT or COPY—Examples

Example 1

```
SORT   FIELDS=(2,5,FS,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 floating sign data. It is to be sorted in ascending order.

FILSZ
The data set to be sorted contains exactly 29483 records.

Example 2

```
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),DYNALLOC=(3390,4)
```

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 could specify E instead of A and include your own E61 user exit routine to normalize the field before the program examined it.

DYNALLOC
Four work data sets are allocated on 3390. The space on each data set is calculated using the FILSZ value.

Example 3

```
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.

Example 4

```
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.

Example 5

```
SORT   FIELDS=COPY
```

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

Example 6

```
OPTION Y2PAST=1950
SORT FIELDS=(21,6,Y2T,A,13,3,Y2X,D)
```

Y2PAST
Sets a century window of 1950–2049.

FIELDS
Sorts on a C'yymmdd' (or Z'yymmdd') date in positions 21-26 in ascending order, and on a P'dddyy' date in positions 13-15 in descending order. ″Real″ dates are sorted using the century window of 1950-2049. Special indicators are sorted correctly relative to the ″real″ dates.

SUM Control Statement

```
                    ┌─,──────────┐
                    ▼            │
►►─SUM─FIELDS=─┬─┬─(──────p,m,f──┴──────)──────────────────────────────►◄
               │ │                                                    │
               │ │                 ┌─,──────────┐                     │
               │ │                 ▼            │                     │
               │ ├─(──────p,m──────┴──────),─FORMAT=f──────────────────┤
               │ │                                                    │
               │ └─┬─NONE──────────────────────────────────────────────┤
               │   └─(NONE)──┘                                        │
               └──────────────────────────────────────────────────────┘
```

The SUM control statement specifies that, whenever two records are found with equal sort or merge 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. [15] 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. See below for permissible length values.

f specifies the format of the data in the summary field:

Table 35. Summary Field Formats and Lengths

Format Code	Length	Description
BI	2, 4, or 8 bytes	Unsigned binary

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

Table 35. Summary Field Formats and Lengths (continued)

Format Code	Length	Description
FI	2, 4, or 8 bytes	Signed fixed-point
FL	4, 8, or 16 bytes	Signed floating-point
PD	1 to 16 bytes	Signed packed decimal
ZD	1 to 18 bytes	Signed zoned decimal

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

NONE or (NONE)
eliminates records with duplicate keys. Only one record with each key is kept and no summing is performed.

Note: The FIRST operand of ICETOOL’s SELECT operator can be used to perform the same function as SUM FIELDS=NONE with OPTION EQUALS. Additionally, SELECT’s ALLDUPS, NODUPS, HIGHER(x), LOWER(y), EQUAL(v) and LAST operands can be used to select records based on other criteria related to duplicate and non-duplicate keys. SELECT’s DISCARD(savedd) operand can be used to save the records discarded by FIRST, ALLDUPS, NODUPS, HIGHER(x), LOWER(y), EQUAL(v) or LAST. See “SELECT Operator” on page 409 for complete details on the SELECT operator.

Default: None; must be specified.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

FORMAT

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

FORMAT=f can be used only when all the summary fields in the entire FIELDS 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 FIELDS parameter. See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

Applicable Functions: See “Appendix B. Specification/Override of DFSORT Options” on page 549.

Note: If format values are specified in both FORMAT and FIELDS, DFSORT issues an information message, uses the format values from FIELDS (f must be specified for each summary field), and does not use the format values from FORMAT.

SUM Statement Notes

- An invalid PD or ZD sign or digit results in a data exception (0C7 ABEND); 0-9 are invalid for the sign and A-F are invalid for the digit. For example, a ZD value such as 3.5 (X'F34BF5') results in an 0C7 because ″.″ (X'4B') is treated as an invalid digit. ICETOOL’s DISPLAY or VERIFY operator can be used to identify

decimal values with invalid digits. ICETOOL's VERIFY operator can be used to identify decimal values with invalid signs.

- Whether or not positive summed ZD results have printable numbers depends on whether NZDPRINT or ZDPRINT is in effect (as set by the ZDPRINT option of ICEMAC and the NZDPRINT and ZDPRINT parameters of the OPTION statement):
  - If NZDPRINT is in effect, positive summed ZD results do not consist of printable numbers, regardless of whether the original values consisted of printable numbers or not. For example, if X'F2F3F1' (prints as '231') and X'F3F0F6' (prints as '306') are summed, the result with NZDPRINT in effect is X'F5F3C7' (prints as '53G').
  - If ZDPRINT is in effect, positive summed ZD results consist of printable numbers, regardless of whether the original values consisted of printable numbers or not. For example, if X'F2F3C1' (prints as '23A') and X'F3F0F6' (prints as '306') are summed, the result with ZDPRINT in effect is X'F5F3F7' (prints as '537').

  Thus, ZDPRINT must be in effect to ensure that positive summed ZD results are printable.

  Unsummed positive ZD values retain their original signs, regardless of whether NZDPRINT or ZDPRINT is in effect. For example, if X'F2F8C5' is not summed, it remains X'F2F8C5' (prints as '28E'). OUTFIL's OUTREC parameter can be used to ensure that all summed or unsummed ZD values are printable, as illustrated by Example 4 below.
- If input records are reformatted by INREC or E15, SUM must refer to fields in the appropriate reformatted record (see the preceding description of *p*).
- Summary fields must not be control fields. They must not overlap control fields, or each other, and must not overlap the RDW.
- FL values to be summed can be normalized or unnormalized. However, the resulting FL values are always normalized. Normalization processing by the hardware can produce different sums for FL values summed in different orders.
- Exponent overflow for summed FL values results in an exponent overflow exception (0CC ABEND)
- Exponent underflow for summed FL values results in a true zero result.
- 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, overflow does not occur, and the BLOCKSET technique is used. In this case, the first record (based on the sequence described under the discussion of the EQUALS or NOEQUALS parameter of the "OPTION Control Statement" on page 134) is chosen to contain the sum.
- Fields other than summary fields remain unchanged and are taken from the record that receives the sum.
- You can control the action that DFSORT takes when overflow occurs for BI, FI, PD or ZD values with the OVFLO parameter as described in "OPTION Control Statement" on page 134.
- DFSORT issues a message and terminates processing if a SUM statement is specified for a tape work data set sort or Conventional merge.

Adding Summary Fields—Examples

Example 1

```
SUM FIELDS=(21,8,PD,11,4,FI)
```

This statement designates an 8-byte packed decimal field at byte 21, and a 4-byte fixed-integer field at byte 11, as summary fields.

Example 2

```
SUM FIELDS=NONE
```

This statement illustrates the elimination of duplicate records.

Example 3

```
SUM FIELDS=(41,8,49,4),FORMAT=ZD
OPTION ZDPRINT
```

These statements illustrate the use of the FORMAT operand and the ZDPRINT option. The SUM statement designates two zoned decimal fields, one 8 bytes long starting at byte 41, and the other 4 bytes long starting at byte 49. As a result of the ZDPRINT option, the positive summed ZD values will be printable. Note, however, that the ZDPRINT option does not affect ZD values which are not summed due to overflow or unique keys. The next example shows how to use OUTFIL to make all summary fields printable.

Example 4

```
SUM FIELDS=(41,8,49,4),FORMAT=ZD
OUTFIL OUTREC=(1,40,41,8,ZD,M11,49,4,ZD,M11,53,28)
```

These statements illustrate the use of the OUTFIL statement to ensure that all positive ZD summary fields in the output data set are printable. Whereas the ZDPRINT option affects only positive summed ZD fields, OUTFIL can be used to edit positive or negative BI, FI, PD, or ZD values, whether they are summed or not. OUTFIL can also be used to produce multiple output data sets, reports, and so on. See “OUTFIL Control Statements” on page 182 for complete details about OUTFIL processing.

Note: For purposes of illustration, this example assumes that the input records are 80 bytes long.

Chapter 4. Using Your Own User Exit Routines

User Exit Routine Overview 280
DFSORT Program Phases 281
Functions of Routines at User Exits 283
DFSORT Input/User Exit/Output Logic Examples 283
Opening and Initializing Data Sets 284
Modifying Control Fields 284
Inserting, Deleting, and Altering Records 285
Summing Records 285
Handling Special I/O 285
Routines for Read Errors 285
Routines for Write Errors 285
VSAM User Exit Functions 286
Determining Action when Intermediate Storage Is Insufficient 286
Closing Data Sets 286
Terminating DFSORT 286
Addressing and Residence Modes for User Exits 286
How User Exit Routines Affect DFSORT Performance 287
Summary of Rules for User Exit Routines 287
Loading User Exit Routines 288
User Exit Linkage Conventions 288
Linkage Examples 289
Dynamically Link-Editing User Exit Routines 289
Assembler User Exit Routines (Input Phase User Exits) 290
E11 User Exit: Opening Data Sets/Initializing Routines 290
E15 User Exit: Passing or Changing Records for Sort and Copy Applications 290
Information DFSORT Passes to Your Routine at E15 User Exit 291
E15 Return Codes 292
Storage Usage for E15 User Exit 293
E16 User Exit: Handling Intermediate Storage Miscalculation 293
E16 Return Codes 293
E17 User Exit: Closing Data Sets 294
E18 User Exit: Handling Input Data Sets 294
Using E18 User Exit with QSAM/BSAM 294
Using E18 User Exit with VSAM 295
E19 User Exit: Handling Output to Work Data Sets 297
Using E19 User Exit with QSAM/BSAM 297
E61 User Exit: Modifying Control Fields 298
Some Uses of E61 User Exit 298
Information DFSORT Passes to Your Routine at E61 User Exit 298
Assembler User Exit Routines (Output Phase User Exits) 299
E31 User Exit: Opening Data Sets/Initializing Routines 300
E32 User Exit: Handling Input to a Merge Only 300
Information DFSORT Passes to Your Routine at E32 User Exit 300
E32 Return Codes 301
E35 User Exit: Changing Records 301
Information DFSORT Passes to Your Routine at E35 User Exit 302
E35 Return Codes 303
Storage Usage for E35 User Exit 304
E37 User Exit: Closing Data Sets 304
E38 User Exit: Handling Input Data Sets 305
Using E38 User Exit with VSAM 305
E39 User Exit: Handling Output Data Sets 305
Using E39 User Exit with QSAM/BSAM 305

© Copyright IBM Corp. 1973, 2000

Using E39 User Exit with VSAM 305
Sample Routines Written in Assembler. 306
E15 User Exit: Altering Record Length 306
E16 User Exit: Sorting Current Records When NMAX Is Exceeded 307
E35 User Exit: Altering Record Length 308
E61 User Exit: Altering Control Fields 308
COBOL User Exit Routines 309
COBOL User Exit Requirements 309
COBOL Requirements for Copy Processing 310
COBOL Storage Requirements 311
COBOL User Exit Routines (Input Phase User Exit) 312
COBOL E15 User Exit: Passing or Changing Records for Sort 312
E15 Interface with COBOL 312
E15 LINKAGE SECTION Fields for Fixed-Length and Variable-Length Records 315
E15 Return Codes 316
E15 Procedure Division Requirements 318
COBOL User Exit Routines (Output Phase User Exit) 318
COBOL E35 User Exit: Changing Records 318
E35 Interface with COBOL 319
E35 LINKAGE SECTION Fields for Fixed-Length and Variable-Length Records 321
E35 Return Codes 322
E35 Procedure Division Requirements 324
Sample Routines Written in COBOL. 324
COBOL E15 User Exit: Altering Records 324
COBOL E35 User Exit: Inserting Records 325
E15/E35 Return Codes and EXITCK 327

User Exit Routine Overview

DFSORT can pass program 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 need to perform these tasks, 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, OUTFIL, OUTREC, and SUM program control statements in “Chapter 3. Using DFSORT Program Control Statements” on page 77. 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 assembleror COBOL, you can write your exit routines in any language that can:

- Pass and accept the address into general register 1 of a:
  - Record
  - Full word of zeros
  - Parameter list.
- Pass a return code in register 15.

You can easily activateuser exit routines at run-time with the MODS program control statement (see “MODS Control Statement” on page 127). 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 5. Invoking DFSORT from a Program” on page 331 for details.

Parameters that affect the way user exit routines are handled include:

- The MODS statement, explained in “MODS Control Statement” on page 127
- The COBEXIT option of the ICEMAC installation macro, explained in “Installation Defaults” on page 16
- The E15=COB and E35=COB PARM options of the EXEC statement, explained in “Specifying EXEC/DFSPARM PARM Options” on page 29
- The COBEXIT option of the OPTION statement, explained in “OPTION Control Statement” on page 134
- The ICEMAC installation option EXITCK, explained in “Installation Defaults” on page 16
- The EXITCK option of the OPTION statement, explained in “OPTION Control Statement” on page 134.

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 “Use Main Storage Efficiently” on page 498 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 processing. The input phase is used only for a sort or copy. When the output phase is completed, DFSORT returns control to the operating system or invoking program. Figure 17 on page 282 is a representation of DFSORT input/output logic.

DFSORT Program Phases

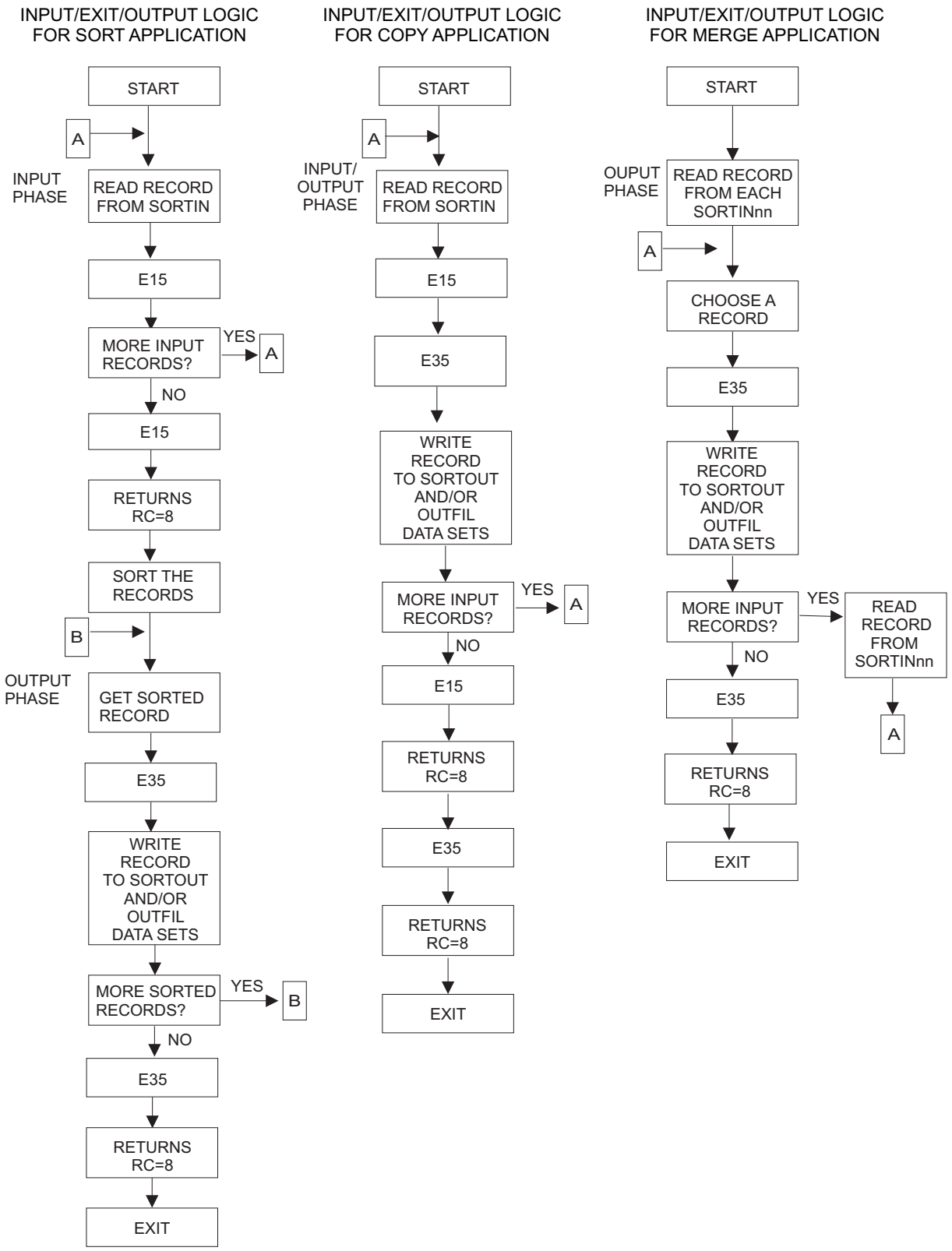


Figure 17. Examples of DFSORT Input/User Exit/Output Logic

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
- Sum records
- Handle special I/O conditions
- Determine action when intermediate storage is insufficient
- Close data sets
- Terminate DFSORT.

Figure 17 on page 282, Table 36, and Table 37 on page 284 summarize the functions of user exit routines and the exits and phases with which they can be associated.

DFSORT Input/User Exit/Output Logic Examples

Figure 17 on page 282 gives examples of the logic flow for sort, copy, and merge applications as it relates to SORTINnn, E15 or E35 user exits, and SORTOUT. The intent is to show how your E15 and E35 user exits 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 user exit supplied the records rather than SORTINnn data sets.)

Figure 17 on page 282 illustrates the following logic:
- E15 and E35 user exits continue to be entered until they pass back a return code of 8. If your user exit passes a return code of 8 to DFSORT when input records still remain to be processed, the records are processed by DFSORT *without* being passed to your exit.
- During a sort, each record is read from SORTIN and passed to E15 user exit. 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 the output data sets.
- During a copy, each record is read from SORTIN, passed to E15 and E35 user exits, and written to the output data sets.
- During a merge, one record is initially read from each SORTINnn data set. The record to be output is chosen, passed to E35, and written to the output data sets. The chosen record is then replaced by reading a record from the same SORTINnn data set and the process continues.

  Note: For a merge application, records deleted during an E35 user exit routine are not sequence-checked. If you use an E35 user exit routine without an output data set, sequence checking is not performed at the time the records are passed to the E35 user exit; therefore, you must ensure that input records are in correct sequence.

Table 36. Functions of Routines at Program User Exits (Sort)

Functions	Sort Input Phase	Sort Output Phase
Open/Initialize	E11, E15 user exits	E31 user exit
Modify control fields	E61 user exit	N/A
Insert, Delete/Alter	E15 user exit	E35 user exit
Sum records		E35 user exit[1]

Table 36. Functions of Routines at Program User Exits (Sort) (continued)

Functions	Sort Input Phase	Sort Output Phase
Handle special I/O conditions: QSAM/BSAM and VSAM SORTIN QSAM/BSAM SORTOUT VSAM SORTOUT	 E18 user exit E19 user exit[2] N/A	 E38 user exit[2] E39 user exit[3] E39 user exit[3]
Determine action when intermediate storage is insufficient	E16 user exit[4]	N/A
Close/housekeeping	E15, E17 user exits	E35, E37 user exits
Terminate DFSORT	E15 user exit	E35 user exit

Notes:
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.
3. E39 can be used for SORTOUT, but not for OUTFIL data sets.
4. Applies only to a tape work data set sort or a Peerage/Vale sort without work data sets.

Table 37. Functions of Routines at Program User Exits (Copy and Merge)

Functions	Copy	Merge
Open/Initialize	E15, E31 user exits	E31 user exit
Modify control fields	N/A	E61 user exit
Insert	E15, E35 user exits	E32, E35 user exits
Delete/alter	E15, E35 user exits	E35 user exit
Sum records	E35 user exit	E35 user exit[1]
Handle special I/O conditions: QSAM/BSAM and VSAM SORTIN(nn) QSAM/BSAM and VSAM SORTOUT	 E38 user exit E39 user exit	 E38 user exit E39 user exit
Close/housekeeping	E35, E37 user exits	E35, E37 user exits
Terminate DFSORT	E15, E35 user exits	E32, E35 user exits

Note:
1. The SUM control statement can be used instead of your own routine to sum records.

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 user exits.

To check labels on input files, use the E18 and E38 user 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. You must associate this routine with the E61 user exit.

When an E61 user exit is used, the subsequent comparisons always arrange the modified control fields in ascending order.

Note: Although you are altering control fields before a compare, your original records are not altered.

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 user exits.

Note: DFSORT also provides INCLUDE and OMIT statements, and OUTFIL INCLUDE and OMIT parameters 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 77.

Summing Records

You can sum records for output by using the E35 user exit. However, you can also use DFSORT’s SUM program control statement to accomplish this without a user exit. See “SUM Control Statement” on page 274.

Handling Special I/O

DFSORT contains four exits to handle special I/O conditions: E18 and E38 user exits for SORTIN and SORTINnn, and E19 and E39 user exits for SORTOUT (but not for OUTFIL data sets). They are particularly useful for a tape work data set sort. With all DASD work data set sorts, E19 and E38 user exits are ignored.

You can use these exits to incorporateyour 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 uncorrectable I/O error, it passes the same parameters as those passed by QSAM/BSAM or VSAM. If no user routines are supplied and an uncorrectable read or write error is encountered, DFSORT issues an error message and then terminates.

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

- General registers 0 and 1 are unchanged; they contain the information passed by QSAM/BSAM, as documented in the data management publications.
- General register 14 contains the return address of DFSORT.
- General register 15 contains the address of your error routine.

VSAM will go directly to any routine specified in the EXLST macro you passed to DFSORT via the E18, E38, or E39 user exit, as appropriate. Your routine must return to VSAM via register 14. For details, see *Macro Instructions for Data Sets* or *Using Data Sets*

Routines for Read Errors

You must associate these routines with the E18 and E38 user 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.

Routines for Write Errors

You must associate these routines with the E19 and E39 user exits. These routines can perform any necessary abnormal end-of-task operations for SORTOUT before DFSORT is terminated.

VSAM User Exit Functions

There are three user exits that can be used with VSAM SORTIN, SORTINnn, and SORTOUT data sets (but not with OUTFIL data sets), to supply passwords or a user exit list to journal a VSAM data set. They can carry out other VSAM exit functions except EODAD. The user exits are E18 for sort SORTIN, E38 for merge SORTINnn or copy SORTIN, and E39 for SORTOUT.

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 user exit for sorts using tape work data sets. For a sort that uses tape data sets, you can choose between sorting current records only, trying to complete the sort, or terminating DFSORT. For more details, see “Exceeding Tape Work Space Capacity” on page 547.

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 user exits. To write SORTOUT labels, use the E19 and E39 user exits. If you have an end-of-file routine you want to use for SORTIN, include it at the E18 user 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 user exits.

Note: If a user exit requests termination for an application using a SmartBatch pipe data set, DFSORT will terminate with user abend zero. This allows for appropriate error propagation by the system to other applications that may be accessing the same SmartBatch pipe data set.

Addressing and Residence Modes for User Exits

To allow user exits called by Blockset or Peerage/Vale to reside above or below 16MB virtual, use either 24-bit or 31-bit addressing, and use a user exit address constant, DFSORT supplies these features:

- To ensure that DFSORT enters your user exit with the correct addressing mode, you must observe these rules:
  - If the user exit name is specified in a MODS control statement, the user 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 user exit is entered with 24-bit addressing in effect.
  - If the address of the user exit is passed to DFSORT via the extended parameter list (preloaded exit), the user exit is entered with 24-bit addressing in effect if bit 0 of the user exit address in the list is 0 or with 31-bit addressing in effect if bit 0 of the user 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, E32, or E35 user 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 user exit is entered with 24-bit addressing in effect, DFSORT passes clean (zeros in the first 8 bits) 24-bit addresses to the user exit. Such a user exit must pass 24-bit addresses back to DFSORT. These must be clean 24-bit addresses if the user exit returns to DFSORT with 31-bit addressing in effect.

– If the user exit is entered with 31-bit addressing in effect, DFSORT passes clean 24-bit addresses to the user exit. Such a user 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.

Note: For a conventional merge or tape work data set sort application, user exits:

– must reside below 16MB virtual

– must use 24-bit addressing mode

– must not use a user exit address constant.

How User Exit Routines Affect DFSORT Performance

Before writing a user exit routine, 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 perform 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 user 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.
- Using INCLUDE, OMIT, INREC, OUTFIL, OUTREC, and SUM instead of user exit routines allows DFSORT to perform more efficiently.

Summary of Rules for User Exit Routines

When preparing your routines, remember that:

- User-written routines must follow standard linkage conventions and use the required interfaces. COBOL E15 and E35 user exits must use the special interface provided.
- To use an E32 user exit, your invoking program must pass its address to DFSORT in the parameter list.
- To use any other user exit, you must associate your routine with the appropriate user exits using the MODS control statement. See “MODS Control Statement” on page 127.
- Your invoking program can alternatively pass the address of an E15, E18, E35, and E39 user exit to DFSORT in the parameter list.
- When Blockset or Peerage/Vale is used and your user exits are reenterable, the entire DFSORT program is reenterable.
- If you are using ISCII/ASCII input, remember that data presented to your user exits at user exits are in EBCDIC format. If the E61 user 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.

Loading User Exit Routines

You must assemble or compile each user exit as a separateprogram. 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 user exits 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 user exit using the external entry name of the routine (for example, the CSECT name).
3. Specify the appropriate ALIAS name for each user exiton the MODS statement.

DFSORT includes the names and locations of your user exits in the list of modules to be run during each phase. No user exit is loaded more than once in a program phase, but the same user 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 a user exit must include storage for the user exit itself as well as any storage used by the user exit outside of the load modules such as I/O buffers or COBOL library subroutines. If you specify a ddname for a user exit in the MODS statement, it must match the DD statement that defines the library containing that user 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 DFSORTreturn 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 performed.

The general registers used by DFSORT for linkage and communication of parameters observe operating system conventions. When your routine gets control, the general registers have the following contents:

Register	Contents
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 user 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 user exit. This register can be used as a base register for your user exit; your user exit can also use it to pass return codes to DFSORT.

You can return control to DFSORT by performing 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 a user exit.

Your user 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.

Linkage Examples

When calling your user 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 then makes a branch to your routine.

Your routine for the E15 user 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 a user 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 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 *Application Development Macro Reference*

Dynamically Link-Editing User Exit Routines

You can dynamically link-edit any user exit routine written in anylanguage that has the ability to pass the location or 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 user exits written in COBOL.

Dynamic link-editing does not support AMODE 31 or RMODE 31 for the link-edit option T. The user exits that are link-edited *together* by DFSORT are not loaded above 16MB virtual and cannot be entered in 31-bit addressing mode. User exits link-edited with the S option retain the AMODE and RMODEattributes of the object modules and are loaded above or below 16MB virtual depending upon the load module's RMODE; they are entered in the addressing mode of the user exit.

Notes:

1. The Blockset technique is not used for dynamic link-editing.
2. Dynamic link-editing cannot be used with copy.

Summary of Rules for User Exit Routines

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 user 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 E35, the following assembler instructions must be included in the user exit routine associated with the E35 user exit:

```
      ENTRY E35
E35    .
       .
```

or

```
E35  CSECT
       .
       .
```

In all other circumstances, the user exit is *not* required to have an entry point that has the same name as that of the associated program user exit.

Assembler User Exit Routines (Input Phase User Exits)

You can use these program user exits in the DFSORTinput phase:

E11
E15
E16
E17
E18
E19
E61

These user exits are discussed in sequence. To determine whether a particular user exit can be used for your application, refer to Table 36 on page 283 and Table 37 on page 284.

E11 User Exit: Opening Data Sets/Initializing Routines

You might use routines at this user exit to open data sets needed by yourother routines in the input phase. It can also be used to initialize your other routines. Return codes are not used, however.

Note: To avoid special linkage editor requirements (see “Summary of Rules for User Exit Routines” on page 287), you can include these functions in your E15 user exit rather than in a separate E11 user exit routine.

E15 User Exit: Passing or Changing Records for Sort and Copy Applications

If you write your E15 user exit in COBOL, see “COBOL User Exit Routines” on page 309 and “COBOL E15 User Exit: Passing or Changing Records for Sort” on page 312.

The EXITCK option 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 327.

DFSORT enters the E15 user 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 user exit tells DFSORT, with a return-code of 8, not to return.

See Figure 17 on page 282 for logic flow details.

Some uses for the E15 user exit are:
- Adding records to an input data set
- Passing an entire input data set to DFSORT
- Deleting records from an input data set
- Changing records in an input data set.

Notes:

1. If your E15 user exit is processing variable-length records, include a 4-byte RDW at the beginning of each record you change or insert, before you pass it back to DFSORT. The format of an RDW is described in *Using Data Sets* or *System Programming Reference*. (Alternatively, you can pad records to the maximum length and process them as fixed-length.)
2. DFSORT uses the specified or defaulted value for L2 in the RECORD statement to determine the length of the records your E15 user exit passes back to DFSORT. For fixed-length records, be sure that the length of each record your E15 user exit changes or inserts corresponds to the specified or defaulted L2 value. For variable-length records, be sure that the RDW of each record your E15 user exit changes or inserts indicates a length that is less than or equal to the specified or defaulted L2 value. Unwanted truncation or abends may occur if DFSORT uses the wrong length for the records passed to it by your E15 user exit.

   For details of the L2 value, see “RECORD Control Statement” on page 260.
3. If you use the E15 user exit to pass all your records to DFSORT, you can omit the SORTIN DD statement, in which case you must include a RECORD statement in the program control statements.
4. If you invoke DFSORT from an assembler program and pass the address of your E15 user exit in the parameter list, DFSORT ignores the SORTIN data set and terminates if you specify E15 in a MODS statement.
5. If you omit the SORTIN DD statement, or it is ignored, all input records are passed to DFSORT through your routine at user exit 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 user exit after the last record has been passed, you return to DFSORT with a return code of 8 in register 15, which indicates ″do not return.″
6. DFSORT continues to reenter your E15 user exit until a return code of 8 is received. However, if STOPAFT is in effect, no additional records are inserted to DFSORT after the STOPAFT count is satisfied (even if you pass back a return code of 12).
7. An RDW must be built for variable-length VSAM records (see *Using Data Sets*).

Information DFSORT Passes to Your Routine at E15 User Exit

Your E15 user exit routine 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**. End of input is reached when there are no more records to pass to your E15 user exit; DFSORT indicates end of input by setting this address to zero before entering your E15 user exit. If there are no records in the input data set (or no input data set), this address is zero the first time your E15 is entered.

After end of input is reached, DFSORT continues to enter your user exit routine until you pass back a return code of 8.

Your E15 user exit must not change the address of the new record.

- **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 user exit the first time it is entered. This address constant can be changed by your E15 user exit any time it is entered; the address constant is passed along on subsequent entries to your E15 user exit and also on the first entry to your E35 user exit. For example, you can obtain a dynamic storage area, use it in your E15 user exit, and pass its address to your E35 user exit.

Note: The user exit address constant must not be used for a tape work data set sort application.

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:

Table 38. E15 User Exit Parameter List

Bytes 1 through 4	Address of the new record
Bytes 5 through 8	User exit address constant

E15 Return Codes

Your E15 routine must pass a return code to DFSORT. Following are the return codes for the E15 user exit:

Return Code	Description
00 (X'00')	No Action/Record Altered
04 (X'04')	Delete Record
08 (X'08')	Do Not Return
12 (X'0C')	Insert Record
16 (X'10')	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).

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 user exit is not used again during the DFSORT application. You need not place an address in general register 1 when you return with a return code of 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*. It does this by passing your routine a zero address in the parameter list.

If your user exit routine passes a return code of 8 to DFSORT when input records still remain to be processed, the remaining records are processed by DFSORT, but are *not* passed to your user exit.

12: Insert Record
To add a record 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 keeps returning 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 327 for complete details of the meanings of return codes in various situations.

Storage Usage for E15 User Exit

DFSORT obtains storage (using GETMAIN or STORAGE OBTAIN) for the parameter list and the records it passes to your E15 user exit routine. You must not attempt to modify or free the storage obtained by DFSORT.

If you need to obtain storage for use by your E15 user exit routine, such as to pass altered records to DFSORT, you can use the following strategy:

1. The first time your exit is called, obtain the storage you need
2. Use the storage you obtained each time your exit is called
3. Free the storage before you pass back return code 8 to DFSORT

Note: When you obtain your storage you can save its address in the user exit address constant and restore it on each subsequent call to your exit.

E16 User Exit: Handling Intermediate Storage Miscalculation

For a tape work data set sort or a Peerage/Vale sort without work data sets,you would use a routine at this user 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 user exit is ignored for a sort with work data sets because DFSORT uses the WRKSEC option to determine whether secondary allocation is allowed. See “SORTWKdd DD Statement” on page 69. See also “Exceeding Tape Work Space Capacity” on page 547.

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 E16 routine must pass a return code to DFSORT. Following are the return codes for the E16 user exit:

Return Code	Description
00 (X'00')	Sort Current Records Only
04 (X'04')	Try to Sort Additional Records
08 (X'08')	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 (zero). 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 Tape Work Space Capacity” on page 547.

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 User Exit: Closing Data Sets

Your E17 user exit routine is entered 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 287), you can include these functions in your E15 user exits rather than in a separate E17 user exit routine.

E18 User Exit: Handling Input Data Sets

You can use this user exit to handle special I/O conditions for QSAM/BSAM and VSAM input data sets.

Using E18 User Exit with QSAM/BSAM

Your routines at this user exit can pass DFSORTa parameter list containing the specifications for three data control block (DCB) fields: SYNAD, EXLST, and EROPT. Your E18 user 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 at the beginning of each phase so that DFSORT can obtain the parameter lists. The routines are entered again during processing of the phase at the points indicated in the parameter lists. For example, if you choose the EXLST option, DFSORT enters your E18 user 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 User Exit: Before returning control to DFSORT, your routine passes the DCB fields in a parameter list by placing the parameter list address 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 as follows:

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
Contains the location of yourread 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 user exit routine. When the routine receives control, it must *not* store registers in the save area pointed to by register 13.

EXLST
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 can only be used for EXLST routines associated with opening the first SORTIN data set.

EROPT
Indicates what action DFSORTmust 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, bytes 2 and 3 must contain zeros, and byte 4 must contain X'01'. You can use the assembler instruction DC AL3(1) to set up bytes 2 through 4.

EODAD
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 *Macro Instructions for Data Sets*

Using E18 User Exit with VSAM

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

E18 User Exit Restrictions: If passwords are to be entered through a user 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 User Exit: When you return to DFSORT, you must place the address of a parameter list in general register 1:

Byte 1	Bytes 2 through 4
05	Address of VSAM user 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 User Exit List: The VSAM user exit list must be built using the VSAM EXLST macro instruction giving the addresses of your routines handling VSAM user exit functions. VSAM branches directly to your routines which must return to VSAM via register 14.

Any VSAM user exit function available for input data sets can be used except EODAD. If you need to do EODAD processing, write a LERAD user 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 *Macro Instructions for Data Sets.*

Figure 18 shows 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
```

Figure 18. E18 User Exit Example

[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 user exit is the data control block user exit.

For more information, refer to *Using Data Sets.*

E19 User Exit: Handling Output to Work Data Sets

This user exit is used to handle write error conditions in the input phasewhen 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 User Exit with QSAM/BSAM

Your routines at this user exit can pass DFSORT a parameter listcontaining 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 User Exit: Before returning control to DFSORT, your routine passes the DCB fields in a parameter list by placing the parameter list address 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 of the list 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 errorroutine. 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. These pointers point to routines that are 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 *Macro Instructions for Data Sets.*

E61 User Exit: Modifying Control Fields

You can use a routine at this user 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 “MERGE Control Statement” on page 124 and “SORT Control Statement” on page 264. After your routine modifies the control field, DFSORT collates the records in ascending order using the format(s) specified.[16]

Notes:

1. Routine E61 will not be used with EFS fields that have a D1 format.
2. Although your E61 routine alters control fields before a compare, your original records are not altered.
3. If locale processing is used for SORT or MERGE fields, an E61 user exit must not be used. DFSORT’s locale processing may eliminate the need for an E61 user exit. See “OPTION Control Statement” on page 134 for information related to locale processing.

Some Uses of E61 User Exit

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 collating of national characters, you can do so without the need for an E61 user exit routine by using the ALTSEQ control statement (as described in “ALTSEQ Control Statement” on page 85).

Information DFSORT Passes to Your Routine at E61 User Exit

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 user exit is as follows:

16. 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.

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.

To alter the control field, change the control field image at the indicated address (changing the address itself will have no effect).

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 user exit E61.

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 user exit 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 as illustrated by the E61 example in Figure 24 on page 309.

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 User Exit Routines (Output Phase User Exits)

You can use these program user exits located in the DFSORToutput phase:

E31
E32
E35
E37
E38
E39

The functions of these user exits are discussed in sequence.

E31 User Exit: Opening Data Sets/Initializing Routines

You might use routines at this user exit to open data sets neededby 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 287), you can include these functions in your E35 user exit rather than in a separate E31 routine.

E32 User Exit: Handling Input to a Merge Only

This user exit can be used only in a merge operationinvoked from a program and cannot be specified on the MODS statement. When an E32 user exit is activated, it must supply all input to the merge. DFSORT ignores SORTINnn data sets when an E32 user exit is used.

You must indicate the number of input files you want to merge using either (1) the FILES=n option on the MERGE control statement, or (2) the X'04' entry in the 24-bit parameter list. Your E32 user exit routine must insert records for these files as DFSORT requests them.

If input is variable-length records, you must be sure the beginning of each record contains a 4-byte RDW before merged. The format of an RDW is described in *Macro Instructions for Data Sets* (Alternatively, you can declare the records as fixed-length and pad them to the maximum length.)

See Figure 17 on page 282 for logic flow details.

Information DFSORT Passes to Your Routine at E32 User Exit

Your E32 user exit routine is entered each time the merge program requires a new input record. DFSORT passes three words to your routine:

- **The increment of the next file to be used for input**. 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.
- **The address of the next input record**. 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.
- **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 E32 user exit the first time it is entered. This address constant can be changed by your E32 user exit any time it is entered; the address constant is passed along on subsequent entries to your E32 user exit and E35 user exit. For example, you can obtain a dynamic storage area, use it in your E32 user exit, and pass its address to your E35 user exit.

Note: The user exit address constant must not be used for a Conventional merge application.

In general register 1, DFSORT places the address of a parameter list that contains the file increment, the record address and the user address constant. The list is three fullwords long and begins on a fullword boundary. The format of the parameter list is:

Table 39. E32 User Exit Parameter List

Bytes 1 through 4	Increment of next file to be used for input
Bytes 5 through 8	Address of next input record
Bytes 9 through 12	User exit address constant

Before returning control to DFSORT, you must:

- Place the address of the next input record from the requested input file in the second word of the parameter list
- Put the return code in register 15.

E32 Return Codes

Your E32 routine must pass a return code to DFSORT. Following are the return codes for the E32 user exit:

Return Code	Description
08 (X'08')	End of input for requested file
12 (X'0C')	Insert record
16 (X'10')	Terminate DFSORT

8: End of input for requested file
DFSORT continues to return control to the user routine until it receives a return code of 8 for every input file. After that, the user exit is not used again during the DFSORT application. You need not place an address in the second word of the parameter list when you return with a return code of 8.

12: Insert Record
To add a record from the requested input file, place the address of the record to be added in the second word of the parameter list and return to DFSORT with a return code of 12. *DFSORT keeps returning to your routine until you pass a return code of 8 for every input file*.

16: Terminate DFSORT
If you want to terminate DFSORT, return with a code of 16. DFSORT then returns to its calling program with a return code of 16.

E35 User Exit: Changing Records

If you write your E35 user exit in COBOL,see “COBOL User Exit Routines” on page 309 and “COBOL E35 User Exit: Changing Records” on page 318.

The EXITCK option 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 327.

DFSORT enters the E35 user exit routine each time it prepares to place a record in the output area.

See Figure 17 on page 282 for logic flow details.

Some uses for the E35 user exit are:
- Adding records for output data sets
- Omitting records for output data sets
- Changing records for output data sets

Notes:

1. If your E35 user exit is processing variable-length records, include a 4-byte RDW at the beginning of each record you change or insert, before you pass it back to DFSORT. The format of an RDW is described in *Using Data Sets* or *System Programming Reference*. (Alternatively, you can pad records to the maximum length and process them as fixed-length.)
2. DFSORT uses the specified or defaulted value for L3 in the RECORD statement to determine the length of the records your E35 user exit passes back to DFSORT. For fixed-length records, be sure that the length of each record your E35 user exit changes or inserts corresponds to the specified or defaulted L3 value. For variable-length records, be sure that the RDW of each record your E35 user exit changes or inserts indicates a length that is less than or equal to the specified or defaulted L3 value. Unwanted truncation or abends may occur if DFSORT uses the wrong length for the records passed to it by your E35 user exit.

   For details of the L3 value, see “RECORD Control Statement” on page 260.
3. If you use the E35 user exit to dispose of all your output records, you can omit the SORTOUT DD statement.
4. If you invoke DFSORT from a program and you pass the address of your E35 user exit in the parameter list:
   - DFSORT ignores the SORTOUT data set (but not any OUTFIL data sets).
   - DFSORT terminates if you specify E35 in a MODS statement.
5. If you omit the SORTOUT DD statement or it is ignored, and you do not specify any OUTFIL data sets, your E35 user 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 to indicate“do not return.”
6. Remember that if input records are variable-length from a VSAM data set, they will have been prefixed by a 4-byte RDW.
7. After records have been put into the output area, their lengths cannot be increased.
8. For a merge application, records deleted by an E35 user exit routine are not sequence-checked. If you use an E35 user exit routine without an output 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 User Exit

Your E35 user exit routine is entered 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. End of input is reached when there are no more records to pass to your E35 user exit; DFSORT indicates end of input by setting this address to zero before entering your E35 user exit.

  After end of input is reached, DFSORT continues to enter your user exit routine until a return code of 8 is passed back.

  Your E35 user exit must not change the address of the record leaving DFSORT.
- **The address of a record in the output area** 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** is passed to your user exit exactly as it was set by your E15 or E32 user exit or invoking program's parameter list.

Note: The user exit address constant must not be used for a Conventional merge or tape work data set sort application.

In general register 1, DFSORT places the address of a parameter list that contains the two record addresses and the user address constant. The list is three fullwords long and begins on a fullword boundary. The format of the parameter list is:

Table 40. E35 User Exit Parameter List

Bytes 1 through 4	Address of record leaving DFSORT
Bytes 5 through 8	Address of record in output area
Bytes 9 through 12	User exit address constant

E35 Return Codes

Your E35 routine must pass a return code to DFSORT. Following are the return codes for the E35 user exit:

Return Code	Description
00 (X'00')	No Action/Record Altered
04 (X'04')	Delete Record
08 (X'08')	Do Not Return
12 (X'0C')	Insert Record
16 (X'10')	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).

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 user exit is not used again during the DFSORT application. When you return with a return code of 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*. This is done by passing a zero as the address of the record leaving DFSORT.

If you do not have an output 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 user exit routine passes a return code of 8 to DFSORT when input records still remain to be processed, the remaining records are processed by DFSORT, but are *not* passed to your user exit.

12: Insert Record
To add an output record 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 user exit for the record leaving DFSORT. DFSORT places the address of the inserted record into the output area. You can make more insertions at that point, or delete the record leaving DFSORT.

DFSORT does not perform sequence checking for DASD work data set sorts. For tape work data set 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 to the system with a return code of 16.

See “E15/E35 Return Codes and EXITCK” on page 327 for complete details of the meanings of return codes in various situations.

Summing Records at E35 User Exit: You can use the SUM control statement to sum records. However, you can sum records for output 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 continue summing. 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.

Storage Usage for E35 User Exit

DFSORT obtains storage (using GETMAIN or STORAGE OBTAIN) for the parameter list and the records it passes to your E35 user exit routine. You must not attempt to modify or free the storage obtained by DFSORT.

If you need to obtain storage for use by your E35 user exit routine, such as to pass altered records to DFSORT, you can use the following strategy:

1. The first time your exit is called, obtain the storage you need
2. Use the storage you obtained each time your exit is called
3. Free the storage before you pass back return code 8 to DFSORT.

Note: When you obtain your storage you can save its address in the user exit address constant and restore it on each subsequent call to your exit.

E37 User Exit: Closing Data Sets

Your E37 user exit routine is entered once at the end ofthe 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 287), you can include these functions in your E35 user exit rather than in a separate E37 user exit.

E38 User Exit: Handling Input Data Sets

The routine here is the same as for E18. If the Blockset or Peerage/Vale technique is selected, I/O error conditions cannot be handled through the E38 user exit.

Using E38 User Exit with VSAM

This user exit can be used during a merge or copy toinsert VSAM passwords into VSAM input ACBs and to perform various VSAM user exit functions. The following example 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
```

Figure 19. E38 User Exit Example

E39 User Exit: Handling Output Data Sets

Your E39 user exit routine is entered for the SORTOUT data set, but not for OUTFIL data sets.

Using E39 User Exit with QSAM/BSAM

The technique is the same as for E19 for QSAM/BSAM. See “E19 User Exit: Handling Output to Work Data Sets” on page 297 for details.

Using E39 User Exit with VSAM

For VSAM, this user exit can be used to insert VSAM passwords intothe VSAM SORTOUT ACB and to perform various VSAM user 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
```

Figure 20. E39 User Exit Example

Sample Routines Written in Assembler

This section provides some sample program user exits written in assembler.

E15 User Exit: Altering Record Length

This routine changes the variable-length input records making them all the same length.

```
E15      CSECT
* IF A RECORD IS GREATER THAN 204 BYTES, TRUNCATE IT TO 204 BYTES.
* IF A RECORD IS LESS THAN 204 BYTES, PAD IT OUT TO 204 BYTES.
* ALL OF THE RESULTING RECORDS WILL BE 204 BYTES LONG
* (4 BYTES FOR THE RDW AND 200 BYTES OF DATA).
         USING E15,12             SHOW BASE REG
         STM   14,12,12(13)       SAVE ALL REGS EXCEPT 13
         LA    12,0(0,15)         SET BASE REG
         ST    13,SAVE15+4        SAVE BACKWARD POINTER
         LA    14,SAVE15          SET FORWARD POINTER
         ST    14,8(13)           IN SAVE AREA
         LR    13,14              SET OUR SAVE AREA
         LR    2,1                SAVE PARM LIST POINTER
         L     3,0(,2)            LOAD ADDR OF RECORD
         LTR   3,3                EOF
         BZ    EOF                YES - DO NOT RETURN
         LH    4,0(,3)            GET RDW
         CH    4,CON204           IS RDW EQ 204
         BE    ACCEPT             YES-ACCEPT IT
         BL    PAD                LESS THAN 204-PAD
         LH    4,CON204           LIMIT LENGTH TO 204
         B     TRUNC              MORE THAN 204-TRUNCATE
PAD      DS    0H                 PAD OR TRUNCATE
         MVI   DATA,X'00'         ZERO OUT THE BUFFER
         MVC   DATA+1(199),DATA
         BCTR  4,0                DECREASE RDW FOR EXECUTE
TRUNC    DS    0H                 PAD OR TRUNCATE
         EX    4,MVPAD            MOVE RECORD INTO PAD/TRUNCATE BUFFER
         MVC   NEWRDW(2),CON204   SET NEW RDW TO 204
         LA    3,BUFFER           POINT TO PADDED/TRUNCATED RECORD
ACCEPT   DS    0H
         SR    15,15              SET RC=0
         LR    1,3                SET RECORD POINTER
         B     GOBACK
EOF      LA    15,8               EOF - SET RC=8
GOBACK   L     13,4(,13)
         L     14,12(,13)
         LM    2,12,28(13)        RESTORE REGS
         BR    14                 RETURN
MVPAD    MVC   BUFFER(*-*),0(3)   FOR EXECUTE
SAVE15   DS    18F
CON204   DC    H'204'
BUFFER   DS    0H
NEWRDW   DS    H                  NEW RDW OF 204
         DC    H'0'
DATA     DC    XL200'00'          BUFFER FOR PADDING/TRUNCATING
         END
```

Figure 21. E15 User Exit Example

E16 User Exit: Sorting Current Records When NMAX Is Exceeded

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
```

Figure 22. E16 User Exit Example

E35 User Exit: Altering Record Length

This routine changes the variable-length output records making them all the same length.

```
E35     CSECT
* IF A RECORD IS GREATER THAN 204 BYTES, TRUNCATE IT TO 204 BYTES.
* IF A RECORD IS LESS THAN 204 BYTES, PAD IT OUT TO 204 BYTES.
* ALL OF THE RESULTING RECORDS WILL BE 204 BYTES LONG
* (4 BYTES FOR THE RDW AND 200 BYTES OF DATA).
         USING E35,12             SHOW BASE REG
         STM   14,12,12(13)       SAVE ALL REGS EXCEPT 13
         LA    12,0(0,15)         SET BASE REG
         ST    13,SAVE15+4        SAVE BACKWARD POINTER
         LA    14,SAVE15          SET FORWARD POINTER
         ST    14,8(13)           IN SAVE AREA
         LR    13,14              SET OUR SAVE AREA
         LR    2,1                SAVE PARM LIST POINTER
         L     3,0(,2)            LOAD ADDR OF RECORD
         LTR   3,3                EOF
         BZ    EOF                YES - DO NOT RETURN
         LH    4,0(,3)            GET RDW
         CH    4,CON204           IS RDW EQ 204
         BE    ACCEPT             YES-ACCEPT IT
         BL    PAD                LESS THAN 204-PAD
         LH    4,CON204           LIMIT LENGTH TO 204
         B     TRUNC              MORE THAN 204-TRUNCATE
PAD      DS    0H                 PAD OR TRUNCATE
         MVI   DATA,X'00'         ZERO OUT THE BUFFER
         MVC   DATA+1(199),DATA
         BCTR  4,0                DECREASE RDW FOR EXECUTE
TRUNC    DS    0H                 PAD OR TRUNCATE
         EX    4,MVPAD            MOVE RECORD INTO PAD/TRUNCATE BUFFER
         MVC   NEWRDW(2),CON204   SET NEW RDW TO 204
         LA    3,BUFFER           POINT TO PADDED/TRUNCATED RECORD
ACCEPT   DS    0H
         SR    15,15              SET RC=0
         LR    1,3                SET RECORD POINTER
         B     GOBACK
EOF      LA    15,8               EOF - SET RC=8
GOBACK   L     13,4(,13)
         L     14,12(,13)
         LM    2,12,28(13)        RESTORE REGS
         BR    14                 RETURN
MVPAD    MVC   BUFFER(*-*),0(3)   FOR EXECUTE
SAVE15   DS    18F
CON204   DC    H'204'
BUFFER   DS    0H
NEWRDW   DS    H                  NEW RDW OF 204
         DC    H'0'
DATA     DC    XL200'00'          BUFFER FOR PADDING/TRUNCATING
         END
```

Figure 23. E35 User Exit Example

E61 User Exit: Altering Control Fields

This routine can be used to change the order of binary control fields passed to it (that is, those for which 'E' is specified) from ascending order to descending order.

```
* E61 PARAMETER LIST DSECT
PARML    DSECT
         DS    3C
PARMNUM  DS    C   CONTROL FIELD NUMBER
PARMPTR  DS    A   ADDRESS OF CONTROL FIELD
         DS    2C
PARMLEN  DS    H   CONTROL FIELD LENGTH
*
E61REV   CSECT
* CHANGE THE ORDER OF EACH  CONTROL FIELD PASSED TO THIS ROUTINE
* FROM ASCENDING TO DESCENDING BY REVERSING ALL OF THE BITS.
* ASSUMES THAT ONLY BI CONTROL FIELDS ARE PASSED.
         USING E61REV,12          SHOW BASE REG
         STM   14,12,12(13)       SAVE ALL REGS EXCEPT R13
         LA    12,0(0,15)         SET BASE REG
         ST    13,SAVE61+4        SAVE BACKWARD POINTER
         LA    14,SAVE61          SET FORWARD POINTER
         ST    14,8(13)           IN SAVE AREA
         LR    13,14              SET OUR SAVE AREA
         LR    3,1                SET PARM LIST POINTER
         USING PARML,3
         L     4,PARMPTR         GET POINTER TO CONTROL FIELD IMAGE
         LH    5,PARMLEN         GET LENGTH OF CONTROL FIELD
         BCTR  5,0               SUBTRACT 1 FOR EXECUTE
         EX    5,REVCF           CHANGE FROM ASCENDING TO DESCENDING
GOBACK   L     13,4(,13)
         LM    14,12,12(13)       RESTORE REGS
         BR    14                 RETURN
REVCF    XC    0(*-*,4),REVFF    REVERSE CONTROL FIELD BITS
SAVE61   DS    18F
REVFF    DC    256X'FF'
         LTORG
         END
```

Figure 24. E61 User Exit Example

COBOL User Exit Routines

You can perform the same tasks with E15 and E35 user exit routineswritten in COBOL that you can perform with E15 and E35 user exit routines written in assembler. However, COBOL routines differ from assembler routines in the way they pass information between themselves and DFSORT.

- COBOL routines 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.
- COBOL routines must use RETURN-CODE, a COBOL special register. Assembler uses register 15 for the return code.
- COBOL routines must use return code 20 to alter or replace a record. Assembler uses return code 0.
- COBOL routines can use the user exit area for E15/E35 communication. Assembler uses the user address constant.

COBOL User Exit Requirements

The following rules apply to COBOL user exits. Failure to observe these COBOL user exit rules can result in termination or unpredictable results.

Note: ″VS COBOL II or later″ means VS COBOL II, COBOL for MVS & VM, COBOL for OS/390 & VM, and Language Environment.

- User exits written in COBOL must not use STOP RUN statements. To return to DFSORT, use the GOBACK statement.
- VS COBOL II user exits must be compiled with the RES/RENT compiler option.
- Compilation of OS/VS COBOL user exits with the RES compiler option aids migration to VS COBOL II or later; however, user exits compiled with NORES will run under DFSORT.
- If a user 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 for MVS & VM or COBOL for OS/390 & VM.

  Note: READY TRACE and EXHIBIT are only supported in OS/VS COBOL.
- COBOL user exits must not contain a SORT or a MERGE verb.
- When coding the MODS control statement to describe a COBOL user exit, use C for the fourth parameter. This instructs DFSORT to build the correct parameter list.
- If invoking DFSORT from a VS COBOL II or later program, you can use a COBOL E15 if the VS COBOL II or later FASTSRT option is in effect for input and a COBOL E35 if FASTSRT is in effect for output. The COBOL user exits must be compiled with VS COBOL II or later.
- If you are running user exits compiled with VS COBOL II, you must use the VS COBOL II library or Language Environment 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 you are running user exits compiled with COBOL for MVS & VM or COBOL for OS/390 & VM, you must use the Language Environment 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 or Language Environment Library even if your COBOL user exit is compiled by the OS/VS COBOL compiler.
- If you run user 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 a user exit compiled with the OS/VS COBOL NORES option. See your OS/VS COBOL manual for information on options that require the COBOL library.
- User exits compiled with OS/VS COBOL can be run with either the OS/VS COBOL or VS COBOL II library or, in some cases, with no library.
- COBOL for MVS & VM and COBOL for OS/390 & VM require that COBOL library routines in Language Environment must be available at run-time.
- User exits compiled with OS/VS COBOL and running 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 directly invoke DFSORT and COBEXIT=COB2 is in effect, you can use *either* a separately compiled COBOL E15 user exit *or* a separately compiled COBOL E35 user exit, but not both.
- When you invoke DFSORT from a VS COBOL II or later program, the following limitations apply when FASTSRT is in effect for:
  - Input only: You can use a separately compiled E15 user exit, but not a separately compiled E35 user exit.
  - Output only: You can use a separately compiled E35 user exit, but not a separately compiled E15 user 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 user exits are found together, DFSORT copy processing terminates. Message ICE161A is issued.

COBOL Storage Requirements

If you are running COBOL user exits compiled with the RES compiler option, make sure that you have enough storage available for the COBOL library subroutines. (This does not apply if the library has been installed resident.)

Besides the minimum DFSORT main storage requirements, you need an additional 40KB of storage in your REGION for the OS/VS COBOL library subroutines and 150KB for the VS COBOL II library subroutines. Most of the VS COBOL II library subroutines can be resident above 16MB virtual. However, whether you can actually load the VS COBOL II library subroutines above 16MB virtual depends on how they were installed. In order to run Language Environment for MVS & VM, you need 1200KB. You can minimize the storage needed by Language Environment for MVS & VM below 16MB virtual by loading the COBPACKS above 16MB virtual. See *Language Environment for MVS & VM Installation and Customization Guide*, SC26-4817, or *Language Environment for OS/390 Installation and Customization Guide*, SC28–1941, for more information.

Under certain conditions, DFSORT can use all the storage in your REGION below 16MB virtual, thus leaving no room to load the COBOL library subroutines required during processing of your user exit.

Main storage is available above 16MB virtual unless the TMAXLIM or SIZE/MAINSIZE options specify an extremely high value (for example, your system limit for main storage above 16MB virtual). In that case, you can use the ARESALL or ARESINV option to release storage.

During processing, the actual amount of storage required for the COBOL library subroutines depends on the functions performed in the COBOL user exit. You must add a minimum of 40KB to the size of the user exit when running with the OS/VS COBOL library subroutines and, in most cases, 20KB when running with the VS COBOL II library or Language Environment for MVS & VM. If the user exit does I/O, additional storage must be reserved for the I/O buffers. Additional storage for buffers is specified by the *m* parameter on the MODS statement. A VS COBOL II or later user exit requires less storage than a similar OS/VS COBOL user exit because DFSORT automatically releases storage for some of the COBOL library subroutines before the user 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 70KB of storage when you are:

- Calling both E15 and E35 user exits
- Running with nonresident VS COBOL II library subroutines
- Performing a sort with DFSORT residing above 16MB virtual.

This can be done by adding 70KB more to one of the following:

- The m parameter of the MODS statement for the E35 user exit (m = E35 user exit size + 20KB+ 70KB)
- The RESALL option when SIZE/MAINSIZE=MAX is in effect.

COBOL User Exit Routines (Input Phase User Exit)

COBOL E15 User Exit: Passing or Changing Records for Sort

The EXITCK option 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 327.

DFSORT enters the E15 user 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 user exit tells DFSORT, with a return code of 8, not to return.

See Figure 17 on page 282 for logic flow details.

Some uses for the E15 user exit are:

- Adding records to an input data set
- Passing an entire input data set to DFSORT
- Deleting records from an input data set
- Changing records in an input data set.

Notes:

1. If both E15 and E35 user exits are used, they must be in the same version of COBOL.
2. If you use the E15 user exit to pass all your records to DFSORT, you can omit the SORTIN DD statement, in which case you must include a RECORD statement in the program control statements.
3. If you omit the SORTIN DD statement, all input records are passed to DFSORT through your COBOL E15 user exit. You return to DFSORT with a return code of 12. When DFSORT returns to the E15 user exit after the last record has been passed, you return to DFSORT with a return code of 8 in register 15, which indicates “do not return.”
4. DFSORT continues to reenter your E15 user exit until a return code of 8 is received. However, if STOPAFT is in effect, no additional records are inserted to DFSORT after the STOPAFT count is satisfied (even if you pass back a return code of 12).
5. You cannot use dynamic link-editing with a COBOL E15 user exit.

E15 Interface with COBOL

Each time the E15 user exit is called, DFSORT supplies the following fields:

- Record flags
- New record
- Length of the new record (for variable-length records)
- Length of user exit area
- User exit area.

COBOL User Exit Routines (Input Phase User Exit)

When E15 returns to DFSORT, the E15 user 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 user exit by setting the COBOL special register RETURN-CODE)
- Return record
- Length of the return record (for VLR)
- Length of user exit area
- User Exit area.

For more information on how these fields are used in a COBOL E15 user exit, see “E15 LINKAGE SECTION Fields for Fixed-Length and Variable-Length Records” on page 315.

Figure 25 on page 314 details the interface to COBOL for the E15 user exit.

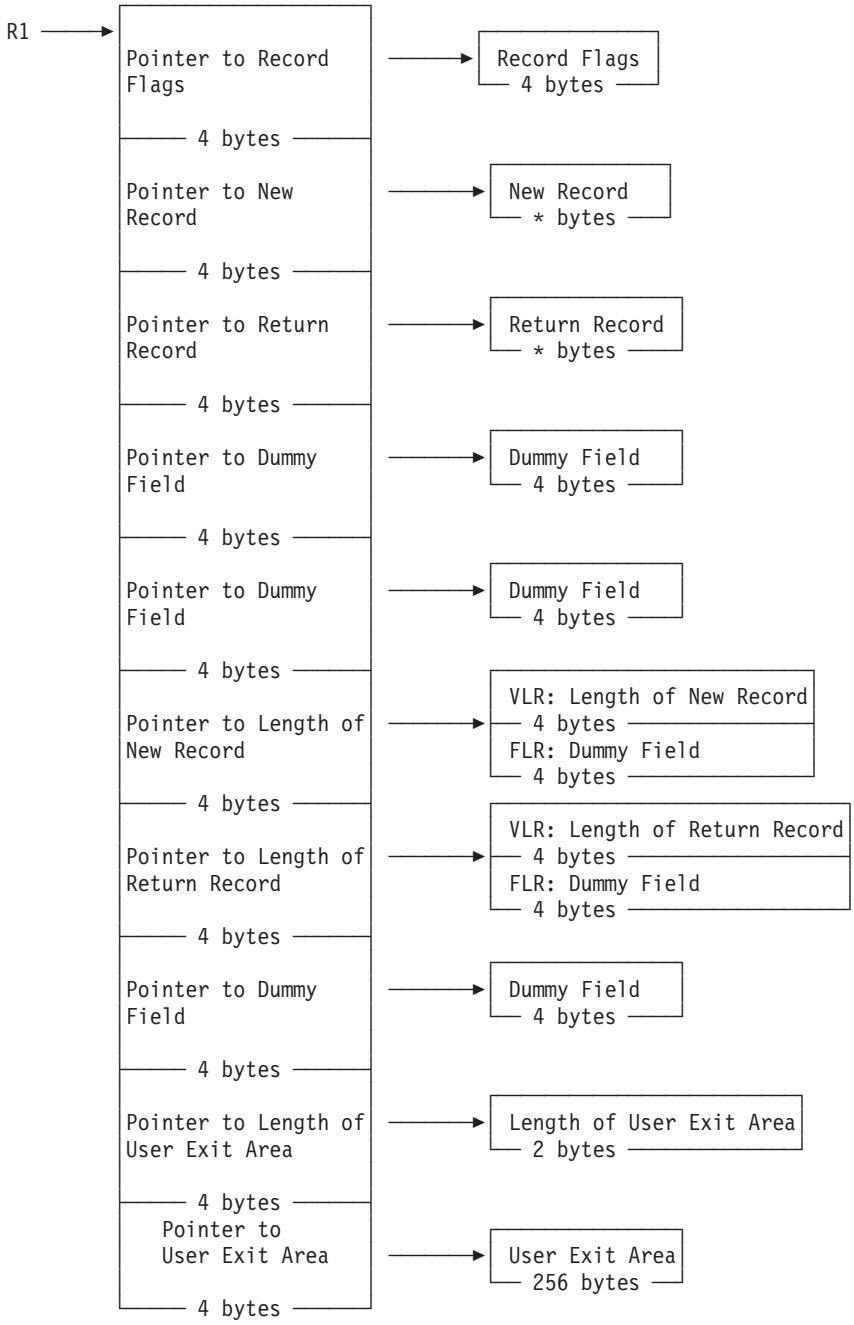


Figure 25. E15 DFSORT Interface with COBOL

E15 LINKAGE SECTION Examples: Figure 26 on page 315 is an example of the LINKAGE SECTION code for a fixed-length record (FLR) data set with a logical record length (LRECL) of 100. The example shows 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 26. LINKAGE SECTION Code Example for E15 (Fixed-Length Records)

Figure 27 is an example of the LINKAGE SECTION code for a variable-length record (VLR) data set with a maximum LRECL of 200. The example shows 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.
    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 27. 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 27 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)
  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 user exit, it passes the user exit a 256-byte user exit area field (EXITAREA). The first time the user exit area field is passed to your COBOL E15 user exit, it contains 256 blanks, and the user exit area length field (EXITAREA-LEN) contains 256.

  Any changes you make to the user exit area field or user exit area length fields are passed back both to your COBOL E15 user exit and your COBOL E35 user exit.

  Notes:

  1. Do not set the user exit area length field to more than 256 bytes.
  2. 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 user exit is 4 bytes less than the LRECL value, because COBOL does not include the 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 user exit.)
  3. 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 user exit area).
  4. DFSORT uses the specified or defaulted value for L2 in the RECORD statement to determine the length of the records your E15 user exit passes back to DFSORT. For fixed-length records, be sure that each record your E15 user exit changes or inserts has a length that is equal to the specified or defaulted L2 value. For variable-length records, be sure that each record your E15 user exit changes or inserts has a length that is less than or equal to the specified or defaulted L2 value. Unwanted truncation or abends may occur if DFSORT uses the wrong length for the records passed to it by your E15 user exit.

     For details of the L2 value, see “RECORD Control Statement” on page 260.

E15 Return Codes

Your COBOL E15 routine must pass a return code to DFSORT in the RETURN-CODE field, a COBOL special register. Following are the return codes for the E15 user exit:

Return Code	Description
00 (X'00')	No Action
04 (X'04')	Delete Record
08 (X'08')	Do Not Return
12 (X'0C')	Insert Record
16 (X'10')	Terminate DFSORT
20 (X'14')	Alter or 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 user exit is not used again 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 user exit routine passes a return code of 8 to DFSORT when input records still remain to be processed, the remaining records are processed by DFSORT but are *not* passed to your user 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 returns to 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 of 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 327 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 26 on page 315, 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 27 on page 315, 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 User Exit Routines (Output Phase User Exit)

COBOL E35 User Exit: Changing Records

The EXITCK option 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 information about E35 return codes in various situations with EXITCK=STRONG and EXITCK=WEAK, see “E15/E35 Return Codes and EXITCK” on page 327.

DFSORT enters the E35 user exit routine each time it prepares to place a record in the output area.

See Figure 17 on page 282 for logic flow details.

Some uses for the E35 user exit are:
- Adding records for output data sets
- Omitting records for output data sets
- Changing records for output data sets

When DFSORT indicates the end of the data set (record flags field set to 8), you must set RETURN-CODE to 8 (unless you are inserting records after the end of the data set); otherwise, DFSORT continues to enter E35.

Notes:

1. If both E15 and E35 user exits are used, they must be in the same version of COBOL.
2. If you use the E35 user exit to dispose of all your output records, you can omit the SORTOUT DD statement.
3. If you omit the SORTOUT DD statement and you do not specify any OUTFIL data sets, your E35 user 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 to indicate “do not return.”
4. You cannot use dynamic link-editing with a COBOL E35 user exit.

E35 Interface with COBOL

Each time your E35 user exit is called, DFSORT supplies the following fields:

- Record flags
- Record leaving DFSORT
- Length of record leaving DFSORT (for variable-length records)
- Length of user exit area
- User Exit area.

When your E35 user exit returns to DFSORT, the E35 user 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 user exit by setting the COBOL special register RETURN-CODE)
- Return record
- Length of return record (for variable-length records)
- Length of user exit area
- User exit area.

For more information on how these fields are used in a COBOL E35 user exit, see “E35 LINKAGE SECTION Fields for Fixed-Length and Variable-Length Records” on page 321.

Figure 28 on page 320 details the interface to COBOL for the E35 user exit.

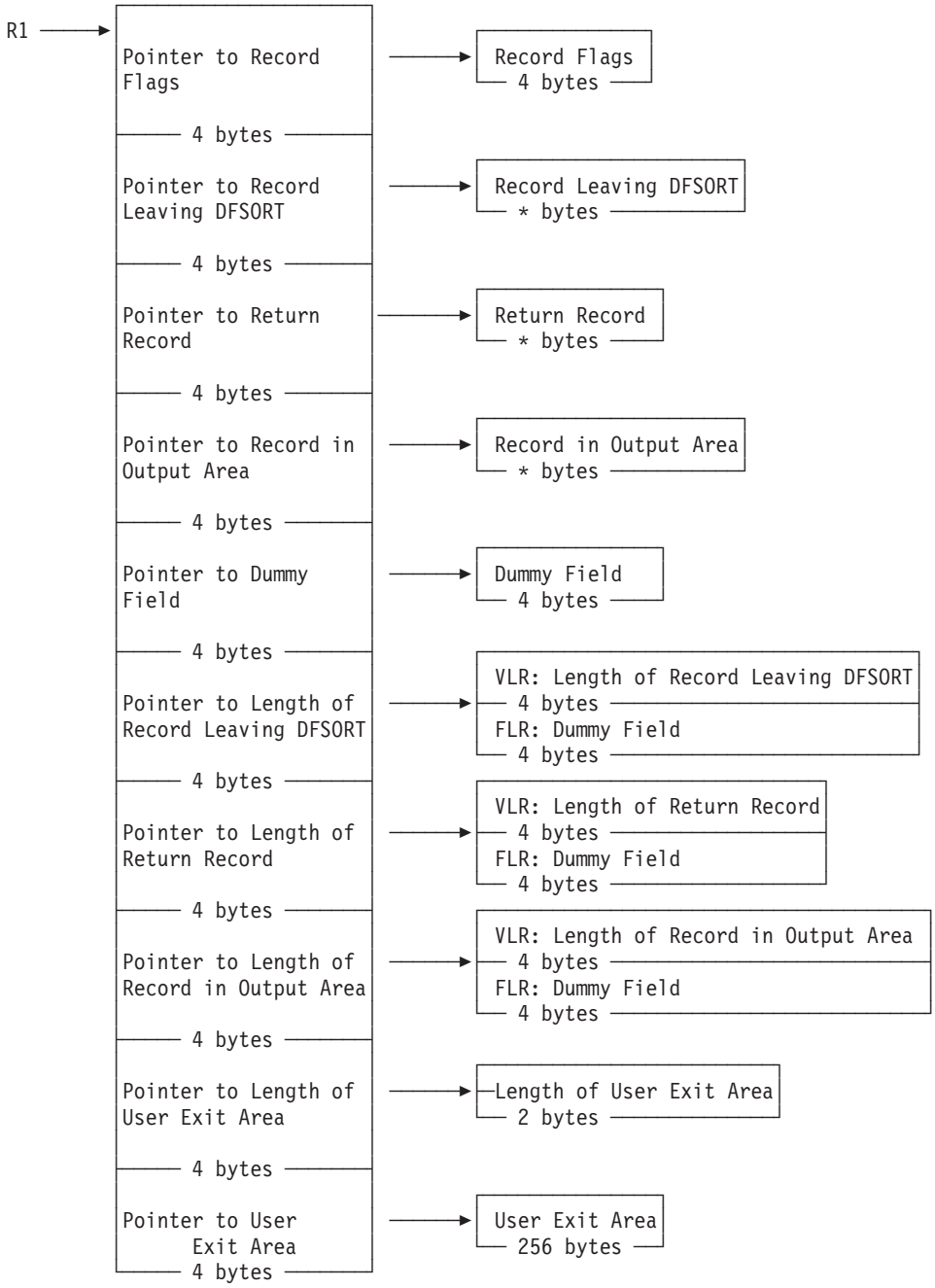


Figure 28. E35 Interface with COBOL

E35 LINKAGE SECTION Examples: Figure 29 is an example of the LINKAGE SECTION code for a fixed-length record (FLR) data set with a logical record length (LRECL) of 100. The example shows 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  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 29. LINKAGE SECTION Code Example for E35 (Fixed-Length Records)

Figure 30 is an example of the LINKAGE SECTION code for a variable-length record (VLR) data set with a maximum LRECL of 200. The example shows 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  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 30. 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 30 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 or 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 user 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 user exit area length field (EXITAREA-LEN) contains 256.

  Any changes you make to the user exit area field or user exit area length field are passed back to your COBOL E35 routine each time it is called by DFSORT.

  Notes:

  1. Do not set the user exit area length field to more than 256 bytes.
  2. VLR records have a 4-byte RDW field at the beginning of each record. The maximum record length plus the RDW will be the length defined for the LRECL attribute of your output data set. COBOL programs do not use the RDW and, therefore, the maximum length defined in your COBOL user exit is 4 bytes less than the LRECL value.
  3. 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 user exit area).
  4. DFSORT uses the specified or defaulted value for L3 in the RECORD statement to determine the length of the records your E35 user exit passes back to DFSORT. For fixed-length records, be sure that each record your E35 user exit changes or inserts has a length that is equal to the specified or defaulted L3 value. For variable-length records, be sure that each record your E35 user exit changes or inserts has a length that is less than or equal to the specified or defaulted L3 value. Unwanted truncation or abends may occur if DFSORT uses the wrong length for the records passed to it by your E35 user exit.

     For details of the L3 value, see “RECORD Control Statement” on page 260.

E35 Return Codes

Your COBOL E35 routine must pass a return code to DFSORT in the RETURN-CODE field, a COBOL special register. Following are the return codes for the E35 exit:

Return Code	Description
00 (X'00')	No Action
04 (X'04')	Delete Record
08 (X'08')	Do Not Return
12 (X'0C')	Insert Record
16 (X'10')	Terminate DFSORT
20 (X'14')	Alter or 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 user exit is not used again 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*. This is done by entering your routine with the record flags field set to 8.

If your user exit routine passes a return code of 8 to DFSORT when input records still remain to be processed, the remaining records are processed by DFSORT but are *not* passed to your user exit.

If you do not have an output 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).

12: Insert Record
If you want DFSORT to add an output record 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 returns to 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 DASD work data set sorts. For tape work data set sorts, DFSORT does not perform sequence checking on inserted records unless you delete the record leaving DFSORT and then 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 327 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 29 on page 321, 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 30 on page 321, 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 user exits written in COBOL.

COBOL E15 User Exit: Altering Records

Figure 31 shows 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 31. COBOL E15 Routine Example (FLR)

COBOL E35 User Exit: Inserting Records

Figure 32 shows 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 32. COBOL E35 Routine Example (VLR)

E15/E35 Return Codes and EXITCK

DFSORT's interpretation of E15 and E35 return codes depends upon whether EXITCK=STRONG or EXITCK=WEAK is in effect. See the discussion of the EXITCK option in "OPTION Control Statement" on page 134 for more information. The following tables show the exact meaning of each E15 and E35 return code with EXITCK=STRONG and EXITCK=WEAK in all possible situations.

Notes:

1. You can determine whether EXITCK=STRONG or EXITCK=WEAK is in effect from message ICE132I.
2. Use of EXITCK=WEAK can make it difficult to detect errors in the logic of your E15 and E35 user exit routines.
3. 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 41. 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 42. E15 With a SORTIN Data Set Before End of Input

E15 Return Code	Meaning with EXITCK=STRONG or 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

E15/E35 Return Codes and EXITCK

Table 43. E15 With a SORTIN Data Set After End of Input

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 44. E35 With a SORTOUT or OUTFIL Data Set Before End of Input

E35 Return Code	Meaning with EXITCK=STRONG or 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 45. E35 Without a SORTOUT or OUTFIL 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 46. E35 With a SORTOUT or OUTFIL 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 47. E35 without a SORTOUT or OUTFIL 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 5. Invoking DFSORT from a Program

Invoking DFSORT Dynamically 331
What Are System Macro Instructions? 331
Using System Macro Instructions 331
Using JCL DD Statements 332
Overriding DFSORT Control Statements from Programs 332
Invoking DFSORT With the 24-Bit Parameter List 333
Providing Program Control Statements 333
CONTROL Statement Images Example 333
Format of the 24-Bit Parameter List 334
Invoking DFSORT With The Extended Parameter List 339
Providing Program Control Statements 339
Format of the Extended Parameter List 340
Writing the Macro Instruction 343
Parameter List Examples 343
Restrictions for Dynamic Invocation 347
Merge Restriction 347
Copy Restrictions 347

Invoking DFSORT Dynamically

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 to initiate DFSORT from within your assembler program using a system macro instruction instead of 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 347.

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 placed in the macro library when your control program 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 *Application Development Guide* and *Application Development Macro Reference*

Using System Macro Instructions

To initiate DFSORT processing with a system macro instruction, you must:

- Write the required job control language (JCL) DD statements.

© Copyright IBM Corp. 1973, 2000

Using System Macro Instructions

- 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 3. Using DFSORT Program Control Statements” on page 77 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; therefore, it should be used for new DFSORT applications.

  Note: DFSORT can also be called using the system’s EXEC PARM parameter list, provided that the rules for passing it are followed (for example, the parameter list must reside below 16MB virtual). DFSORT interprets a call using the EXEC PARM parameter list as a direct invocation rather than a program invocation.
- Prepare the macro instruction specifying one of the following as the entry point name: ICEMAN, SORT, IERRCO00, or IGHRCO00.

Note: The save area passed to DFSORT must begin on a fullword boundary.

In addition, the following rule applies:

- If you are invoking DFSORT repeatedly (for example, from an E15 or E35 user exit), you must always wait for the last invoked sort to end before you can give control back to any of your user exits in an earlier invoked sort.

Using 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.

Overriding DFSORT Control Statements from Programs

You can override the control statements generated or passed by a program (for example, a COBOL SORT verb or PLISRTx routine) with DFSORT’s DFSPARM data set.

For example, you could use the following to override the SORT statement generated by a COBOL SORT verb in order to use DFSORT’s Year 2000 features:

```
//DFSPARM  DD    *
  OPTION Y2PAST=1956        * set fixed CW of 1956-2055
  SORT FIELDS=(11,6,Y2T,A,  * sort C'yymmdd' using CW
               31,10,CH,A)  * sort other key
/*
```

You can also use DFSPARM to provide different DFSORT control statements for multiple invocations of DFSORT from a program. However, the control statements must be located in temporary or permanent data sets and FREE=CLOSE must be used. Here’s an example of using DFSPARM to override the control statements for a COBOL program with three SORT verbs:

```
//DFSPARM DD DSN=DP1,DISP=SHR,FREE=CLOSE
//DFSPARM DD DSN=DP2,DISP=SHR,FREE=CLOSE
//DFSPARM DD DSN=DP3,DISP=SHR,FREE=CLOSE
```

DP1, DP2, and DP3 can contain any DFSORT control statements you like. The statements in DP1 will be used for the first call to DFSORT, the statements in DP2 for the second and the statements in DP3 for the third.

For complete details on DFSPARM, see “DFSPARM DD Statement” on page 74.

Invoking DFSORT With the 24-Bit Parameter List

Providing Program Control Statements

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 run-time. 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 as follows:

- At least two control statements must be specified—generally SORT or MERGE, and RECORD. If more than 15 control statements are specified, only the first 15 control statements are accepted and all others are ignored. Control statements can also be specified in SORTCNTL or DFSPARM.
- The MODS statement is required when user exits other than E15, E32, and E35 are to be used. It is also required when the E15 or E35 routine addresses are not passed by the parameter list.
- The following control statements can be passed using the 24-bit parameter list: SORT or MERGE, RECORD, ALTSEQ, DEBUG, MODS, SUM, INREC, OUTREC, INCLUDE or OMIT, and OUTFIL.
- At least one blank must follow the operation definer (SORT, for example). A control statement can start and end with one or more blanks; however, 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 77, except:
  - Labels are not allowed although 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 B. Specification/Override of DFSORT Options” on page 549.

CONTROL Statement Images 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.

```
INCLBEG   DC   C' INCLUDE COND=(5,3,CH,NE,C'J82'')'
INCLEND   DC   C' '
```

Note: Assembler requires two single apostrophes to represent one single apostrophe.

Format of the 24-Bit Parameter List

Figure 33 on page 335 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 B. Specification/Override of DFSORT Options” on page 549.

```
Register 1
|
|
|
```

Notes to Figure 33 on page 335:

1. Required entry. Must appear in the relative position shown. The offset shown is the actual offset of this entry.

2. Optional entry. Can appear anywhere after the required entries. The displayed offset is for identification purposes only—the actual offset of this entry can vary. Optional entries must be consecutive but can appear in any order.

3. A specific control statement. Shown for illustrative purposes only. SORT or MERGE, RECORD, ALTSEQ, DEBUG, MODS, SUM, INREC, OUTREC, INCLUDE or OMIT, and OUTFIL can be passed using any of the following hex entry codes: X’00’ (see Note 1), X’02’, X’05’ through X’0B’, X’10’, X’16’, X’18’ and X’20’ through X’29’.

4. Required entry if the MERGE statement is present and input is supplied through an E32 user exit. This entry is not required if the FILES option of the MERGE statement is specified.

5. Required entry. Contains the ending address for a control statement and must immediately follow the entry containing the starting address for that same control statement.

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 in bytes of the parameter list. This two byte field is not included when counting the number of bytes occupied by the list.

The total length of the required entries is 24 (X'0018'). All optional entries are four bytes long except those referring to control statement images which are eight bytes each.

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. 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'.

34-37 Main storage value. 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 B. Specification/Override of DFSORT Options” on page 549. For an explanation of this value, see the discussion of the MAINSIZE parameter in “OPTION Control Statement” on page 134.

38-41 A reserved main storage value. 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 134.

You can use this option to temporarily override the RESINV installation option. For full override and applicability details, see “Appendix B. Specification/Override of DFSORT Options” on page 549.

42-45 Message ddname. The first byte must contain X’03’. The next three bytes contain the address of an 8-byte DD statement name for the message data set, padded with blanks on the right if necessary. The name can be any valid DD statement name, but must be unique.

You can use this option to temporarily override the MSGDDN installation option. For full override details, see “Appendix B. Specification/Override of DFSORT Options” on page 549. For details on the use of the message data set, see *Messages, Codes and Diagnosis.*

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 user 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 B. Specification/Override of DFSORT Options” on page 549.

50-53 The starting address of the DEBUG statement image. Must be in the last three bytes. The first byte must contain X'05'.

54-57 The ending address of the DEBUG statement image. Must be in the last three bytes. The first byte must contain X'00'.

58-61 The starting address of the ALTSEQ statement image. Must be in the last three bytes. The first byte must contain X'06'.

62-65 The ending address of the ALTSEQ 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 B. Specification/Override of DFSORT Options” on page 549.

70-73 User exit address constant. These 4 bytes are passed to E15 (at offset 4 in the E15 parameter list), to E32 (at offset 8 in the E32 parameter list) or to E35 (at offset 8 in the E35 parameter list) after DFSORT replaces the X'F7' with X'00'.

Note: The user exit address constant must not be used for a Conventional merge or tape work data set sort application.

74-77 X'FD' in the first byte (the VLSHRT option) specifies that DFSORT is to continue processing if it finds a variable-length input record too short to contain all specified control fields or compare fields. For full details of this option, see the discussion of the VLSHRT parameter in "OPTION Control Statement" on page 134. You can use this option to temporarily override the VLSHRT installation option. For full override and applicability details, see "Appendix B. Specification/Override of DFSORT Options" on page 549.

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 E. DFSORT Abend Processing" on page 595

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 *Messages, Codes and Diagnosis.* For full override and applicability details, see "Appendix B. Specification/Override of DFSORT Options" on page 549.

For compatibility reasons, the forms (NO, (AB, (AP, (AC, (CB, (CC, (CP, (PC, (SC, and (SP are also accepted.

The following list shows the equivalent specifications for these aliases:

Table 48. Aliases for Message Option

Option	MSGPRT	MSGCON
(NO	NONE	NONE
(AB	ALL	ALL
(AP	ALL	CRITICAL
(AC	NONE	ALL
(CB	CRITICAL	CRITICAL
(CC	NONE	CRITICAL
(CP	CRITICAL	CRITICAL
(PC	ALL	ALL
(SC	ALL	CRITICAL
(SP	CRITICAL	ALL

86-89 Four characters, which replace “SORT” in the following ddnames: SORTIN, SORTOUT, SORTINn, SORTINnn, SORTOFd, SORTOFdd, SORTWKd, SORTWKdd, 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 @) characters. 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.

For example, if you use ABC# as replacement characters, DFSORT uses statements ABC#IN, ABC#CNTL, ABC#WKdd, and ABC#OUT instead of SORTIN, SORTCNTL, SORTWKdd, and SORTOUT.

Note: This parameter is equivalent to the SORTDD=cccc run-time option.

90-93 The starting address of the SUM statement image. Must be in the last three bytes. The first byte must contain X'07'.

94-97 The ending address of the SUM statement image. Must be in the last three bytes. The first byte must contain X'00'.

98-101
The starting address of the INCLUDE or OMIT statement image. Must be in the last three bytes. The first byte must contain X'08'.

102-105
The ending address of the INCLUDE or OMIT statement image. Must be in the last three bytes. The first byte must contain X'00'.

106-109
The starting address of the OUTREC statement image. Must be in the last three bytes. The first byte must contain X'09'.

110-112
The ending address of the OUTREC statement image. Must be in the last three bytes. The first byte must contain X'00'.

114-116
The starting address of the INREC statement image. Must be in the last three bytes. The first byte must contain X'0A'.

118-121
The ending address of the INREC statement image. Must be in the last three bytes. The first byte must contain X'00'.

122-125
The starting address of the OUTFIL statement image. Must be in the last three bytes. The first byte must contain X'0B'.

126-129
The ending address of the OUTFIL statement image. Must be in the last three bytes. The first byte must contain X'00'.

Invoking DFSORT With The Extended Parameter List

Providing Program Control Statements

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 run-time.

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 user 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 77 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 B. Specification/Override of DFSORT Options” on page 549.

Format of the Extended Parameter List

Figure 34 on page 341 shows the format of the extended parameter list and the pointer, which you must pass to DFSORT, containing its address.

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 B. Specification/Override of DFSORT Options” on page 549.

```
     Register 1
                    |
(Hex)  (Dec)      Bit 0
                    |
                    ↓
```

(Hex)	(Dec)	Bit 0	
0	0	0	Address of control statement area (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	Address of user exit 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	4-character call identifier (zeros if none)	
24	36	X'FFFFFFFF'	

Figure 34. 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.

Refer to the previous section for the format of the control statement area. Note that the area must start with a two-byte length field.

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 user exit 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:
0 = Enter the user exit with 24-bit addressing in effect (AMODE 24).
1 = Enter the user exit with 31-bit addressing in effect (AMODE 31).

Note: If the Blockset or Peerage/Vale technique is not selected, the user exit is always entered with 24-bit addressing in effect (AMODE 24).

8-11 Optional. The address of the E35 user exit 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:
0 = Enter the user exit with 24-bit addressing in effect (AMODE 24).
1 = Enter the user exit with 31-bit addressing in effect (AMODE 31).

Note: If the Blockset or Peerage/Vale technique is not selected, the user exit is always entered with 24-bit addressing in effect (AMODE 24).

12-15 Optional. This field will be passed to the E15, E32 or E35 user exit routines.

Note: The user exit address constant must not be used for a Conventional merge or tape work data set sort application.

16-19 Optional. The address of a 256-byte alternate collating sequence table supplied instead of an ALTSEQ statement, if any; otherwise, all zeros. You can use this option to override any alternate collating sequence table specified at installation. For full override and applicability details, see "Appendix B. Specification/Override of DFSORT Options" on page 549.

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 software diagnostic work area (SDWA) into this area before returning to your ESTAE recovery routine.

24-27 Optional. The address of the E18 user exit 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:
0 = Enter the user exit with 24-bit addressing in effect (AMODE 24).
1 = Enter the user exit with 31-bit addressing in effect (AMODE 31).

Note: If the Blockset or Peerage/Vale technique is not selected, the user exit is always entered with 24-bit addressing in effect (AMODE 24).

28-31 Optional. The address of the E39 user exit 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.

f (bit 0) has the following meaning:
0 = Enter the user exit with 24-bit addressing in effect (AMODE 24).
1 = Enter the user exit with 31-bit addressing in effect (AMODE 31).

Note: If the Blockset or Peerage/Vale technique is not selected, the user exit is always entered with 24-bit addressing if effect (AMODE 24).

32-35 Optional. 4 characters to be used as an identifier for this call to DFSORT. This field can be used to uniquely identify each call to DFSORT from a program that calls DFSORT more than once. DFSORT prints message ICE200I to display the field identifier exactly as you specify it; the field is not checked for valid characters.

If the field identifier is specified, it must appear in the indicated position. If the identifier field contains zeros (X'00000000'), or X'FFFFFFFF' is used to end the parameter list before or at the field identifier, DFSORT does not print message ICE200I.

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 user exit routine address in the parameter list, DFSORT ignores the SORTIN data set; your E15 routine must pass all input records to DFSORT. The same applies to a merge if you specify an E32 routine 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”).

DFSORT ignores the SORTOUT data set if you provide an E35 routine address in the parameter list. Unless you use OUTFIL processing, 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 is done, 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 B. Specification/Override of DFSORT Options” on page 549.

If you ATTACH more than one DFSORT application from the same program, you must wait for each to complete before attaching the next unless DFSORT and your user exits are installed re-entrant.

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.

Parameter List Examples

24-Bit Parameter List Example 1

Figure 35 on page 344 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 35. Specifying the Main Storage Option (24-Bit Parameter List)

24-Bit Parameter List Example 2

Figure 36 shows the format of the 24-bit parameter list that you would use for a merge application when you want to supply input through an E32 routine 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 MERGE statement	
A	10	X'00'	Starting address of RECORD statement	
E	14	X'00'	Ending address of RECORD statement	
12	18	X'00'	Address of E32 routine	
16	22	X'00'	Zeros (no E35 routine provided)	
1A	26	X'04'	Number of input files	
1E	30	X'FE'	(Zeros—no work area address provided)	

Figure 36. Specifying E32 and ESTAE Routine (24-Bit Parameter List)

24-Bit Parameter List Example 3

Figure 37 on page 345 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).

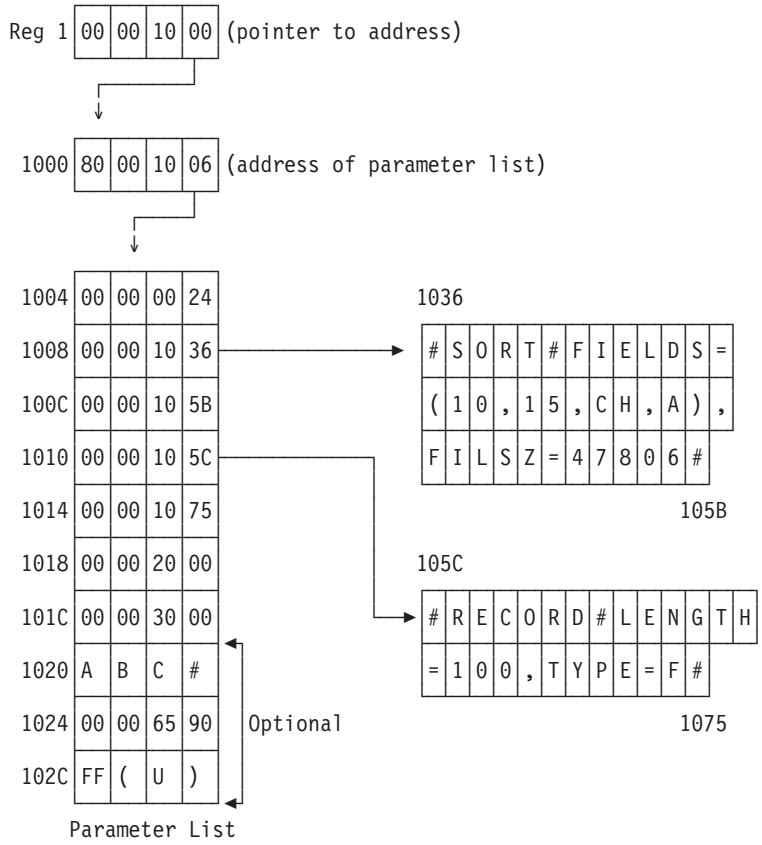


Figure 37. The 24-Bit Parameter List in Main Storage

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 user exit (location 2000) and E35 user 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.

24-Bit Parameter List Example 4

The example in Figure 38 on page 346 shows, in assembler language, how to code the parameters and statement images needed for the 24-bit parameter list in Figure 37. It also shows 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=E4780'          (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 E15 user exit)
        .
        .
        USING *,15
MOD2    (routine for E35 user exit)
```

Figure 38. Coding a 24-Bit Parameter List

Extended Parameter List Example 1

The example in Figure 39 on page 347 shows, in assembler language, how to use an extended parameter list to code parameters and statement images and how to 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 39. 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 or later, 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 or later FASTSRT compiler 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.

Restrictions for Dynamic Invocation

See “COBOL Requirements for Copy Processing” on page 310 for user exit requirements.

Chapter 6. Using ICETOOL

Overview . . . 350
ICETOOL/DFSORT Relationship . . . 351
ICETOOL JCL Summary . . . 351
ICETOOL Operator Summary . . . 352
Complete ICETOOL Examples. . . . 353
Using Symbols . . . 353
Invoking ICETOOL . . . 354
Putting ICETOOL to Use . . . 354
Obtaining Various Statistics . . . 354
Creating Multiple Versions/Combinations of Data Sets . . . 355
Job Control Language for ICETOOL . . . 357
JCL Restrictions . . . 359
ICETOOL Statements . . . 359
General Coding Rules . . . 359
COPY Operator . . . 360
COPY Examples . . . 362
Example 1 . . . 362
Example 2 . . . 363
Example 3 . . . 364
COUNT Operator . . . 364
COUNT Example . . . 365
DEFAULTS Operator . . . 365
DEFAULTS Example . . . 367
DISPLAY Operator . . . 369
Simple Report. . . . 370
Tailored Report . . . 370
Sectioned Report . . . 371
DISPLAY Examples. . . . 385
Example 1 . . . 385
Example 2 . . . 385
Example 3 . . . 386
Example 4 . . . 387
Example 5 . . . 388
Example 6 . . . 389
Example 7 . . . 390
Example 8 . . . 392
Example 9 . . . 394
Example 10 . . . 396
MODE Operator . . . 397
MODE Example . . . 398
OCCUR Operator . . . 398
Simple Report. . . . 399
Tailored Report . . . 400
OCCUR Examples . . . 404
Example 1 . . . 404
Example 2 . . . 405
Example 3 . . . 405
Example 4 . . . 406
RANGE Operator . . . 406
RANGE Example . . . 408
SELECT Operator . . . 409
SELECT Examples . . . 412
Example 1 . . . 412

© Copyright IBM Corp. 1973, 2000

Example 2 413
Example 3 413
Example 4 413
Example 5 414
SORT Operator 414
SORT Examples 416
Example 1 416
Example 2 417
Example 3 418
STATS Operator 418
STATS Example 420
UNIQUE Operator 420
UNIQUE Example 421
VERIFY Operator 422
VERIFY Example 423
Calling ICETOOL from a Program 424
TOOLIN Interface 424
Parameter List Interface 424
Explanation of Fields 425
Parameter List Interface Example 427
ICETOOL Notes and Restrictions. 430
ICETOOL Return Codes 431

Overview

This chapter describes ICETOOL, a multi-purpose DFSORT utility. ICETOOL uses the capabilities of DFSORT to perform multiple operations on one or more data sets in a single job step. These operations include the following:

- Creating multiple copies of sorted, edited, or unedited input data sets
- Creating output data sets containing subsets of input data sets based on various criteria for character and numeric field values or the number of times unique values occur
- Creating output data sets containing different field arrangements of input data sets
- Creating list data sets showing character and numeric fields in a variety of simple, tailored, and sectioned report formats, allowing control of title, date, time, page numbers, headings, lines per page, field formats, and total, maximum, minimum and average values for the columns of numeric data
- Printing messages that give statistical information for selected numeric fields such as minimum, maximum, average, total, count of values, and count of unique values
- Printing messages that identify invalid decimal values
- Creating a list data set showing the DFSORT installation defaults in use
- Creating list data sets showing unique values for selected character and numeric fields and the number of times each occurs, in a variety of simple and tailored report formats
- Creating list and output data sets for records with: duplicate values, non-duplicate values, or values that occur n times, less than n times or more than n times
- Using three different modes (stop, continue, and scan) to control error checking and actions after error detection for groups of operators.

ICETOOL/DFSORT Relationship

ICETOOL is a batch front-end utility that uses the capabilities of DFSORT to perform the operations you request.

ICETOOL is comprised of twelve operators that perform sort, copy, statistical, and report operations. Most of the operations performed by ICETOOL require only simple JCL and operator statements. Some ICETOOL operations require or allow you to specify complete DFSORT control statements (such as SORT, INCLUDE, and OUTFIL) to take full advantage of DFSORT's capabilities.

ICETOOL automatically calls DFSORT with the particular DFSORT control statements and options required for each operation (such as DYNALLOC for sorting).

ICETOOL also produces messages and return codes describing the results of each operation and any errors detected. Although you generally do not need to look at the DFSORT messages produced as a result of an ICETOOL run, they are available in a separate data set if you need them.

ICETOOL can be called directly or from a program. ICETOOL allows operator statements (and comments) to be supplied in a data set or in a parameter list passed by a calling program. For each operator supplied in the parameter list, ICETOOL puts information in the parameter list pertaining to that operation, thus allowing the calling program to use the information derived by ICETOOL.

ICETOOL JCL Summary

The JCL statements used with ICETOOL are summarized below. See "Job Control Language for ICETOOL" on page 357 for more detailed information. See also "JCL Restrictions" on page 359 and "ICETOOL Notes and Restrictions" on page 430.

//JOBLIB DD
Defines your program link library if it is not already known to the system.

//STEPLIB DD
Same as //JOBLIB DD

//TOOLMSG DD
Defines the ICETOOL message data set for all operations.

//DFSMSG DD
Defines the DFSORT message data set for all operations.

//SYMNAMES DD
Defines the SYMNAMES data set containing statements to be used for symbol processing.

//SYMNOUT DD
Defines the data set in which SYMNAMES statements and the symbol table are to be listed.

//TOOLIN DD
Contains ICETOOL control statements.

//indd DD
Defines an input data set for a COPY, COUNT, DISPLAY, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, or VERIFY operation.

//outdd DD
Defines an output data set for a COPY, SELECT, or SORT operation.

//savedd DD
Defines an output data set for a SELECT operation.

//listdd DD
Defines a list data set for a DEFAULTS, DISPLAY, or OCCUR operation.

//xxxxCNTL DD
Contains DFSORT control statements for a COPY, COUNT, or SORT operation.

ICETOOL Operator Summary

ICETOOL has twelve operators which are used to perform a variety of functions. The functions of these operators are summarized below. See “ICETOOL Statements” on page 359 for more detailed information. Additionally, information pertaining to each operator is provided to calling programs which supply statements to ICETOOL using a parameter list. See “Parameter List Interface” on page 424 for details.

COPY
Copies a data set to one or more output data sets.

COUNT
Prints a message containing the count of records in a data set.

DEFAULTS
Prints the DFSORT installation defaults in a separate list data set.

DISPLAY
Prints the values or characters of specified numeric or character fields in a separate list data set. Simple, tailored, or sectioned reports can be produced.

MODE
Three modes are available which can be set or reset for groups of operators:

- STOP mode (the default) stops subsequent operations if an error is detected
- CONTINUE mode continues with subsequent operations if an error is detected
- SCAN mode allows ICETOOL statement checking without actually performing any operations.

OCCUR
Prints each unique value for specified numeric or character fields and how many times it occurs in a separate list data set. Simple or tailored reports can be produced. The values printed can be limited to those for which the value count meets specified criteria (for example, only duplicate values or only non-duplicate values).

RANGE
Prints a message containing the count of values in a specified range for a specified numeric field in a data set.

SELECT
Selects records from a data set for inclusion in an output data set based on meeting criteria for the number of times specified numeric or character field values occur (for example, only duplicate values or only non-duplicate values). Records that are not selected can be saved in a separate output data set.

SORT
Sorts a data set to one or more output data sets.

STATS
Prints messages containing the minimum, maximum, average, and total for specified numeric fields in a data set.

UNIQUE
Prints a message containing the count of unique values for a specified numeric or character field.

VERIFY
Examines specified decimal fields in a data set and prints a message identifying each invalid value found for each field.

Complete ICETOOL Examples

"ICETOOL Example" on page 535 contains a complete ICETOOL sample job with all required JCL and control statements. The example below shows the JCL and control statements for a simple ICETOOL job.

Table 49. Simple ICETOOL Job

```
//EXAMP JOB A402,PROGRAMMER
//RUNIT EXEC PGM=ICETOOL,REGION=1024K
//TOOLMSG DD SYSOUT=A
//DFSMSG DD SYSOUT=A
//TOOLIN DD *
* Show installation (ICEMAC) defaults
  DEFAULTS LIST(SHOWDEF)
* Create three copies of a data set
  COPY FROM(IN1) TO(OUT1,OUT2,OUT3)
* Print a report
  DISPLAY FROM(IN2) LIST(REPORT) DATE TITLE('Monthly Report') PAGE -
    HEADER('Location') ON(1,25,CH) -
    HEADER('Revenue') ON(23,10,FS) -
    HEADER('Profit') ON(45,10,FS) -
    TOTAL('Totals') AVERAGE('Averages') BLANK
* Select all records with duplicate (non-unique) keys
  SELECT FROM(IN2) TO(DUPKEYS) ON(1,25,CH) ALLDUPS -
* Save all records with non-duplicate (unique) keys
  DISCARD (UNQKEYS)
/*
//SHOWDEF DD SYSOUT=A
//IN1 DD DSN=FLY.INPUT1,DISP=SHR
//IN2 DD DSN=FLY.INPUT2,DISP=SHR
//OUT1 DD DSN=FLY.NEW,DISP=OLD
//OUT2 DD DSN=FLY.BU1,DISP=OLD
//OUT3 DD DSN=FLY.BU2,DISP=OLD
//DUPKEYS DD DSN=FLY.DUPS,DISP=OLD
//UNQKEYS DD DSN=FLY.UNQS,DISP=OLD
//REPORT DD SYSOUT=A
```

Using Symbols

You can define and use a symbol for any field or constant in the following ICETOOL operators: DISPLAY, OCCUR, RANGE, SELECT, STATS, UNIQUE and VERIFY. You can also use symbols in the DFSORT control statements you specify for an ICETOOL run. This makes it easy to create and reuse collections of symbols (that is, mappings) representing information associated with various record layouts. See "Chapter 7. Using Symbols for Fields and Constants" on page 433 for complete details.

Invoking ICETOOL

ICETOOL can be invoked in the following three ways:

- Directly (that is, not from a program) using the TOOLIN Interface
- From a program using the TOOLIN Interface
- From a program using the Parameter List Interface.

With the TOOLIN Interface, you supply ICETOOL statements in a data set defined by the TOOLIN DD statement. ICETOOL prints messages in the data set defined by the TOOLMSG DD statement.

With the Parameter List Interface, your program supplies ICETOOL statements in a parameter list. ICETOOL prints messages in the data set defined by the TOOLMSG DD statement and also puts information in the parameter list for use by your program.

Putting ICETOOL to Use

By using various combinations of the twelve ICETOOL operators, you can easily create applications that perform many complex tasks. The two small samples that follow show some things you can do with ICETOOL.

Obtaining Various Statistics

Table 50. Obtaining Various Statistics

```
MODE STOP
VERIFY FROM(DATA1) ON(22,7,PD)
DISPLAY FROM(DATA1) LIST(SALARIES) -
  TITLE('Employee Salaries') DATE TIME -
  HEADER('Employee Name') HEADER('Salary') -
  ON(1,20,CH)             ON(22,7,PD)      BLANK -
  AVERAGE('Average Salary')
STATS FROM(DATA1) ON(22,7,PD)
RANGE FROM(DATA1) ON(22,7,PD) LOWER(20000)
RANGE FROM(DATA1) ON(22,7,PD) HIGHER(19999) LOWER(40000)
RANGE FROM(DATA1) ON(22,7,PD) HIGHER(40000)
OCCUR FROM(DATA1) LIST(SALARIES) -
  TITLE('Employees Receiving Each Salary') DATE TIME -
  HEADER('Salary') HEADER('Employee Count') -
  ON(22,7,PD)      ON(VALCNT)               BLANK
```

Assume that you specify DD statements with the following ddnames for the indicated data sets:

DATA1
: A data set containing the name, salary, department, location and so on, of each of your employees. The name field is in positions 1 through 20 in character format and the salary field is in positions 22 through 28 in packed decimal format.

SALARIES
: A SYSOUT data set.

You can use the ICETOOL operators in Table 50 to do the following:

MODE STOP
: If an error is found while processing one of the operators, subsequent operators are not processed (that is, each operator is dependent on the success of the previous operator).

VERIFY
Prints error messages in the TOOLMSG data set identifying any invalid values in the packed decimal salary field.

DISPLAY
Prints a report with each employee's name and salary and the average for all employee salaries in the SALARIES list data set.

STATS
Prints messages in the TOOLMSG data set showing the minimum, maximum, average, and total of the individual salaries.

RANGE
The three RANGE operators print messages in the TOOLMSG data set showing the number of salaries below $20,000, from $20,000 to $39,999, and above $40,000.

OCCUR
Prints a report with each unique salary and the number of employees who receive it in the SALARIES list data set.

Creating Multiple Versions/Combinations of Data Sets

Table 51. Creating Multiple Versions/Combinations of Data Sets

```
* GROUP 1
  MODE CONTINUE
  COPY FROM(DATA1) TO(DATA2)
  COPY FROM(MSTR1) TO(MSTR2)
  SELECT FROM(DATA1) TO(SMALLDPT) ON(30,4,CH) LOWER(10)
  UNIQUE FROM(MSTR1) ON(30,4,CH)
* GROUP 2
  MODE STOP
  COPY FROM(DATA1) TO(TEMP1) USING(NEW1)
  COPY FROM(DATA1) TO(TEMP2) USING(NEW2)
  COPY FROM(DATA1) TO(TEMP3) USING(NEW3)
  SORT FROM(CONCAT) TO(FINALD,FINALP) USING(FINL)
```

Assume that you specify DD statements with the following ddnames for the indicated data sets:

DATA1
A data set containing the name, salary, department, location, and so on, of each of your employees. The department field is in positions 30 through 33 in character format.

MSTR1
Master data set containing only the name and department of each of your employees. The department field is in positions 30 through 33 in character format.

DATA2, MSTR2, and SMALLDPT
Permanent data sets.

NEW1CNTL
A data set containing DFSORT control statements to INCLUDE employees in department X100 and change the records to match the format of MSTR1.

NEW2CNTL
Same as NEW1CNTL but for department X200.

NEW3CNTL
Same as NEW1CNTL but for department X300.

TEMP1, TEMP2, and TEMP3
Temporary data sets.

FINLCNTL
A data set containing a DFSORT control statement to sort by department and employee name.

CONCAT
A concatenation of the TEMP1, TEMP2, TEMP3, and MSTR1 data sets.

FINALD
A permanent data set.

FINALP
A SYSOUT data set.

You can use the ICETOOL operators in Table 51 to do the following:

MODE CONTINUE
If an error is found while processing any of the group 1 operators, subsequent group 1 operators are still processed; that is, group 1 operators are not dependent on the success of the previous group 1 operators.

COPY The two copy operators create backup copies of DATA1 and MSTR1.

SELECT
Creates a permanent output data set containing the name, salary, department, location, and so on, of each employee in departments with less than 10 people.

UNIQUE
Prints a message in the TOOLMSG data set showing the number of unique departments.

MODE STOP
If an error is found while processing one of the group 2 operators, subsequent group 2 operators are not processed; that is, each group 2 operator is dependent on the success of previous group 2 operators.

COPY The three COPY operators create an output data set for the employees in each department containing only name and department. Note that the ddname requested by the USING(xxxx) operand is xxxxCNTL. For example, USING(NEW1) requests ddname NEW1CNTL.

SORT Sorts the three output data sets created by the COPY operators along with the master name/department data set and creates permanent and SYSOUT data sets containing the resulting sorted records.

You can combine both of these examples into a single ICETOOL job step.

Job Control Language for ICETOOL

An overview of the job control language (JCL) statements for ICETOOL is given below followed by discussions of each ICETOOL DD statement and the use of reserved DD statements and ddnames.

Table 52. JCL Statements for ICETOOL

```
//EXAMPL JOB ...
//* ICETOOL CAN BE CALLED DIRECTLY OR FROM A PROGRAM
//STEP EXEC PGM=ICETOOL (or PGM=program_name)
//* THE FOLLOWING DD STATEMENTS ARE ALWAYS REQUIRED
//TOOLMSG DD SYSOUT=A (or DSN=...)
//DFSMSG  DD SYSOUT=A
//* THE FOLLOWING DD STATEMENTS ARE USED FOR SYMBOL PROCESSING
//* SYMNAMES DD ...
//* SYMNOUT DD SYSOUT=A (OR DSN=...)
//* THE TOOLIN DD STATEMENT IS ONLY REQUIRED IF THE TOOLIN INTERFACE
//* IS USED.
//TOOLIN DD *
   ICETOOL statements
/*
//* THE FOLLOWING DD STATEMENTS ARE ONLY REQUIRED IF SPECIFIED IN
//* ICETOOL STATEMENTS.
//indd DD ...
     .
     .
     .
//outdd DD ...
   .
   .
   .
//listdd DD SYSOUT=A (or DSN=...)
   .
   .
   .
//xxxxCNTL DD *
   DFSORT control statements
/*
   .
   .
   .
```

TOOLMSG DD Statement

Defines the ICETOOL message data set for all operations. ICETOOL messages and statements appear in this data set. ICETOOL uses RECFM=FBA, LRECL=121 and the specified BLKSIZE for the TOOLMSG data set. If the BLKSIZE you specify is not a multiple of 121, ICETOOL uses BLKSIZE=121. If you do not specify the BLKSIZE, ICETOOL selects the block size as directed by the SDBMSG installation option (see *Installation and Customization*).

The TOOLMSG DD statement *must* be present.

DFSMSG DD Statement

Defines the DFSORT message data set for all operations. The DFSORT messages and control statements from all ICETOOL calls to DFSORT appear in this data set. Refer to the discussion of SYSOUT in “System DD Statements” on page 62 for details.

The DFSMSG DD statement *must* be present.

Note: A SYSOUT data set should be used for DFSMSG. If you define DFSMSG as a temporary or permanent data set, you will only see the DFSORT messages from the last call to DFSORT unless you allocate a new data set using a disposition of MOD.

//SYMNAMES DD
Defines the SYMNAMES data set containing statements to be used for symbol processing. See “Chapter 7. Using Symbols for Fields and Constants” on page 433 for complete details.

//SYMNOUT DD
Defines the data set in which SYMNAMES statements and the symbol table are to be listed. See “Chapter 7. Using Symbols for Fields and Constants” on page 433 for complete details.

TOOLIN DD statement
Defines the ICETOOL statement data set which must have the following attributes: RECFM=F or RECFM=FB and LRECL=80.

If the TOOLIN Interface is used, the TOOLIN DD statement must be present. If the Parameter List Interface is used, the TOOLIN DD statement is not required and is ignored if present.

indd DD Statement
Defines the input data set for an operation. Refer to“SORTIN DD Statement” on page 66 for details. ICETOOL imposes the additional restriction that the LRECL of this data must be at least 4.

An indd DD statement must be present for each unique indd name specified in each FROM operand.

outdd DD Statement
Defines an output data set for a COPY, SELECT, or SORT operation. Refer to “SORTOUT and OUTFIL DD Statements” on page 71 for details.

An outdd DD statement must be present for each unique outdd name specified in each TO operand.

savedd DD Statement
Defines an output data set for a SELECT operation. Refer to“SORTOUT and OUTFIL DD Statements” on page 71 for details.

A savedd DD statement must be present for each unique savedd name specified in each DISCARD operand.

listdd DD Statement
Defines the list data set for a DEFAULTS, DISPLAY, or OCCUR operation. For each listdd data set, ICETOOL uses RECFM=FBA, LRECL=121 (for DEFAULTS) or the LRECL specified in the WIDTH operand or calculated as needed if WIDTH is not specified (DISPLAY and OCCUR), and the specified block size. If the BLKSIZE you specify is not a multiple of the LRECL, ICETOOL uses BLKSIZE=LRECL. If you do not specify BLKSIZE, ICETOOL selects the block size as directed by the SDBMSG installation option (see *Installation and Customization*).

A listdd DD statement must be present for each unique listdd name specified in each LIST operand.

xxxxCNTL DD Statement

Defines the DFSORT control statement data set for a SORT, COPY, or COUNT operation. Refer to “SORTCNTL DD Statement” on page 73 for more details.

An xxxxCNTL DD statement must be present for each unique xxxx specified in each USING operand.

JCL Restrictions

You should avoid using ddnames reserved for ICETOOL and DFSORT in ICETOOL operands (FROM, TO, LIST,DISCARD). In general, you should also avoid supplying DD statements with ddnames reserved for DFSORT when using ICETOOL because doing so can cause unpredictable results. Specifically:

- SORTCNTL should not be used as a ddname in ICETOOL operators nor should it be supplied as a DD statement. A xxxxCNTL DD statement should only be supplied when you specify a USING(xxxx) operand. xxxx must be four characters which are valid in a ddname of the form xxxxCNTL. xxxx must not be SYSx.
- SYSIN, SORTCNTL, SORTIN, SORTOUT, SORTINnn, and xxxxINnn (where xxxx is specified in a USING operand) should not be used as ddnames in ICETOOL operators nor supplied as DD statements.
- TOOLMSG, DFSMSG, SYMNAMES, SYMNOUT, TOOLIN, SYSUDUMP, and SYSABEND should not be used as ddnames in ICETOOL operators.
- SORTWKdd and xxxxWKdd (where xxxx is specified in a USING operand) should not be used as ddnames in ICETOOL operators. DD statements for these ddnames should only be supplied as work data sets to override dynamic allocation for ICETOOL operators OCCUR, SELECT, SORT, and UNIQUE, if appropriate.
- DFSPARM (or the ddname specified for ICEMAC option PARMDDN) should not be used as a ddname in ICETOOL operators. It should only be used as a DD statement to override DFSORT options for all operators, if appropriate. Refer to “DFSPARM DD Statement” on page 74 for details.
- xxxxOFdd (where xxxx is specified in a USING operand) is required as the ddname when an OUTFIL statement in the xxxxCNTL data set specifies FILES=dd. To avoid this requirement, use the FNAMES=ddname operand rather than the FILES=dd operand in OUTFIL statements, and include a DD statement for the specified ddname. See “OUTFIL Control Statements” on page 182 for details of the FNAMES operand.

ICETOOL Statements

Each operation must be described to ICETOOL using an operator statement. Additionally, ICETOOL allows comment statements and blank statements. An explanation of the general rules for coding ICETOOL statements is given below followed by a detailed discussion of each operator.

General Coding Rules

The general format for all ICETOOL operator statements is:

```
OPERATOR operand ... operand
```

where each operand consists of KEYWORD(parameter, parameter...) or just KEYWORD. Any number of operators can be specified and in any order.

The following rules apply for operator statements:

- The operator and operands must be in uppercase EBCDIC.

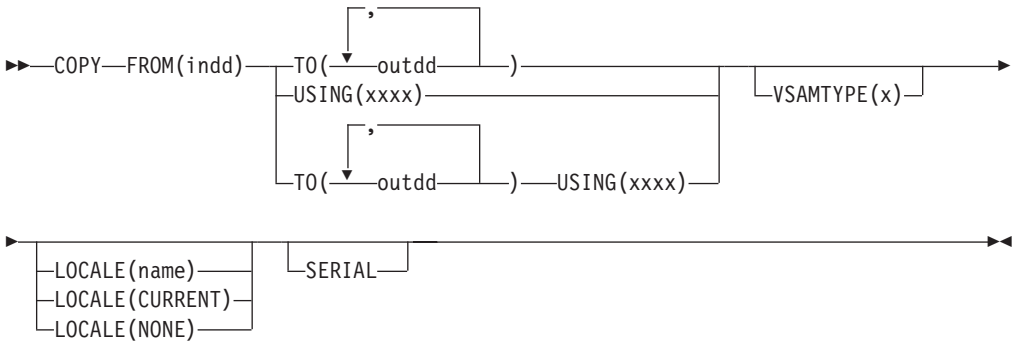
- The operator must be specified first.
- One blank is required between the operator and the first operand.
- One blank is required between operands.
- Any number of blanks can be specified before or after the operator or any operand, but blanks cannot be specified anywhere else except within quoted character strings.
- Parentheses must be used where shown. Commas or semicolons must be used where commas are shown.
- Operands can be in any order.
- Columns 1-72 are scanned; columns 73-80 are ignored.
- Continuation can be indicated by a dash (-) **after** the operator or after any operand. The next operand must then be specified on the next line. For example: Any characters specified after the dash are ignored. Each operand **must** be

```
SORT FROM(INDD) -
   USING(ABCD) -
   TO(OUTPUT1,OUTPUT2,OUTPUT3)
```

  completely specified on one line.

A statement with an asterisk (*) in column 1 is treated as a comment statement. It is printed with the other ICETOOL statements, but otherwise not processed. A statement with blanks in columns 1 through 72 is treated as a blank statement. It is ignored since ICETOOL prints blank lines where appropriate.

COPY Operator



Copies an input data set to one or more output data sets.

DFSORT is called to copy the indd data set to the outdd data sets; the DFSORT control statements in xxxxCNTL are used if USING(xxxx) is specified. You can use DFSORT control statements and options in the xxxxCNTL data set to copy a subset of the input records (INCLUDE or OMIT statement; SKIPREC and STOPAFT options; OUTFIL INCLUDE, OMIT, STARTREC, ENDREC, and SPLIT operands; user exit routines), reformat records for output (INREC and OUTREC statements, OUTFIL OUTREC operand, user exit routines), and so on.

If an INCLUDE or OMIT statement or an OUTFIL INCLUDE or OMIT operand is specified in the xxxxCNTL data set, the active locale's collating rules affect INCLUDE and OMIT processing, as explained in the "Cultural Environment Considerations" discussion in "INCLUDE Control Statement" on page 93.

The operands described below can be specified in any order.

FROM(indd)
Specifies the ddname of the input data set to be read by DFSORT for this operation. An indd DD statement must be present and must define an input data set that conforms to the rules for DFSORT's SORTIN data set.

Refer to "JCL Restrictions" on page 359 for more information regarding the selection of ddnames.

TO(outdd,...)
Specifies the ddnames of the output data sets to be written by DFSORT for this operation. From 1 to 10 outdd names can be specified. An outdd DD statement must be present for each outdd name specified. If a single outdd data set is specified, DFSORT is called once to copy the indd data set to the outdd data set, using SORTOUT processing; the outdd data set must conform to the rules for DFSORT's SORTOUT data set. If multiple outdd data sets are specified and SERIAL is not specified, DFSORT is called once to copy the indd data set to the outdd data sets, using OUTFIL processing; the outdd data sets must conform to the rules for DFSORT's OUTFIL data sets.

TO and USING can both be specified. If USING is not specified, TO must be specified. If TO is not specified, USING must be specified.

A ddname specified in the FROM operand must not also be specified in the TO operand.

Refer to "JCL Restrictions" on page 359 for more information regarding the selection of ddnames.

USING(xxxx)
Specifies the first 4 characters of the ddname for the control statement data set to be used by DFSORT for this operation. xxxx must be four characters which are valid in a ddname of the form xxxxCNTL. xxxx must not be SYSx.

If USING is specified, an xxxxCNTL DD statement must be present and the control statements in it must conform to the rules for DFSORT's SORTCNTL data set.

TO and USING can both be specified. If USING is not specified, TO must be specified. If TO is not specified, USING must be specified and the xxxxCNTL data set must contain either one or more OUTFIL statements or a MODS statement for an E35 routine that disposes of all records. Other statements are optional.

Refer to "JCL Restrictions" on page 359 for more information regarding the selection of ddnames.

VSAMTYPE(x)
Specifies the record type for a VSAM input data set. x must be either F for fixed-length record processing or V for variable-length record processing.

If VSAMTYPE(x) is specified, ICETOOL will pass a RECORD TYPE=x control statement to DFSORT. (If you specify a RECORD TYPE=x statement in the xxxxCNTL data set, it will override the one passed by ICETOOL.)

For complete information on record type processing for VSAM input, see "RECORD Control Statement" on page 260.

LOCALE(name)
Specifies that locale processing is to be used and designates the name of the locale to be made active during DFSORT processing. LOCALE(name) can be used to override the LOCALE installation option. For complete details on LOCALE(name), see the discussion of the LOCALE operand in “OPTION Control Statement” on page 134.

LOCALE(CURRENT)
Specifies that locale processing is to be used, and the current locale active when DFSORT is entered will remain the active locale during DFSORT processing. LOCALE(CURRENT) can be used to override the LOCALE installation option. For complete details on LOCALE(CURRENT), see the discussion of the LOCALE operand in “OPTION Control Statement” on page 134.

LOCALE(NONE)
Specifies that locale processing is not to be used. DFSORT will use the binary encoding of the code page defined for your data for collating and comparing. LOCALE(NONE) can be used to override the LOCALE installation option. For complete details on LOCALE(NONE), see the discussion of the LOCALE operand in “OPTION Control Statement” on page 134.

SERIAL
Specifies that OUTFIL processing is not to be used when multiple outdd data sets are specified. DFSORT is called multiple times and uses SORTOUT processing; the outdd data sets must conform to the rules for DFSORT’s SORTOUT data set. SERIAL is not recommended because the use of serial processing (that is, multiple calls to DFSORT) instead of OUTFIL processing can degrade performance and imposes certain restrictions as detailed below. SERIAL is ignored if a single outdd data set is specified.

DFSORT is called to copy the indd data set to the first outdd data set using the DFSORT control statements in the xxxxCNTL data set if USING(xxxx) is specified. If the first copy is successful, DFSORT is called as many times as necessary to copy the first outdd data set to the second and subsequent outdd data sets. Therefore, for maximum efficiency, use a DASD data set as the first in a list of outdd data sets on both DASD and tape. If more than one outdd data set is specified, DFSORT must be able to read the *first* outdd data set after it is written in order to copy it to the other outdd data sets. Do not use a SYSOUT or DUMMY data set as the first in a list of outdd data sets because:

- if the first data set is SYSOUT, DFSORT abends when it tries to copy the SYSOUT data set to the second outdd data set.
- if the first data set is DUMMY, DFSORT copies the empty DUMMY data set to the other outdd data sets, with the result that all outdd data sets are then empty.

COPY Examples

Although the COPY operators in the examples below could all be contained in a single ICETOOL job step, they are shown and discussed separately for clarity.

Example 1

```
* Method 1
COPY FROM(MASTER) TO(PRINT,TAPE,DASD)
```

```
* Method 2
COPY FROM(MASTER) TO(DASD,TAPE,PRINT) SERIAL
```

This example shows two different methods for creating multiple output data sets.

Method 1 requires one call to DFSORT, one pass over the input data set, and allows the output data sets to be specified in any order. The COPY operator copies all records from the MASTER data set to the PRINT (SYSOUT), TAPE, and DASD data sets, using OUTFIL processing.

Method 2 requires three calls to DFSORT, three passes over the input data set, and imposes the restriction that the SYSOUT data set must not be the first TO data set. The COPY operator copies all records from the MASTER data set to the DASD data set and then copies the resulting DASD data set to the TAPE and PRINT (SYSOUT) data sets. Since the first TO data set is processed three times (written, read, read), placing the DASD data set first is more efficient than placing the TAPE data set first. PRINT must not be the first in the TO list because a SYSOUT data set cannot be read.

Example 2

```
* Method 1
COPY FROM(IN) TO(DEPT1) USING(DPT1)
COPY FROM(IN) TO(DEPT2) USING(DPT2)
COPY FROM(IN) TO(DEPT3) USING(DPT3)
```

```
* Method 2
COPY FROM(IN) USING(ALL3)
```

This example shows two different methods for creating subsets of an input data set. Assume that:

- The DPT1CNTL data set contains:
  ```
  INCLUDE COND=(5,3,CH,EQ,C'D01')
  ```
- The DPT2CNTL data set contains:
  ```
  INCLUDE COND=(5,3,CH,EQ,C'D02')
  ```
- The DPT3CNTL data set contains:
  ```
  INCLUDE COND=(5,3,CH,EQ,C'D03')
  ```
- The ALL3CNTL data set contains:
  ```
  OUTFIL FNAMES=DEPT1,INCLUDE=(5,3,CH,EQ,C'D01')
  OUTFIL FNAMES=DEPT2,INCLUDE=(5,3,CH,EQ,C'D02')
  OUTFIL FNAMES=DEPT3,INCLUDE=(5,3,CH,EQ,C'D03')
  ```

Method 1 requires three calls to DFSORT and three passes over the input data set:

- The first COPY operator copies the records from the IN data set that contain D01 in positions 5-7 to the DEPT1 data set.
- The second COPY operator copies the records from the IN data set that contain D02 in positions 5-7 to the DEPT2 data set.
- The third COPY operator copies the records from the IN data set that contain D03 in positions 5-7 to the DEPT3 data set.

Method 2 accomplishes the same result as method 1, but because it uses OUTFIL statements instead of TO operands, requires only one call to DFSORT and one pass over the input data set.

Example 3

```
COPY FROM(VSAMIN) TO(VSAMOUT) VSAMTYPE(V)
```

The COPY operator copies all records from the VSAMIN data set to the VSAMOUT data set. The VSAM records are treated as variable-length.

COUNT Operator

```
►►──COUNT──FROM(indd)──┬──────────────┬──┬──────────────┬──┬──────────────────────┬──►◄
                       └─USING(xxxx)──┘  └─VSAMTYPE(x)──┘  ├─LOCALE(name)─────────┤
                                                           ├─LOCALE(CURRENT)──────┤
                                                           └─LOCALE(NONE)─────────┘
```

Prints a message containing the count of records in a data set.

DFSORT is called to copy the indd data set to ICETOOL's E35 user exit. The DFSORT control statements in xxxxCNTL are used if USING(xxxx) is specified. You can use a DFSORT INCLUDE or OMIT statement in the xxxxCNTL data set to count a subset of the input records.

If an INCLUDE or OMIT statement is specified in the xxxxCNTL data set, the active locale's collating rules affect INCLUDE and OMIT processing as explained in the "Cultural Environment Considerations" discussion in "INCLUDE Control Statement" on page 93.

ICETOOL prints a message containing the record count as determined by its E35 user exit.

You must not supply your own DFSORT MODS statement because it would override the MODS statement passed by ICETOOL for this operator.

Note: The record count is also printed for the DISPLAY, OCCUR, RANGE, SELECT, STATS, UNIQUE, and VERIFY operators.

The operands described below can be specified in any order.

FROM(indd)
See the discussion of this operand on the COPY statement in "COPY Operator" on page 360.

USING(xxxx)
Specifies the first 4 characters of the ddname for the control statement data set to be used by DFSORT for this operation. xxxx must be four characters which are valid in a ddname of the form xxxxCNTL. xxxx must not be SYSx.

If USING is specified, an xxxxCNTL DD statement must be present and the control statements in it:
1. Must conform to the rules for DFSORT's SORTCNTL data set.
2. Should generally be used only for an INCLUDE or OMIT statement or comments statements.

Refer to “JCL Restrictions” on page 359 for more information regarding the selection of ddnames.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

LOCALE(name)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

LOCALE(CURRENT)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

LOCALE(NONE)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

COUNT Example

For the following example, assume that the CTL1CNTL data set contains a DFSORT INCLUDE statement.

```
COUNT FROM(IN1)
COUNT FROM(IN2) USING(CTL1)
```

The first COUNT operator prints a message containing the count of records in the IN1 data set.

The second COUNT operator prints a message containing the count of records included from the IN2 data set.

DEFAULTS Operator

```
►►──DEFAULTS──LIST(listdd)──────────────────────────────────────────►◄
```

Prints the DFSORT installation defaults in a separate list data set.

DFSORT enables you to maintain eight separate sets of installation defaults using eight installation modules as follows:

- Environment installation modules
  - JCL (ICEAM1 module) - batch JCL directly invoked installation module
  - INV (ICEAM2 module) - batch program invoked installation module
  - TSO (ICEAM3 module) - TSO directly invoked installation module
  - TSOINV (ICEAM4 module) - TSO program invoked installation module
- Time-of-day installation modules
  - TD1 (ICETD1 module) - first time-of-day installation module
  - TD2 (ICETD2 module) - second time-of-day installation module
  - TD3 (ICETD3 module) - third time-of-day installation module
  - TD4 (ICETD4 module) - fourth time-of-day installation module

DEFAULTS Operator

Each installation default has two or more possible values; DFSORT is shipped with a set of IBM-supplied defaults that can be modified using the ICEMAC macro. The DEFAULTS operator provides an easy way to determine the installation defaults selected for each of the installation modules when DFSORT was installed. See *Installation and Customization* for a complete discussion of ICEMAC, the eight installation modules and the installation defaults and their values.

DEFAULTS produces a report showing the installation defaults for ICEAM1-4 followed by the installation defaults for ICETD1-4. The format of the report produced by DEFAULTS varies depending on the defaults selected, but might look like this conceptually:

```
DFSORT REL 14.0 INSTALLATION (ICEMAC) DEFAULTS       - p -

   * IBM-SUPPLIED DEFAULT (ONLY SHOWN IF DIFFERENT FROM THE SPECIFIED DEFAULT)

ITEM           JCL (ICEAM1)             INV (ICEAM2)             TSO (ICEAM3)             TSOINV (ICEAM4)
----------     --------------------     --------------------     --------------------     --------------------
item           value                    value                    value                    value
 .
 .
 .
item           value                    value                    value                    value
item           value                    value                    value                    value
                                        * IBM_value
 .
 .
 .

DFSORT REL 14.0 INSTALLATION (ICEMAC) DEFAULTS       - p -

   * IBM-SUPPLIED DEFAULT (ONLY SHOWN IF DIFFERENT FROM THE SPECIFIED DEFAULT)

ITEM           TD1 (ICETD1)             TD2 (ICETD2)             TD3 (ICETD3)             TD4 (ICETD4)
----------     --------------------     --------------------     --------------------     --------------------
item           value                    value                    value                    value
 .
 .
 .
item           value                    value                    value                    value
               * IBM_value              * IBM_value
item           value                    value                    value                    value
 .
 .
 .
```

The value for each item (for each of the eight installation environments) is shown as it is set in the ICEAM1-4 and ICETD1-4 installation modules loaded from the STEPLIB, JOBLIB, or link library. For any value that is different from the IBM-supplied value, the IBM-supplied value is shown below it.

The value for each item (for each of the eight installation environments) is shown as it is set in the ICEAM1-4 and ICETD1-4 installation modules loaded from the STEPLIB, JOBLIB, or link library. For any value that is different from the IBM-supplied value, the IBM-supplied value is shown below it.

The control character occupies the first byte of each record. The title and headings are always printed; p is the page number. The item name column occupies 10 bytes, each of the item value columns occupies 20 bytes, and 5 blanks appear between columns.

LIST(listdd)

Specifies the ddname of the list data set to be produced by ICETOOL for this operation. A listdd DD statement must be present. ICETOOL uses RECFM=FBA, LRECL=121 and the specified BLKSIZE for the list data set. If the BLKSIZE you specify is not a multiple of 121, ICETOOL uses BLKSIZE=121. If you do not specify the BLKSIZE, ICETOOL selects the block size as directed by the SDBMSG installation option from ICEAM2 or ICEAM4 (see *Installation and Customization*).

Refer to “JCL Restrictions” on page 359 for more information regarding the selection of ddnames.

DEFAULTS Example

```
DEFAULTS LIST(OPTIONS)
```

Prints, in the OPTIONS data set, the DFSORT installation defaults. The OPTIONS output starts on a new page and might look as follows (the first few items are shown with illustrative values for the ICEAM1-4 report and for the ICETD1-4 report):

DEFAULTS Operator

```
DFSORT REL 14.0 INSTALLATION (ICEMAC) DEFAULTS       - 1 -

   * IBM-SUPPLIED DEFAULT (ONLY SHOWN IF DIFFERENT FROM THE SPECIFIED DEFAULT)

ITEM          JCL (ICEAM1)            INV (ICEAM2)            TSO (ICEAM3)            TSOINV (ICEAM4)
----------    --------------------    --------------------    --------------------    --------------------
RELEASE       14.0                    14.0                    14.0                    14.0
MODULE        ICEAM1                  ICEAM2                  ICEAM3                  ICEAM4
APAR LEVEL    PQ35111                 PQ35111                 PQ35111                 PQ35111
COMPILED      07/31/00                07/31/00                07/31/00                07/31/00

ENABLE        NONE                    TD1                     NONE                    NONE

ABCODE        MSG                     99                      MSG                     99
                                      * MSG                                           * MSG
ALTSEQ        SEE BELOW               SEE BELOW               SEE BELOW               SEE BELOW
ARESALL       0                       0                       0                       0
ARESINV       NOT APPLICABLE          0                       NOT APPLICABLE          0
CFW           YES                     YES                     YES                     YES
CHALT         YES                     YES                     NO                      NO
              * NO                    * NO
CHECK         YES                     YES                     YES                     YES
CINV          YES                     YES                     YES                     YES
COBEXIT       COB2                    COB2                    COB2                    COB2
              * COB1                  * COB1                  * COB1                  * COB1
DIAGSIM       NO                      NO                      NO                      NO
DSA           32                      32                      32                      32
 .
 .
 .
DFSORT REL 14.0 INSTALLATION (ICEMAC) DEFAULTS       - 4 -

   * IBM-SUPPLIED DEFAULT (ONLY SHOWN IF DIFFERENT FROM THE SPECIFIED DEFAULT)

ITEM          TD1 (ICETD1)            TD2 (ICETD2)            TD3 (ICETD3)            TD4 (ICETD4)
----------    --------------------    --------------------    --------------------    --------------------
RELEASE       14.0                    14.0                    14.0                    14.0
MODULE        ICETD1                  ICETD2                  ICETD3                  ICETD3
APAR LEVEL    PQ35111                 PQ35111                 PQ35111                 PQ35111
COMPILED      07/31/00                07/24/00                07/24/00                07/24/00
SUN           0600-2000               NONE                    NONE                    NONE
              * NONE
MON           NONE                    NONE                    NONE                    NONE
TUE           NONE                    NONE                    NONE                    NONE
WED           NONE                    NONE                    NONE                    NONE
THU           NONE                    NONE                    NONE                    NONE
FRI           NONE                    NONE                    NONE                    NONE
SAT           0600-2000               NONE                    NONE                    NONE
              * NONE

ABCODE        99                      MSG                     MSG                     MSG
              * MSG
ALTSEQ        SEE BELOW               SEE BELOW               SEE BELOW               SEE BELOW
ARESALL       0                       0                       0                       0
ARESINV       0                       0                       0                       0
CFW           YES                     YES                     YES                     YES
CHALT         YES                     NO                      NO                      NO
              * NO
CHECK         YES                     YES                     YES                     YES
CINV          YES                     YES                     YES                     YES
COBEXIT       COB2                    COB1                    COB1                    COB1
              * COB1
DIAGSIM       NO                      NO                      NO                      NO
DSA           48                      32                      32                      32
              * 32
 .
 .
 .
```

The title and appropriate heading lines appear at the top of each page. The specified and IBM-supplied ALTSEQ tables are printed separately after the other items.

DISPLAY Operator

```
►►─DISPLAY─FROM(indd)─┬─ON(p,m,f)──────────────────┬─LIST(listdd)──────────►
                      ├─ON(p,m,f,formatting)───────┤
                      ├─ON(p,m,HEX)────────────────┤
                      ├─ON(VLEN)───────────────────┤
                      └─ON(NUM)────────────────────┘

►─┬─TITLE('string')─┬─┬─PAGE─┬─┬─DATE──────────┬─┬─TIME──────────┬─┬─BLANK─┬─
  └─────────────────┘ └──────┘ └─DATE(abcd)────┘ └─TIME(abc)────┘ └─PLUS──┘

►─┬─HEADER('string')─┬─┬─LINES(n)─┬─┬─TOTAL('string')─┬──────────────────────
  ├─HEADER(NONE)─────┤ └──────────┘ └─────────────────┘
  └─NOHEADER─────────┘

►─┬─MAXIMUM('string')─┬─┬─MINIMUM('string')─┬─┬─AVERAGE('string')─┬──────────
  └───────────────────┘ └───────────────────┘ └───────────────────┘

►─┬─LIMIT(n)─┬─┬─VSAMTYPE(x)─┬─┬─WIDTH(n)─┬──────────────────────────────────
  └──────────┘ └─────────────┘ └──────────┘

►─┬─BREAK(p,m,f)─┬─┬─BTITLE('string')─┬─┬─BTOTAL('string')─┬──────────────────
  └──────────────┘ └──────────────────┘ └──────────────────┘

►─┬─BMAXIMUM('string')─┬─┬─BMINIMUM('string')─┬─┬─BAVERAGE('string')─┬──────
  └────────────────────┘ └────────────────────┘ └────────────────────┘

►─┬─UZERO─┬──────────────────────────────────────────────────────────────►◄
  └───────┘
```

Prints the values or characters of specified numeric or character fields in a separate list data set. Simple, tailored, and sectioned reports can be produced. From 1 to 20 fields can be specified, but the resulting list data set line length must not exceed the limit specified by the WIDTH operand or 2048 bytes if WIDTH is not specified. The record number can be printed as a special field.

DFSORT is called to copy the indd data set to ICETOOL's E35 user exit. ICETOOL uses its E35 user exit to print appropriate titles, headings and data in the list data set.

You must not supply your own DFSORT MODS, INREC, or OUTREC statement since they would override the DFSORT statements passed by ICETOOL for this operator.

Specifying formatting items or the PLUS or BLANK operand, which can "compress" the columns of output data, can enable you to include more

fields in your report, up to a maximum of 20, if your line length is limited by the character width your printer or display supports.

Simple Report

You can produce a simple report by specifying just the required operands. For example, if you specify FROM and LIST operands, and ON operands for 10-byte character and 7-byte zoned decimal fields, the output in the list data set can be represented as follows:

```
(p,m,f)                (p,m,f)
characters             sddddddddddddd
 .                                 .
 .                                 .
 .                                 .
```

A control character occupies the first byte of each list data set record. Left-justified standard headings are printed at the top of each page to indicate the contents of each column, followed by a line for each record showing the characters and numbers in the fields of that record.

The fields are printed in columns in the same order in which they are specified in the DISPLAY statement. All fields are left-justified. For numeric fields, leading zeros are printed, a - is used for the minus sign, and a + is used for the plus sign.

Three blanks appear between columns.

The standard column widths are as follows:

- Character data: the length of the character field or 20 bytes if the field length is less than 21 bytes
- Numeric data: 16 bytes
- Record number: 15 bytes

HEADER operands can be used to change or suppress the headings. Formatting items or the PLUS or BLANK operand can be used to change the appearance of numeric fields in the report. PLUS, BLANK, and HEADER operands can be used to change the width of the columns for numeric and character fields and the justification of headings and fields.

The NOHEADER operand can be used to create list data sets containing only data records. Data sets created in this way can be processed further by other operators (for example, STATS or UNIQUE) using CH format for character values or CSF/FS format for numeric values.

TOTAL, MAXIMUM, MINIMUM, and AVERAGE can be used to print statistics for numeric fields after the columns of data.

Tailored Report

You can tailor the output in the list data set using various operands that control title, date, time, page number, headings, lines per page, field formats, and total, maximum, minimum, and average values for the columns of numeric data. The optional operands can be used in many different combinations to produce a wide variety of report formats. For example, if you specify FROM, LIST, BLANK, TITLE, PAGE, DATE, TIME, HEADER and AVERAGE operands, and ON operands for 10-byte character and 7-byte zoned decimal fields, the output in the list data set can be represented as follows:

```
title        - p -        mm/dd/yy        hh:mm:ss

header         header
----------   --------
characters         sd
.                   .
.                   .
.                   .

average            sd
```

A control character occupies the first byte of each list data set record. The title line is printed at the top of each page of the list data set. It contains the elements you specify (title string, page number, date and time) in the order in which you specify them. Eight blanks appear between title elements. A blank line is printed after the title line.

Your specified headings (underlined) are printed after the title line on each page to indicate the contents of each column, followed by a line for each record showing the characters and numbers in the fields of that record. Headings for character fields are left-justified and headings for numeric fields are right-justified.

Your specified statistical lines (total, maximum, minimum, and average, and their associated strings) are printed for each numeric field after the columns of data.

The fields are printed in columns in the same order in which they are specified in the DISPLAY statement. Character fields are left-justified and numeric fields are right-justified. For numeric fields, leading zeros are suppressed, a - is used for the minus sign, and a blank is used for the plus sign (you can specify PLUS rather than BLANK if you want a + to be used for the plus sign).

Formatting items can be used to change the appearance of individual numeric fields in the report with respect to separators, decimal point, decimal places, and signs; division by 1000, 1000 000 (1000*1000), 1000 000 000 (1000*1000*1000), 1024, 1 048 576 (1024*1024), or 1 073 741 824 (1024*1024*1024); leading strings, floating strings, and trailing strings.

Three blanks appear between columns.

The column widths are dynamically adjusted according to the length of the headings and the maximum number of bytes needed for the character or numeric data.

Sectioned Report

You can produce a sectioned report (simple or tailored) by including a BREAK operand to indicate the break field to be used to divide the report into sections. Each set of sequential input records (previously sorted on the break field and other fields, as appropriate), with the same value for the specified break field, results in a corresponding set of data lines that is treated as a section in the report. Optional break operands can be used to modify the break title for each section (the break value is always printed as part of the break title) and to print statistics for each section. For example, if you add BTITLE, BREAK, BMAXIMUM, and BMINIMUM to the operands for the tailored report discussed above, each section of the output in the list data set starts on a new page and can be represented as follows:

```
title       - p -       mm/dd/yy       hh:mm:ss

btitle  bvalue

header        header
----------   --------
characters        sd
.                  .
.                  .
.                  .

bmaximum          sd

bminimum          sd
```

The final page showing the overall statistics starts on a new page and can be represented as follows:

```
title       - p -       mm/dd/yy       hh:mm:ss

header        header
----------   --------

average           sd
```

The operands described below can be specified in any order.

FROM(indd)
Specifies the ddname of the input data set to be read by DFSORT for this operation. An indd DD statement must be present and must define an input data set that conforms to the rules for DFSORT's SORTIN data set. In addition, the LRECL of the data set must be at least 4.

ON(p,m,f)
Specifies the position, length, and format of a numeric or character field to be used for this operation. '(p,m,f)' is used for the standard column heading (see HEADER('string'), HEADER(NONE) and NOHEADER for alternative heading options).

p specifies the first byte of the field relative to the beginning of the input record. p is 1 for the first **data** byte of a fixed-length record and 5 for the first **data** byte of a variable-length record as illustrated below (RRRR represents the 4-byte record descriptor word):

```
   Fixed-length record     |    Variable-length record
   | D | A | T | A | ...  |    | R | R | R | R | D | A | T | A | ...
p=   1   2   3   4         | p=   1   2   3   4   5   6   7   8
```

m specifies the length of the field in bytes. A field must not extend beyond position 32752 or beyond the end of a record. The maximum length for a field depends on its format.

f specifies the format of the field as shown below.

Format Code	Length	Description
BI	1 to 4 bytes	Unsigned binary
FI	1 to 4 bytes	Signed fixed-point
PD	1 to 8 bytes	Signed packed decimal

Format Code	Length	Description
ZD	1 to 15 bytes	Signed zoned decimal
CH	1 to 80 bytes	Character
CSF or FS	1 to 16 bytes (15 digit limit)	Signed numeric with optional leading floating sign

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

For a CSF or FS format field:

- A maximum of 15 digits is allowed. If a CSF/FS value with 16 digits is found, ICETOOL issues an error message and terminates the operation.

For a ZD or PD format field:

- If a decimal value contains an invalid digit (A-F), ICETOOL identifies the bad value in a message and prints asterisks for that value, and for the total, maximum, minimum and average (if specified) for that field, in the list data set. If the number of bad values reaches the LIMIT for invalid decimal values, ICETOOL terminates the operation. If the LIMIT operand is not specified, a default of 200 is used for the invalid decimal value limit.
- A value is treated as positive if its sign is F, E, C, A, 8, 6, 4, 2, or 0.
- A value is treated as negative if its sign is D, B, 9, 7, 5, 3, or 1.

ON(p,m,f,formatting)

Specifies the position, length, and format of a numeric or character field to be used for this operation and how the data for this field is to be formatted for printing. ’(p,m,f)’ is used for the standard column heading (see HEADER(’string’), HEADER(NONE) and NOHEADER for alternative heading options). If the PLUS operand is not specified, the BLANK operand is automatically used. Column widths are dynamically adjusted according to the maximum number of bytes needed for the formatted data.

See ON(p,m,f) for a discussion of **p, m** and **f**.

formatting

specifies formatting items that indicate how the data for this field is to be

```
   ┌─,──────────────────────────┐
   ▼                            │
►►─┬─mask──────────────────────┬─┴──────────────────────────────────────────►◄
   │  ┌─/K──────────────────┐  │
   │  ├─/M──────────────────┤  │
   │  ├─/G──────────────────┤  │
   │  ├─/KB─────────────────┤  │
   │  ├─/MB─────────────────┤  │
   │  └─/GB─────────────────┘  │
   ├─L’string’ ────────────────┤
   ├─F’string’ ────────────────┤
   └─T’string’ ────────────────┘
```

formatted for printing. For each ON field, formatting items can be specified in any order and combination, but each item can only be specified once, and only one division item (/K, /M, /G, /KB, /MB or /GB) can be specified. The column width is dynamically adjusted to accommodate the maximum bytes to be inserted as a result of all formatting items specified.

mask

specifies an edit mask to be applied to the numeric data for this field. Thirty-three pre-defined edit masks are available, encompassing many of the numeric notations throughout the world with respect to separators, decimal point, decimal places, signs, and so forth. ICETOOL edits the data according to the selected mask. If other formatting items are specified but mask is not, the default mask of A0 is applied to the data.

The attributes of each group of masks is shown below.

Table 53. Attributes of Edit Masks

Masks	Separators	Decimal Places	Positive Sign	Negative Sign
A0	No	0	blank	-
A1-A5	Yes	0	blank	-
B1-B6	Yes	1	blank	-
C1-C6	Yes	2	blank	-
D1-D6	Yes	3	blank	-
E1-E4	Yes	0	blank	( )
F1-F5	Yes	2	blank	( )

The table below describes the available masks and shows how the values 12345678 and -1234567 would be printed for each mask. In the pattern:

- **d** is used to represent a decimal digit (0-9)
- **w** is used to represent a leading sign that will be blank for a positive value or - for a negative value
- **x** is used to represent a trailing sign that will be blank for a positive value or - for a negative value
- **y** is used to represent a leading sign that will be blank for a positive value or (for a negative value
- **z** is used to represent a trailing sign that will be blank for a positive value or) for a negative value

Table 54. Edit Mask Patterns

Mask	Pattern	12345678	-1234567
A0	wddddddddddddddd	12345678	-1234567
A1	wddd,ddd,ddd,ddd,ddd	12,345,678	-1,234,567
A2	wddd.ddd.ddd.ddd.ddd	12.345.678	-1.234.567
A3	wddd ddd ddd ddd ddd	12 345 678	-1 234 567
A4	wddd’ddd’ddd’ddd’ddd	12'345'678	-1'234'567
A5	ddd ddd ddd ddd dddx	12 345 678	1 234 567-
B1	wdd,ddd,ddd,ddd,ddd.d	1,234,567.8	-123,456.7
B2	wdd.ddd.ddd.ddd.ddd,d	1.234.567,8	-123.456,7
B3	wdd ddd ddd ddd ddd,d	1 234 567,8	-123 456,7
B4	wdd’ddd’ddd’ddd’ddd.d	1'234'567.8	-123'456.7
B5	wdd’ddd’ddd’ddd’ddd,d	1'234'567,8	-123'456,7

Table 54. Edit Mask Patterns (continued)

Mask	Pattern	12345678	-1234567
B6	dd ddd ddd ddd ddd,dx	1 234 567,8	123 456,7-
C1	wd,ddd,ddd,ddd,ddd.dd	123,456.78	-12,345.67
C2	wd.ddd.ddd.ddd.ddd,dd	123.456,78	-12.345,67
C3	wd ddd ddd ddd ddd,dd	123 456,78	-12 345,67
C4	wd'ddd'ddd'ddd'ddd.dd	123'456.78	-12'345.67
C5	wd'ddd'ddd'ddd'ddd,dd	123'456,78	-12'345,67
C6	d ddd ddd ddd ddd,ddx	123 456,78	12 345,67-
D1	wddd,ddd,ddd,ddd.ddd	12,345.678	-1,234.567
D2	wddd.ddd.ddd.ddd,ddd	12.345,678	-1.234,567
D3	wddd ddd ddd ddd,ddd	12 345,678	-1 234,567
D4	wddd'ddd'ddd'ddd.ddd	12'345.678	-1'234.567
D5	wddd'ddd'ddd'ddd,ddd	12'345,678	-1'234,567
D6	ddd ddd ddd ddd,dddx	12 345,678	1 234,567-
E1	yddd,ddd,ddd,ddd,dddz	12,345,678	(1,234,567)
E2	yddd.ddd.ddd.ddd.dddz	12.345.678	(1.234.567)
E3	yddd ddd ddd ddd dddz	12 345 678	(1 234 567)
E4	yddd'ddd'ddd'ddd'dddz	12'345'678	(1'234'567)
F1	yd,ddd,ddd,ddd,ddd.ddz	123,456.78	(12,345.67)
F2	yd.ddd.ddd.ddd.ddd,ddz	123.456,78	(12.345,67)
F3	yd ddd ddd ddd ddd,ddz	123 456,78	(12 345,67)
F4	yd'ddd'ddd'ddd'ddd.ddz	123'456.78	(12'345.67)
F5	yd'ddd'ddd'ddd'ddd,ddz	123'456,78	(12'345,67)

Leading zeros are suppressed except when inappropriate. For example, 00000 is shown as 0 with A1 and as 0.00 with C1.

The leading sign appears to the left of the first non-suppressed digit of the formatted value. For example, -000001 is shown as -1 with A2 and as -0,01 with C2.

/K specifies division of the numeric data for this field by 1000 before formatting. The resulting values are rounded down to the nearest integer. For example, -1234567890 is shown as -1 234 567 with ON(1,11,FS,/K,A3) and as (1 234 567) with ON(1,11,FS,/K,E3).

/M specifies division of the numeric data for this field by 1000 000 (1000*1000) before formatting. The resulting values are rounded down to the nearest integer. For example, -123456789 is shown as -1.23 with ON(31,10,FS,/M,C4) and as (1.23) with ON(31,10,FS,/M,F4).

/G specifies division of the numeric data for this field by 1000 000 000 (1000*1000*1000) before formatting. The resulting values are rounded down to the nearest integer. For example, 1234567898765 is shown as 1'234 with ON(15,13,ZD,/G,A4).

/KB
specifies division of the numeric data for this field by 1024 before

formatting. The resulting values are rounded down to the nearest integer. For example, 1234567890 is shown as 1 205 632 with ON(45,10,ZD,/KB,A3).

/MB
specifies division of the numeric data for this field by 1048576 (1024*1024) before formatting. The resulting values are rounded down to the nearest integer. For example, 123456789 is shown as 117 with ON(60,9,FS,/MB).

/GB
specifies division of the numeric data for this field by 1073741824 (1024*1024*1024) before formatting. The resulting values are rounded down o the nearest integer. For example, 1234567898765 is shown as 1,149 with ON(15,13,ZD,/GB,A1).

L'string'
specifies a leading string to appear at the beginning of the character or numeric data column for this field. For example, 'DFSORT ' is shown as '**DFSORT ' with ON(1,8,CH,L'**').

The string (1 to 10 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (').

F'string'
specifies a floating string to appear to the left of the first non-blank character of the formatted numeric data for this field. For example, 0001234 is shown as $12.34 with ON(9,7,ZD,C1,F'$').

The string (1 to 10 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (').

T'string'
specifies a trailing string to appear at the end of the character or numeric data column for this field. For example, 'DFSORT ' is shown as '**DFSORT ***' with ON(1,8,CH,L'**',T'***').

The string (1 to 10 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (').

ON(p,m,HEX)
Specifies the position and length of a character field to be used for this operation and printed in hexadecimal format (00-FF for each byte). ' (p,m,HEX)' is used for the standard column heading. See HEADER('string'), HEADER(NONE), and NOHEADER for alternative heading options.

See ON(p,m,f) for a discussion of **p**.

m specifies the length of the field in bytes. A field must not extend beyond position 32752 or beyond the end of a record. A field can be 1 to 50 bytes.

ON(VLEN)
Equivalent to specifying ON(1,2,BI); a two-byte binary field starting at position 1. For variable-length records, ON(VLEN) represents the record-length for each

record. ' RECORD LENGTH' is used for the standard column heading. See HEADER('string'), HEADER(NONE), and NOHEADER for alternative heading options.

ON(NUM)
Specifies that the record number is to be printed. The record number starts at 1 and is incremented by 1 for each record printed in the list data set. ' RECORD NUMBER' is used for the standard column heading. See HEADER('string'), HEADER(NONE), and NOHEADER for alternative heading options.

LIST(listdd)
Specifies the ddname of the list data set to be produced by ICETOOL for this operation. A listdd DD statement must be present. ICETOOL sets the attributes of the list data set as follows:

- RECFM is set to FBA.
- LRECL is set to one of the following:
  - If WIDTH(n) is specified, LRECL is set to n. Use WIDTH(n) if your LRECL must be set to a particular value (for example, if you use DISP=MOD to place several reports in the same data set).
  - If WIDTH(n) is not specified, LRECL is set to 121 or to the calculated required line length if it is greater than 121 characters. If your LRECL does not need to be set to a particular value, you can let ICETOOL determine and set the appropriate LRECL value by not specifying WIDTH(n).
- BLKSIZE is set to one of the following:
  - The BLKSIZE from the DD statement, DSCB, or label, if it is a multiple of the LRECL used.
  - The LRECL if the BLKSIZE from the DD statement, DSCB, or label is not a multiple of the LRECL used.
  - The block size as directed by the SDBMSG installation option (see *Installation and Customization* if the BLKSIZE is not available from the DD statement, DSCB, or label.

Refer to "JCL Restrictions" on page 359 for more information regarding the selection of ddnames.

TITLE('string')
Specifies printing of a title string in the title line. The title line is printed at the top of each page of the list data set. It contains the elements you specify (title string, page number, date and time) in the order in which you specify them. Eight blanks appear between title elements. A blank line is printed after the title line.

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). Blanks at the start of the string move the text to the right. Blanks at the end of the string increase the spacing between the string and the next title element.

PAGE
Specifies printing of the page number in the title line. The page number is printed in the form - p - where p is in decimal with no leading zeros. The page number is 1 for the first page and is incremented by 1 for each subsequent page.

The title line is printed at the top of each page of the list data set. It contains the elements you specify (title string, page number, date and time) in the order in which you specify them. Eight blanks appear between title elements. A blank line is printed after the title line.

DATE

Specifies printing of the date in the title line. The date is printed in the form mm/dd/yy where mm is the month, dd is the day, and yy is the year. DATE is equivalent to specifying DATE(MDY/).

The title line is printed at the top of each page of the list data set. It contains the elements you specify (title string, page number, date and time) in the order in which you specify them. Eight blanks appear between title elements. A blank line is printed after the title line.

DATE(abcd)

Specifies printing of the date in the title line. The date is printed in the form aadbbdcc according to the specified values for abc and d.

abc can be any combination of M, D, and Y or 4 (each specified once) where M represents the month (01-12), D represents the day (01-31), Y represents the last two digits of the year (for example, 95), and 4 represents the four digits of the year (for example, 1995).

d can be any character and is used to separate the month, day, and year.

The title line is printed at the top of each page of the list data set. It contains the elements you specify (title string, page number, date and time) in the order in which you specify them. Eight blanks appear between title elements. A blank line is printed after the title line.

TIME

Specifies printing of the time in the title line. The time is printed in the form hh:mm:ss where hh is hours, mm is minutes and ss is seconds. TIME is equivalent to specifying TIME(24:).

The title line is printed at the top of each page of the list data set. It contains the elements you specify (title string, page number, date and time) in the order in which you specify them. Eight blanks appear between title elements. A blank line is printed after the title line.

TIME(abc)

Specifies printing of the time in the title line. The time is printed in the form hhcmmcss xx according to the specified value for ab and c.

ab can be:

- 12 to indicate 12-hour time. hh (hours) is 1-12, mm (minutes) is 0-59, ss (seconds) is 0-59 and xx is am or pm.
- 24 to indicate 24-hour time. hh (hours) is 0-23, mm (minutes) is 0-59, ss (seconds) is 0-59 and xx is not included.

c can be any character and is used to separate the hours, minutes, e is printed at the top of each page of the list data set. It contains the elements you specify (title string, page number, date and time) in the order in which you specify them. Eight blanks appear between title elements. A blank line is printed after the title line.

BLANK

Specifies an alternate format for printing character and numeric data as follows:

- Numeric values for which formatting is not specified are printed with blank for plus sign, - for minus sign and no leading zeros (overriding the default of + for plus sign and leading zeros).

  Numeric values are thus displayed as:

  – d...d for positive values (blank sign immediately to the left of the digits and no leading zeros)

  – -d...d for negative values (- sign immediately to the left of the digits and no leading zeros)
- Column widths are dynamically adjusted according to the length of the headings and the maximum number of bytes needed for the character or numeric data
- Headings and data for numeric fields are right-justified (overriding the default of left-justified headings and data for numeric fields)

PLUS

Specifies an alternate format for printing character and numeric data as follows:

- Numeric values for which formatting is not specified are printed with + for plus sign, - for minus sign and no leading zeros (overriding the default of leading zeros).

  Numeric values are thus displayed as:

  – +d...d for positive values (+ sign immediately to the left of the digits and no leading zeros)

  – -d...d for negative values (- sign immediately to the left of the digits and no leading zeros)
- Column widths are dynamically adjusted according to the length of the headings and the maximum number of bytes needed for the character or numeric data
- Headings and data for numeric fields are right-justified (overriding the default of left-justified headings and data for numeric fields)

For ON(NUM), PLUS is treated as BLANK.

HEADER('string')

Specifies a heading to be printed for the corresponding ON field. The specified string is used instead of the standard column heading for the corresponding ON field. (ON fields and HEADER operands correspond one-for-one according to the order in which they are specified; that is, the first HEADER operand corresponds to the first ON field, the second HEADER operand corresponds to the second ON field, and so on.)

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). If the string length is greater than the column width for the corresponding ON field, the column width is increased to the string length.

The heading is left-justified for character fields or right-justified for numeric fields and is underlined with dashes for the entire column width (overriding the default of left-justified, non-underlined headings). Character values are left-justified and numeric values are right-justified (overriding the default of left-justified field values).

Blanks at the start or end of a heading string may alter the justification of the heading or the width of the column.

If HEADER('string') is used for any ON field, HEADER('string') or HEADER(NONE) must be used for each ON field.

HEADER(NONE)
Specifies that a heading is not to be printed for the corresponding ON field. The standard column heading for the corresponding ON field is suppressed.

If HEADER('string') is used for any ON field, HEADER('string') or HEADER(NONE) must be used for each ON field. Specifying HEADER(NONE) for every ON field is equivalent to specifying NOHEADER.

NOHEADER
Specifies that headings for ON fields are not to be printed (overriding the default of printing standard headings for ON fields).

If NOHEADER is used, it must be specified only once and HEADER('string') or HEADER(NONE) must not be used.

If NOHEADER is specified without any TITLE, DATE, TIME, or PAGE operands, the resulting list data set contains only data records. Data sets created in this way can be processed further by other operators (for example, STATS or UNIQUE) using CH for character values or CSF/FS for numeric values.

LINES(n)
Specifies the number of lines per page for the list data set (overriding the default of 58). n must be greater than 9, but less than 1000.

TOTAL('string')
Specifies an overall TOTAL line is to be printed after the columns of data for the report. The specified string is printed starting in column 2 of the overall TOTAL line, followed by the overall total for each numeric data column on the same line as the string or on the next line, as appropriate. A blank line is printed before the overall TOTAL line.

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). To suppress printing of a string, specify TOTAL('').

The overall total for each numeric ON field is printed in the format (formatting, PLUS, BLANK, or standard) you specify. Totals are printed for ON(VLEN) fields, but not for ON(NUM) fields.

The column widths for numeric ON fields are adjusted to allow for a maximum of a sign and 15 digits for the totals. If the overall total for an ON field overflows 15 digits, ICETOOL prints asterisks for the overall total for that field.

The TOTAL, MAXIMUM, MINIMUM, and AVERAGE lines are printed in the order in which you specify them.

MAXIMUM('string')
Specifies an overall MAXIMUM line is to be printed after the columns of data for the report. The specified string is printed starting in column 2 of the overall MAXIMUM line, followed by the overall maximum for each numeric data column on the same line as the string or on the next line, as appropriate. A blank line is printed before the overall MAXIMUM line.

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). To suppress printing of a string, specify MAXIMUM('').

The overall maximum for each numeric ON field is printed in the format (formatting, PLUS, BLANK, or standard) you specify. Maximums are printed for ON(VLEN) fields, but not for ON(NUM) fields.

The TOTAL, MAXIMUM, MINIMUM, and AVERAGE lines are printed in the order in which you specify them.

MINIMUM('string')
Specifies an overall MINIMUM line is to be printed after the columns of data for the report. The specified string is printed starting in column 2 of the overall MINIMUM line, followed by the overall minimum for each numeric data column on the same line as the string or on the next line, as appropriate. A blank line is printed before the overall MINIMUM line.

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). To suppress printing of a string, specify MINIMUM('').

The overall minimum for each numeric ON field is printed in the format (formatting, PLUS, BLANK, or standard) you specify. Minimums are printed for ON(VLEN) fields, but not for ON(NUM) fields.

The TOTAL, MAXIMUM, MINIMUM, and AVERAGE lines are printed in the order in which you specify them.

AVERAGE('string')
Specifies an overall AVERAGE line is to be printed after the columns of data for the report. The specified string is printed starting in column 2 of the overall AVERAGE line, followed by the overall average for each numeric data column on the same line as the string or on the next line, as appropriate. A blank line is printed before the overall AVERAGE line.

The overall average (or mean) is calculated by dividing the overall total by the number of values in the report and rounding down to the nearest integer (examples: 23 / 5 = 4, -23 / 5 = -4).

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). To suppress printing of a string, specify AVERAGE('').

The overall average for each numeric ON field is printed in the format (formatting, PLUS, BLANK, or standard) you specify. Averages are printed for ON(VLEN) fields, but not for ON(NUM) fields.

If the overall total for an ON field overflows 15 digits, ICETOOL prints asterisks for the overall average for that field.

The TOTAL, MAXIMUM, MINIMUM, and AVERAGE lines are printed in the order in which you specify them.

LIMIT(n)
Specifies a limit for the number of invalid decimal values (overriding the default

of 200). If n invalid decimal values are found, ICETOOL terminates the operation. n can be 1 to 15 decimal digits, but must be greater than 0.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

WIDTH(n)
Specifies the line length and LRECL you want ICETOOL to use for your list data set. n can be from 121 to 2048.

ICETOOL always calculates the line length required to print all titles, headings, data, and statistics and uses it as follows:

- If WIDTH(n) is specified and the calculated line length is greater than n, ICETOOL issues an error message and terminates the operation. Otherwise, ICETOOL sets the line length and LRECL to n.
- If WIDTH(n) is not specified and the calculated line length is less than or equal to 121, ICETOOL sets the line length and LRECL to 121.
- If WIDTH(n) is not specified and the calculated line length is greater than 121, ICETOOL sets the line length and LRECL to the calculated line length.

Use WIDTH(n) if your LRECL must be set to a particular value (for example, if you use DISP=MOD to place several reports in the same data set) or if you want to ensure that the line length for your report does not exceed a specific maximum (for example, 133 bytes). Otherwise, you can let ICETOOL calculate and set the appropriate line length and LRECL by not specifying WIDTH(n).

BREAK(p,m,f)
Specifies a numeric or character break field to be used to divide the report into sections. Each set of sequential input records, with the same value for the specified break field, results in a corresponding set of data lines that is treated as a section in the report. The DISPLAY operator should be preceded by a SORT operator (or another application) that sorts the break field and any other appropriate fields in the desired sequence for the report.

Each section starts on a new page. Each page of a section includes a break title line showing the break value for the section. Numeric break values are printed with blank for plus sign, - for minus sign, and no leading zeros. BTITLE can be used to specify a string to appear in the break title line. The break value and break title string appear in the order in which you specify BREAK and BTITLE. Two blanks appear between break title elements. A blank line is printed after the break title line.

BTOTAL, BMAXIMUM, BMINIMUM, and BAVERAGE can be used to produce break statistics for each numeric ON field-for example, the maximum of the values in the section for ON(5,3,ZD) and the maximum of the values in the section for ON(22,2,BI). The break statistics for each section are printed at the end of the section (on one or more pages which include the break title). TOTAL, MAXIMUM, MINIMUM, and AVERAGE can be used to produce overall statistics for each numeric ON field-for example, the maximum of the values in the report for ON(5,3,ZD) and the maximum of the values in the report for ON(22,2,BI). The overall statistics for each section are printed at the end of the report (on a separate page which does not include the break title).

See ON(p,m,f) for a discussion of **p** and **m**.

f specifies the format of the field as shown for ON(p,m,f).

For a CSF or FS format break field:

- A maximum of 15 digits is allowed. If a value with 16 digits is found, ICETOOL issues an error message and terminates the operation.

For a ZD or PD format break field:

- If a decimal value with an invalid digit (A-F) is found, ICETOOL issues an error message and terminates the operation.
- A value is treated as positive if its sign is F, E, C, A, 8, 6, 4, 2, or 0.
- A value is treated as negative if its sign is D, B, 9, 7, 5, 3, or 1.

BTITLE('string')

Specifies a string to appear in the break title line printed for each page of a section. BTITLE can only be specified if BREAK is specified. The break value and break title string appear in the order in which you specify BREAK and BTITLE. Two blanks appear between break title elements. A blank line is printed after the break title line.

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). Blanks at the start of the string move the text to the right. Blanks at the end of the string increase the spacing between the string and the break value if BTITLE is specified before BREAK.

BTOTAL('string')

Specifies a break TOTAL line is to be printed after the columns of data for each section. BTOTAL can only be specified if BREAK is specified. The specified string is printed starting in column 2 of the break TOTAL line, followed by the break total for each numeric data column on the same line as the string or on the next line, as appropriate. A blank line is printed before the break TOTAL line.

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). To suppress printing of a string, specify BTOTAL('').

The break total for each numeric ON field is printed in the format (formatting, PLUS, BLANK, or standard) you specify. Totals are printed for ON(VLEN) fields, but not for ON(NUM) fields.

The column widths for numeric ON fields are adjusted to allow for a maximum of a sign and 15 digits for the totals. If the break total for an ON field overflows 15 digits, ICETOOL prints asterisks for the break total for that field.

The BTOTAL, BMAXIMUM, BMINIMUM, and BAVERAGE lines are printed in the order in which you specify them.

BMAXIMUM('string')

Specifies a break MAXIMUM line is to be printed after the columns of data for each section. BMAXIMUM can only be specified if BREAK is specified. The specified string is printed starting in column 2 of the break MAXIMUM line, followed by the break maximum for each numeric data column on the same line as the string or on the next line, as appropriate. A blank line is printed before the break MAXIMUM line.

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). To suppress printing of a string, specify BMAXIMUM('').

The break maximum for each numeric ON field is printed in the format (formatting, PLUS, BLANK, or standard) you specify. Maximums are printed for ON(VLEN) fields, but not for ON(NUM) fields.

The BTOTAL, BMAXIMUM, BMINIMUM, and BAVERAGE lines are printed in the order in which you specify them.

BMINIMUM('string')
Specifies a break MINIMUM line is to be printed after the columns of data for each section. BMINIMUM can only be specified if BREAK is specified. The specified string is printed starting in column 2 of the break MINIMUM line, followed by the break minimum for each numeric data column on the same line as the string or on the next line, as appropriate. A blank line is printed before the break MINIMUM line.

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). To suppress printing of a string, specify BMINIMUM('').

The break minimum for each numeric ON field is printed in the format (formatting, PLUS, BLANK, or standard) you specify. Minimums are printed for ON(VLEN) fields, but not for ON(NUM) fields.

The BTOTAL, BMAXIMUM, BMINIMUM, and BAVERAGE lines are printed in the order in which you specify them.

BAVERAGE('string')
Specifies a break AVERAGE line is to be printed after the columns of data for each section. BAVERAGE can only be specified if BREAK is specified. The specified string is printed starting in column 2 of the break AVERAGE line, followed by the break average for each numeric data column on the same line as the string or on the next line, as appropriate. A blank line is printed before the break AVERAGE line.

The break average (or mean) is calculated by dividing the break total by the number of values in the section and rounding down to the nearest integer (examples: 23 / 5 = 4, -23 / 5 = -4).

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes (''). To suppress printing of a string, specify BAVERAGE('').

The break average for each numeric ON field is printed in the format (formatting, PLUS, BLANK, or standard) you specify. Averages are printed for ON(VLEN) fields, but not for ON(NUM) fields.

If the break total for an ON field overflows 15 digits, ICETOOL prints asterisks for the break average for that field.

The BTOTAL, BMAXIMUM, BMINIMUM, and BAVERAGE lines are printed in the order in which you specify them.

UZERO
Specifies that -0 and +0 are to be treated as unsigned zero values, that is, as the same value. With UZERO, -0 and +0 are treated as positive for ON, MINIMUM, MAXIMUM, BREAK, BMINIMUM and BMAXIMUM processing.

UZERO overrides the default of treating -0 and +0 as signed zero values, that is, as different values. Without UZERO, -0 is treated as negative and +0 is treated as positive for ON, MINIMUM, MAXIMUM, BREAK, BMINIMUM and BMAXIMUM processing.

DISPLAY Examples

Although the DISPLAY operators in the examples below could all be contained in a single ICETOOL job step, they are shown and discussed separately for clarity. See “OCCUR Operator” on page 398 for additional examples of tailoring the report format.

Example 1

```
DISPLAY FROM(SOURCE) LIST(FIELDS) ON(NUM) ON(40,12,CH) -
ON(20,8,PD)
```

Prints, in the FIELDS data set:

- A heading line containing the standard headings
- Data lines in the standard format containing:
  - The record number in the standard format
  - The characters from positions 40-51 of the SOURCE data set
  - The packed decimal values from positions 20-27 of the SOURCE data set in the standard format

The FIELDS output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
RECORD NUMBER     (40,12,CH)            (20,8,PD)
000000000000001   SAN JOSE              000000000003745
000000000000002   MORGAN HILL           000000000016502
              .   .                                   .
              .   .                                   .
              .   .                                   .
```

The heading line appears at the top of each page.

Example 2

```
DISPLAY FROM(IN) LIST(LIST1) -
      TITLE('National Accounting Report') -
      PAGE DATE TIME -
      HEADER('Division') HEADER('Revenue') HEADER('Profit/Loss') -
      ON(1,25,CH)        ON(45,10,ZD)       ON(35,10,ZD) -
      BLANK -
      TOTAL('Company Totals') -
      AVERAGE('Company Averages')
```

Prints, in the LIST1 data set:

- A title line containing the specified title, the page number, the date and the time
- A heading line containing the specified underlined headings
- Data lines in the BLANK format containing:
  - The characters from positions 1-25 of the IN data set
  - The zoned decimal values from positions 45-54 of the IN data set
  - The zoned decimal values from positions 35-44 of the IN data set

- A TOTAL line containing the specified string and the total for each of the two zoned decimal fields in the BLANK format
- An AVERAGE line containing the specified string and the average for each of the two zoned decimal fields in the BLANK format.

The LIST1 output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
National Accounting Report        - 1 -        10/21/92        18:52:44

Division                              Revenue        Profit/Loss
-------------------------   ----------------   ----------------
Research and Development            54323456            -823325
Manufacturing                      159257631            1372610
.                                          .                  .
.                                          .                  .
.                                          .                  .

Company Totals                     612867321            5277836

Company Averages                    76608415             659729
```

The title line and underlined heading line appear at the top of each page.

Example 3

```
DISPLAY FROM(DATA) LIST(JUSTDATA) -
  NOHEADER -
  ON(17,5,PD) ON(1,2,FI)
```

Prints, in the JUSTDATA data set:

- Data lines in the standard format containing:
  - The packed decimal values from positions 17-21 of the DATA data set in the standard format
  - The fixed-point values from positions 1-2 of the DATA data set in the standard format

The JUSTDATA output contains no page ejects or heading lines and looks as follows (the first 2 records are shown with illustrative values):

```
-0000000000273216   +0000000000000027
+0000000000993112   +0000000000000321
                .                   .
                .                   .
                .                   .
```

Example 4

```
COPY FROM(INPUT) TO(TEMP) USING(TREG)
DISPLAY FROM(TEMP) LIST(REGULAR) -
  TITLE('Report on Regular Tools      ') PAGE -
  HEADER(NONE) ON(1,18,CH) -
  HEADER('Item') ON(35,5,CH) -
  HEADER('Percent Change') ON(28,4,FS,B1) -
  LINES(66)
COPY FROM(INPUT) TO(TEMP) USING(TPOW)
DISPLAY FROM(TEMP) LIST(POWER) -
  TITLE('Report on Power Tools        ') PAGE -
  HEADER(NONE) ON(1,18,CH) -
  HEADER('Item') ON(35,5,CH) -
  HEADER('Percent Change') ON(28,4,FS,B1) -
  LINES(66)
```

This example shows how reports for different subsets of data can be produced. Assume that:

- The TREGCNTL data set contains:

  ```
  INCLUDE COND=(44,8,CH,EQ,C'Regular')
  ```

- The TPOWCNTL data set contains:

  ```
  INCLUDE COND=(44,8,CH,EQ,C'Power')
  ```

The first COPY operator copies the records from the INPUT data set that contain ’Regular ’ in positions 44-51 to the TEMP (temporary) data set

The first DISPLAY operator uses the first subset of records in the TEMP data set to print, in the REGULAR data set:

- A title line containing the specified title and the page number; the page number is moved to the right as a result of the extra blanks at the end of the TITLE string and the 8 blanks between the title string and the page number
- A heading line containing the specified underlined headings (with no heading for the first ON field)
- Data lines for the first subset of records containing:
  - The characters from positions 1-18
  - The characters from positions 35-39
  - The floating sign values from positions 28-31 formatted with one decimal place and a period as the decimal point

The second COPY operator copies the records from the INPUT data set that contain ’Power ’ in positions 44-51 to the TEMP (temporary) data set

The second DISPLAY operator uses the second subset of records in the TEMP data set to print, in the POWER data set:

- A title line containing the specified title and the page number; the page number is moved to the right as a result of the extra blanks at the end of the TITLE string and the 8 blanks between the title string and the page number

- A heading line containing the specified underlined headings (with no heading for the first ON field)
- Data lines for the second subset of records containing:
  - The characters from positions 1-18
  - The characters from positions 35-39
  - The floating sign values from positions 28-31 formatted with one decimal place and a period as the decimal point

The REGULAR output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
Report on Regular Tools              - 1 -

                     Item    Percent Change
                     -----   --------------
Hammers              10325            -7.3
Wrenches             00273            15.8
.                    .                   .
.                    .                   .
.                    .                   .
```

The title line and underlined heading line appear at the top of each page. The number of lines per page is 66, overriding the default of 58.

The POWER output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
Report on Power Tools               - 1 -

                     Item    Percent Change
                     -----   --------------
Saws                 31730             9.8
Drills               68321           123.0
.                    .                   .
.                    .                   .
.                    .                   .
```

The title line and underlined heading line appear at the top of each page. The number of lines per page is 66, overriding the default of 58.

Example 5

```
    DISPLAY FROM(INV) LIST(RDWLIST1) -
      TITLE('No Frills RDW Report') -
      ON(NUM) -
      ON(VLEN) -
      ON(1,4,HEX) -
      MINIMUM('Smallest') -
      MAXIMUM('Largest')
```

Prints, in the RDWLIST1 data set:

- A title line containing the specified title
- A heading line containing the standard headings
- Data lines in the standard format containing:
  - The record number
  - The record length
  - The record descriptor word (RDW) in hexadecimal

- A MINIMUM line containing the specified string and the minimum record length in the standard format
- A MAXIMUM line containing the specified string and the maximum record length in the standard format.

The RDWLIST1 output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
No Frills RDW Report

RECORD NUMBER     RECORD LENGTH     (1,4,HEX)
000000000000001   +000000000000075  004B0000
000000000000002   +000000000000071  00470000
              .                  .  .
              .                  .  .
              .                  .  .

Smallest          +000000000000058

Largest           +000000000000078
```

The title line and heading line appear at the top of each page.

Example 6

```
    DISPLAY FROM(INV) LIST(RDWLIST2) -
      DATE(DMY.) -
      TITLE('  Fancy RDW Report  ') -
      TIME(12:) -
      HEADER('Relative Record') ON(NUM) -
      HEADER('    RDW (length)') ON(VLEN) -
      HEADER('RDW (Hex)') ON(1,4,HEX) -
      BLANK -
      MINIMUM('Smallest Record in Variable Data Set:') -
      MAXIMUM('Largest Record in Variable Data Set:')
```

Prints, in the RDWLIST2 data set:

- A title line containing the date, the specified title and the time
- A heading line containing the specified underlined headings
- Data lines in the BLANK format containing:
  - The record number
  - The record length
  - The record descriptor word (RDW) in hexadecimal
- A MINIMUM line containing the specified string and the minimum record length in the BLANK format
- A MAXIMUM line containing the specified string and the maximum record length in the BLANK format.

RDWLIST2 output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
21.09.92          Fancy RDW Report          01:52:28 pm

Relative Record       RDW (length)  RDW (Hex)
---------------   ----------------  ---------
              1                 75  004B0000
              2                 71  00470000
              .                  .  .
              .                  .  .
              .                  .  .

Smallest Record in Variable Data Set:
                                58

Largest Record in Variable Data Set:
                                78
```

The title line and underlined heading line appear at the top of each page.

Example 7

```
SORT FROM(PARTS) TO(TEMP) USING(SRT1)
DISPLAY FROM(TEMP) LIST(USA) -
  TITLE('Parts Completion Report for USA') DATE -
  HEADER('Part')  HEADER('Completed')  HEADER('Value ($)') -
  ON(15,6,CH)     ON(3,4,ZD,A1)        ON(38,8,ZD,C1) -
  TOTAL('Total:')
DISPLAY FROM(TEMP) LIST(FRANCE) -
  TITLE('Parts Completion Report for France') DATE(DM4/) -
  HEADER('Part')  HEADER('Completed')  HEADER('Value (F)') -
  ON(15,6,CH)     ON(3,4,ZD,A3)        ON(38,8,ZD,C3) -
  TOTAL('Total:')
DISPLAY FROM(TEMP) LIST(DENMARK) -
  TITLE('Parts Completion Report for Denmark') DATE(DMY-) -
  HEADER('Part')  HEADER('Completed')  HEADER('Value (kr)') -
  ON(15,6,CH)     ON(3,4,ZD,A2)        ON(38,8,ZD,C2) -
  TOTAL('Total:')
```

This example shows how reports for three different countries can be produced. The reports differ only in the way that date and numeric values are displayed.

Assume that the SRT1CNTL data set contains:

```
SORT FIELDS=(15,6,CH,A)
```

The SORT operator sorts the PARTS data set to the TEMP data set using the SORT statement in SRT1CNTL.

The first DISPLAY operator uses the sorted records in the TEMP data set to print, in the USA data set:

- A title line containing the specified title and the date in the format commonly used in the United States
- A heading line containing the specified underlined headings
- Data lines containing:
  - The characters from positions 15-20
  - The zoned decimal values from positions 3-6 formatted with the separators commonly used in the United States
  - The zoned decimal values from positions 38-45 formatted with two decimal places and the separators and decimal point commonly used in the United States.

- A TOTAL line containing the specified string and the total for each of the two zoned decimal fields formatted in the same way as the data values.

The second DISPLAY operator uses the sorted records in the TEMP data set to print, in the FRANCE data set:

- A title line containing the specified title and the date in the format commonly used in France
- A heading line containing the specified underlined headings
- Data lines containing:
  - The characters from positions 15-20
  - The zoned decimal values from positions 3-6 formatted with the separators commonly used in France
  - The zoned decimal values from positions 38-45 formatted with two decimal places and the separators and decimal point commonly used in France.
- A TOTAL line containing the specified string and the total for each of the two zoned decimal fields formatted in the same way as the data values.

The third DISPLAY operator uses the sorted records in the TEMP data set to print, in the DENMARK data set:

- A title line containing the specified title and the date in the format commonly used in Denmark
- A heading line containing the specified underlined headings
- Data lines containing:
  - The characters from positions 15-20
  - The zoned decimal values from positions 3-6 formatted with the separators commonly used in Denmark
  - The zoned decimal values from positions 38-45 formatted with two decimal places and the separators and decimal point commonly used in Denmark.
- A TOTAL line containing the specified string and the total for each of the two zoned decimal fields formatted in the same way as the data values.

The USA output starts on a new page and looks as follows (several records are shown with illustrative values):

```
Parts Completion Report for USA        01/14/95

Part                Completed               Value ($)
------   --------------------   ---------------------
000310                    562                8,317.53
001184                  1,234               23,456.78
029633                     35                  642.10
192199                  3,150              121,934.65
821356                    233                2,212.34

Total:                  5,214              156,563.40
```

The title line and underlined heading line appear at the top of each page.

The FRANCE output starts on a new page and looks as follows (several record are shown with illustrative values):

```
Parts Completion Report for France        14/01/1995

Part                Completed               Value (F)
------   --------------------   ---------------------
000310                    562                8 317,53
001184                  1 234               23 456,78
029633                     35                  642,10
192199                  3 150              121 934,65
821356                    233                2 212,34

Total:                  5 214              156 563,40
```

The title line and underlined heading line appear at the top of each page.

The DENMARK output starts on a new page and looks as follows (several records are shown with illustrative values):

```
Parts Completion Report for Denmark        14-01-95

Part                Completed              Value (kr)
------   --------------------   ---------------------
000310                    562                8.317,53
001184                  1.234               23.456,78
029633                     35                  642,10
192199                  3.150              121.934,65
821356                    233                2.212,34

Total:                  5.214              156.563,40
```

The title line and underlined heading line appear at the top of each page.

Example 8

```
SORT FROM(DATA) TO(TEMP) USING(SRTX)
DISPLAY FROM(TEMP) LIST(WEST) -
  DATE TITLE('Western Region Profit/Loss Report') PAGE -
  BTITLE('Division:') BREAK(3,10,CH) -
  HEADER('Branch Office') ON(16,13,CH) -
  HEADER('Profit/Loss (K)') ON(41,4,PD,/K,E1) -
  BMINIMUM('Lowest Profit/Loss in this Division:') -
  BMAXIMUM('Highest Profit/Loss in this Division:') -
  BAVERAGE('Average Profit/Loss for this Division:') -
  MINIMUM('Lowest Profit/Loss for all Divisions:') -
  MAXIMUM('Highest Profit/Loss for all Divisions:') -
  AVERAGE('Average Profit/Loss for all Divisions:')
```

This example shows how a report with sections can be produced.

Assume that the SRTXCNTL data set contains:

```
SORT FIELDS=(3,10,A,16,13,A),FORMAT=CH
```

The SORT operator sorts the DATA data set to the TEMP data set using the SORT statement in SRTXCNTL.

The DISPLAY operator uses the sorted records in the TEMP data set to print, in the WEST data set, sections with:

- A title line containing the date, the specified title string, and the page number
- A break title containing the specified break title string, and the break field characters from positions 3-12
- A heading line containing the specified underlined headings

- Data lines containing:
  - The characters from positions 16-28
  - The packed decimal values from positions 41-44 divided by 1000 and formatted with separators and signs as specified.
- Break MINIMUM, MAXIMUM, and AVERAGE lines containing the specified strings and statistics for the packed decimal field values in this section, formatted in the same way as the data values.

The last page of the report contains:

- A title line containing the date, the specified title string, and the page number
- A heading line containing the specified underlined headings
- Overall MINIMUM, MAXIMUM, and AVERAGE lines containing the specified strings and statistics for the packed decimal field values in the report, formatted in the same way as the data values.

The first section of the WEST output starts on a new page and looks as follows (several records are shown with illustrative values):

```
01/14/95        Western Region Profit/Loss Report        - 1 -

Division:  Chips

Branch Office   Profit/Loss (K)
-------------   ---------------
Gilroy                   3,293
Los Angeles               (141)
Morgan Hill                213
Oakland                  1,067
San Francisco              (31)
San Jose                    92
San Martin               1,535

Lowest Profit/Loss in this Division:
                          (141)

Highest Profit/Loss in this Division:
                         3,293

Average Profit/Loss for this Division:
                           861
```

The title line, break title line, and underlined heading line appear at the top of each page of the section.

The second section of the WEST output starts on a new page and looks as follows (several records are shown with illustrative values):

```
01/14/95        Western Region Profit/Loss Report        - 2 -

Division:  Ice Cream

Branch Office   Profit/Loss (K)
-------------   ---------------
Marin                     673
Napa                       95
San Francisco            (321)
San Jose                2,318
San Martin                 21

Lowest Profit/Loss in this Division:
                         (321)

Highest Profit/Loss in this Division:
                        2,318

Average Profit/Loss for this Division:
                           557
```

The title line, break title line, and underlined heading line appear at the top of each page of the section.

The last page of the WEST output starts on a new page and looks as follows:

```
01/15/95        Western Region Profit/Loss Report        - 3 -

Branch Office   Profit/Loss (K)
-------------   ---------------

Lowest Profit/Loss for all Divisions:
                         (321)

Highest Profit/Loss for all Divisions:
                        3,293

Average Profit/Loss for all Divisions:
                           734
```

Example 9

```
MODE CONTINUE
VERIFY FROM(CHECK) ON(2,3,PD) LIMIT(500)
DISPLAY FROM(CHECK) LIST(PDREPORT) BLANK LIMIT(500) -
  HEADER('Relative Record') ON(NUM) -
  HEADER('Numeric') ON(2,3,PD) -
  HEADER('Hexadecimal') ON(2,3,HEX) -
  HEADER('Associated Field') ON(21,20,CH)
```

This example shows how each record containing an invalid decimal value can be identified either by its relative record number or an associated field in the record.

The MODE operator ensures that the DISPLAY operator is processed if the VERIFY operator identifies an invalid decimal value.

The VERIFY operator checks for invalid digits (A-F) and invalid signs (0-9) in the packed decimal values from positions 2-4 of the CHECK data set. Message ICE618A is printed in the TOOLMSG data set for each value (if any) that contains an invalid digit or sign. If 500 invalid values are found, the operation is terminated.

The DISPLAY operator checks for invalid digits (A-F) in the packed decimal values from positions 2-4 of the CHECK data set. Message ICE618A is printed in the

TOOLMSG data set for each value (if any) that contains an invalid digit. If 500 invalid values are found, the operation is terminated. If a check for invalid signs is required, the VERIFY operator must be used, since the DISPLAY operator only checks for invalid digits. The VERIFY operator is not required if signs need not be checked.

The DISPLAY operator also prints, in the PDREPORT data set:

- A heading line containing the specified underlined headings
- Data lines in the BLANK format containing:
  - The relative record number. This number can be matched against the RECORD numbers printed in the ICE618A messages to find the records with invalid signs.
  - The numeric representation of the packed decimal value in positions 2-4. Asterisks are shown for values with invalid digits, making them easy to identify. Asterisks are not shown for values with invalid signs; these must be identified by matching the relative record number against the RECORD number in ICE618A.
  - The hexadecimal representation of the packed decimal value in positions 2-4 (also shown in ICE618A). This makes it easy to find the specific hexadecimal digits or signs that are invalid.
  - The characters in positions 21-40. An associated field such as this can be used to make identification of the records with invalid values easier.

The ICE618A messages in TOOLMSG for the VERIFY operator are:

```
ICE618A 0 INVALID (2,3,PD)       VALUE - RECORD:  000000000000003,
          HEX VALUE  53A54C
ICE618A 0 INVALID (2,3,PD)       VALUE - RECORD:  000000000000012,
          HEX VALUE  621540
ICE618A 0 INVALID (2,3,PD)       VALUE - RECORD:  000000000000019,
          HEX VALUE  400F3C
```

The ICE618A messages in TOOLMSG for the DISPLAY operator are:

```
ICE618A 0 INVALID (2,3,PD)       VALUE - RECORD:  000000000000003,
          HEX VALUE  53A54C
ICE618A 0 INVALID (2,3,PD)       VALUE - RECORD:  000000000000019,
          HEX VALUE  400F3C
```

The PDREPORT output looks as follows:

```
Relative Record   Numeric   Hexadecimal   Associated Field
---------------   -------   -----------   --------------------
              1     18600   18600C        Wagar
              2       -93   00093B        Gellai
              3    ******   53A54C        Giulianelli
              4     86399   86399C        Mehta
              5     24215   24215F        Johnson
              6      8351   08351C        Packer
              7     19003   19003C        Childers
              8    -31285   31285D        Burg
              9     88316   88316C        Monkman
             10      1860   01860C        Vezinaw
             11    -29285   29285D        Mead
             12     62154   621540        Wu
             13      -328   00328D        Madrid
             14    -11010   11010D        Warren
             15      1363   01363F        Burt
             16     92132   92132C        Mao
             17    -48500   48500D        Shen
             18       -55   00055D        Yamamoto-Smith
             19    ******   400F3C        Yaeger
             20     33218   33218C        Leung
             21     96031   96031C        Kaspar
```

PDREPORT can be used in conjunction with the ICE618A messages to identify that:

- Record 3 has an invalid digit of A and an associated field of “Giulianelli”
- Record 12 has an invalid sign of 0 and an associated field of “Wu”
- Record 19 has an invalid digit of F and an associated field of “Yaeger”.

Example 10

```
COPY FROM(IN) USING(OUTF)
DISPLAY FROM(TEMP) LIST(EMPCT) BLANK -
      TITLE('Employees by Function') -
      DATE -
      HEADER('Function') HEADER('Employees') -
      ON(1,25,CH)        ON(30,4,ZD)
```

This example shows how the OUTFIL table lookup feature can be used to substitute meaningful phrases for cryptic values in ICETOOL reports. Assume that:

- The OUTFCNTL data set contains:

```
OUTFIL FNAMES=TEMP,
  OUTREC=(1:9,2,CHANGE=(25,
                 C'MN',C'Manufacturing',
                 C'RD',C'Research and Development',
                 C'FN',C'Finance',
                 C'MR',C'Marketing',
                 C'IS',C'Information Systems'),
            30:4,4)
```

The COPY operator uses the OUTFIL statement in OUTFCNTL to reformat the IN data set records to the TEMP (temporary) data set. Two fields are extracted for use by the DISPLAY operator:

- The 2-character department code in positions 9-10 is changed to a 25-character name in positions 1-25 using the table lookup feature.
- The zoned decimal value in positions 4-7 is moved to positions 30-33.

The DISPLAY operator uses the reformatted fields in the TEMP data set to print, in the EMPCT data set:

- A title line containing the specified title and the date
- A heading line containing the specified underlined headings
- Data lines in the BLANK format containing:
  - The names from positions 1-25 that were substituted for the department codes
  - The zoned decimal values from positions 30-33.

The EMPCT output starts on a new page and looks as follows:

```
Employees by Function        02/14/95

Function                    Employees
-------------------------   ---------
Manufacturing                     486
Marketing                          21
Research and Development           55
Information Systems               123
Finance                            33
```

MODE Operator

```
►►──MODE──┬─STOP─────┬──────────────────────────────────────────────────►◄
          ├─CONTINUE─┤
          └─SCAN─────┘
```

Specifies one of three modes to control error checking and actions after error detection. A MODE operator effects the “processing” (that is, error checking of ICETOOL statements and calling DFSORT) of the operators which follow it, up to the next MODE operator (if any). MODE operators allow you to do the following for groups of operators or all operators:

1. Stop or continue processing operators after a return code of 12 or 16. A return code of 12 or 16 can be set as the result of a statement or run-time error detected by ICETOOL or DFSORT.
2. Check for errors in ICETOOL statements, but do not call DFSORT.

STOP

Stops subsequent operations if a return code of 12 or 16 is set. If an error is detected for an operator, SCAN mode is automatically set in effect; DFSORT is not called for subsequent operators, although checking ICETOOL statements for errors continues.

STOP mode can be used to group dependent operators (that is, if an operation fails, do not process the remaining operators).

STOP MODE is set in effect automatically at the start of the ICETOOL run.

CONTINUE

Continues with subsequent operations regardless of whether or not a return code of 12 or 16 is set. If an operator results in an error, processing continues for subsequent operators.

CONTINUE mode can be used to group independent operators (that is, process each operator regardless of the success or failure of the others).

SCAN

ICETOOL statements are checked for errors, but DFSORT is not called.

MODE Operator

SCAN mode can be used to test ICETOOL statements for errors.

Note: SCAN mode is set automatically if an error is detected while in STOP mode.

MODE Example

```
MODE SCAN
  RANGE ...
  UNIQUE ...
MODE STOP
  VERIFY ...
  DISPLAY ...
MODE CONTINUE
  COPY ...
  SORT ...
  STATS ...
```

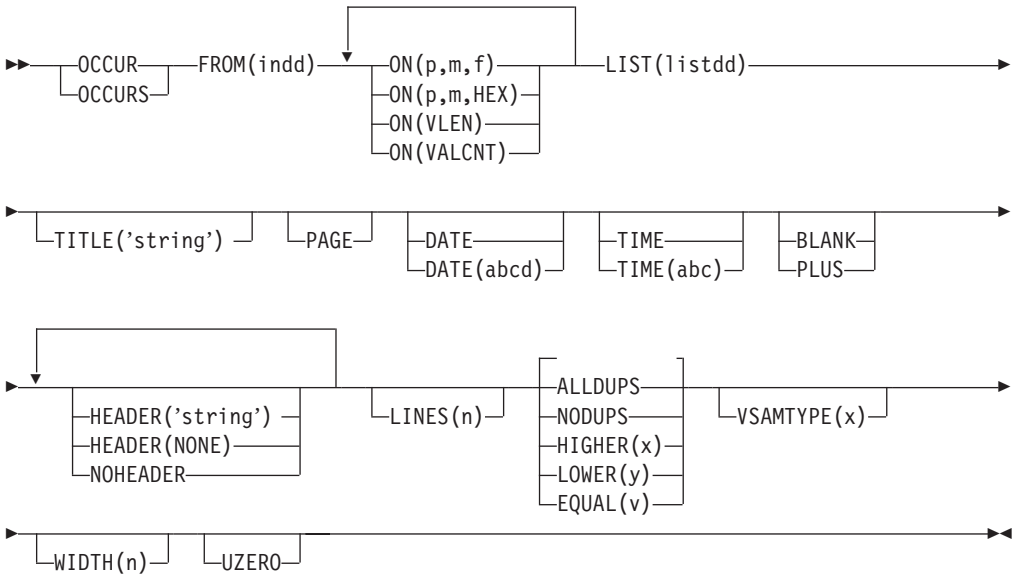
SCAN mode: RANGE and UNIQUE are checked for statement errors, but DFSORT is not called.

STOP mode: DISPLAY is dependent on VERIFY. If the return code for VERIFY is 12 or 16, SCAN mode is entered; DISPLAY is checked for statement errors, but DFSORT is not called.

CONTINUE mode: COPY, SORT, and STATS are independent of each other. SORT is processed even if the return code for COPY is 12 or 16. STATS is processed even if the return code for COPY or SORT is 12 or 16.

Note that the return codes for one group of operators does not affect the other groups of operators.

OCCUR Operator



Prints each unique value for specified numeric or character fields and how many times it occurs in a separate list data set. Simple or tailored reports can be produced. The values printed can be limited to those for which the value count meets specified criteria.

From 1 to 10 fields can be specified, but the resulting list data set line length must not exceed the limit specified by the WIDTH operand or 2048 bytes if WIDTH is not specified. At least one ON(VLEN) or ON(p,m,f) field must be specified; all such ON fields specified are used to determine whether a record contains a unique value. A single list data set record is printed for each unique value. If ON(VALCNT) is specified, the ″value count″ (that is, the number of times the ON values occur) is printed in the list data set record along with the other ON values.

Specifying the PLUS or BLANK operand, which can ″compress″ the columns of output data, can enable you to include more fields in your report, up to a maximum of 10, if your line length is limited by the character width your printer or display supports.

ALLDUPS, NODUPS, HIGHER(x), LOWER(y) or EQUAL(v) can be specified to limit the ON values printed to those for which the value count meets the specified criteria (for example, ALLDUPS for duplicate values only). The default criteria is HIGHER(0) resulting in the ON values being printed for each unique value.

DFSORT is called to sort the indd data set to ICETOOL's E35 user exit. ICETOOL uses its E35 exit to print appropriate titles, headings and data in the list data set.

You must not supply your own DFSORT MODS, INREC, OUTREC, SUM, or RECORD statement since they override the DFSORT statements passed by ICETOOL for this operator.

The DYNALLOC option is passed to DFSORT to ensure that work space is available for the sort. If your installation defaults for dynamic allocation are inappropriate for an OCCUR operator, you can take one of the following actions:

1. Override the DYNALLOC option using an OPTION control statement such as: in the DFSPARM data set (applies to all OCCUR, SELECT, SORT, and UNIQUE

   ```
   OPTION DYNALLOC=(3390,5)
   ```

   operators).
2. Use SORTWKdd DD statements to override the use of dynamic allocation (applies to all OCCUR, SELECT and UNIQUE operators). Refer to “SORTWKdd DD Statement” on page 69 for details.

Tape work data sets **cannot** be used with ICETOOL.

Simple Report

You can produce a simple report by specifying just the required operands. For example, if you specify FROM and LIST operands, and ON operands for 10-byte character and 7-byte zoned decimal fields and the value count, the output in the list data set can be represented as follows:

```
(p,m,f)                (p,m,f)            VALUE COUNT
characters             sdddddddddddddd    ddddddddddddddd
.                                    .                 .
.                                    .                 .
.                                    .                 .
```

A control character occupies the first byte of each list data set record. Left-justified standard headings are printed at the top of each page to indicate the contents of each column, followed by a line for each record showing the characters and numbers in the fields of that record, and the count of occurrences (value count) of the specified values.

The fields are printed in columns in the same order in which they are specified in the OCCUR statement. All fields are left-justified. For numeric fields, leading zeros are printed, a - is used for the minus sign, and a + is used for the plus sign. For the value count, leading zeros are printed.

Three blanks appear between columns.

The standard column widths are as follows:

- Character data: the length of the character field or 20 bytes if the field length is less than 21 bytes
- Numeric data: 16 bytes
- Value count: 15 bytes

HEADER operands can be used to change or suppress the headings. PLUS or BLANK operands can be used to change the format of numeric fields. PLUS, BLANK and HEADER operands can be used to change the width of the columns for numeric and character fields and the justification of headings and fields.

The NOHEADER operand can be used to create list data sets containing only data records. Data sets created in this way can be processed further by other operators (for example, STATS or UNIQUE) using CH format for character values or CSF/FS format for numeric values (including the value count).

Tailored Report

You can tailor the output in the list data set using various operands that control title, date, time, page number, headings, lines per page and field formats. The optional operands can be used in many different combinations to produce a wide variety of report formats. For example, if you specify FROM, LIST, BLANK, TITLE, PAGE, DATE, TIME, and HEADER operands, and ON operands for 10-byte character and 7-byte zoned decimal fields and the value count, the output in the list data set looks as follows:

```
title        - p -        mm/dd/yy        hh:mm:ss

header         header           header
----------   --------   --------------
characters         sd                d
.                   .                .
.                   .                .
.                   .                .
```

A control character occupies the first byte of each list data set record. The title line is printed at the top of each page of the list data set. It contains the elements you

specify (title string, page number, date and time) in the order in which you specify them. Eight blanks appear between title elements. A blank line is printed after the title line.

Your specified headings (underlined) are printed after the title line on each page to indicate the contents of each column, followed by a line for each record showing the characters and numbers in the fields of that record. Headings for character fields are left-justified and headings for numeric fields are right-justified.

The fields are printed in columns in the same order in which they are specified in the OCCUR statement. Character fields are left-justified and numeric fields are right justified. For numeric fields, leading zeros are suppressed, a - is used for the minus sign, and a blank is used for the plus sign (you can specify PLUS rather than BLANK if you want a + to be used for the plus sign). For the value count, leading zeros are suppressed.

Three blanks appear between columns.

The column widths are dynamically adjusted according to the length of the headings and the maximum number of bytes needed for the character or numeric data.

The operands described below can be specified in any order.

FROM(indd)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

ON(p,m,f)
Specifies the position, length, and format of a numeric or character field to be used for this operation. ’ (p,m,f)’ is used for the standard column heading (see HEADER(’string’), HEADER(NONE) and NOHEADER for alternative heading options).

p specifies the first byte of the field relative to the beginning of the input record. p is 1 for the first **data** byte of a fixed-length record and 5 for the first **data** byte of a variable-length record as illustrated below (RRRR represents the 4-byte record descriptor word):

```
   Fixed-length record     |    Variable-length record
   | D | A | T | A | ...  |    | R | R | R | R | D | A | T | A | ...
p=   1   2   3   4         | p=   1   2   3   4   5   6   7   8
```

m specifies the length of the field in bytes. A field must not extend beyond position 32752 or beyond the end of a record. The maximum length for a field depends on its format.

f specifies the format of the field as shown below.

Format Code	Length	Description
BI	1 to 4 bytes	Unsigned binary
FI	1 to 4 bytes	Signed fixed-point
PD	1 to 8 bytes	Signed packed decimal
ZD	1 to 15 bytes	Signed zoned decimal
CH	1 to 80 bytes	Character

Format Code	Length	Description
CSF or FS	1 to 16 bytes (15 digit limit)	Signed numeric with optional leading floating sign

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

For a CSF or FS format field:

- A maximum of 15 digits is allowed. If a CSF/FS value with 16 digits is found, ICETOOL issues an error message and terminates the operation.

For a ZD or PD format field:

- If a decimal value contains an invalid digit (A-F), ICETOOL identifies the bad value in a message and terminates the operation.
- F, E, C, A, 8, 6, 4, 2, and 0 are treated as equivalent positive signs. Thus the zoned decimal values F2F3C1, F2F3F1 and 020301 are counted as only one unique value.
- D, B, 9, 7, 5, 3, and 1 are treated as equivalent negative signs. Thus the zoned decimal values F2F3B0, F2F3D0, and 020310 are counted as only one unique value.

The fields of records that do not meet the specified criteria are not checked for invalid digits (PD and ZD) or excessive digits (CSF and FS).

ON(p,m,HEX)
Specifies the position and length of a character field to be used for this operation and printed in hexadecimal format (00-FF for each byte). ’ (p,m,HEX)’ is used for the standard column heading (see HEADER(’string’), HEADER(NONE), and NOHEADER for alternative heading options).

See ON(p,m,f) for a discussion of **p**.

m specifies the length of the field in bytes. A field must not extend beyond position 32752 or beyond the end of a record. A field can be 1 to 50 bytes.

ON(VLEN)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

ON(VALCNT)
Specifies that the number of occurrences for each unique value is to be printed. ’ VALUE COUNT’ is used for the standard column heading (see HEADER(’string’), HEADER(NONE) and NOHEADER for alternative heading options).

LIST(listdd)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

TITLE(’string’)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

PAGE
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

DATE
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

DATE(abcd)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

TIME
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

TIME(abc)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

BLANK
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

PLUS
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

For ON(VALCNT), PLUS is treated as BLANK.

HEADER(’string’)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

HEADER(NONE)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

NOHEADER
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

LINES(n)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

ALLDUPS
Limits the ON values printed to those that occur more than once (that is, those with duplicate field values). The ON values are printed when value count > 1.

ALLDUPS is equivalent to HIGHER(1).

NODUPS
Limits the ON values printed to those that occur only once (that is, those with no duplicate field values). The ON values are printed when value count = 1.

NODUPS is equivalent to EQUAL(1) or LOWER(2).

HIGHER(x)
Limits the ON values printed to those that occur more than x times. The ON values are printed when value count > x.

x must be specified as n or +n where n can be 1 to 15 decimal digits.

LOWER(y)
Limits the ON values printed to those that occur less than y times. The ON values are printed when value count < y.

y must be specified as n or +n where n can be 1 to 15 decimal digits.

EQUAL(v)
Limits the ON values printed to those that occur v times. The ON values are printed when value count = v.

v must be specified as n or +n where n can be 1 to 15 decimal digits.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

WIDTH(n)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

UZERO
Specifies that −0 and +0 are to be treated as unsigned zero values, that is, as the same value. With UZERO, −0 and +0 are treated as positive for ON processing.

UZERO overrides the default of treating −0 and +0 as signed zero values, that is, as different values. Without UZERO, −0 is treated as negative and +0 is treated as positive for ON processing.

OCCUR Examples

Although the OCCUR operators in the examples below could all be contained in a single ICETOOL job step, they are shown and discussed separately for clarity. See “DISPLAY Operator” on page 369 for additional examples of tailoring the report format.

Example 1

```
OCCUR FROM(SOURCE) LIST(VOLSERS) ON(40,6,CH) ON(VALCNT)
```

Prints, in the VOLSERS data set:

- A heading line containing the standard headings
- A data line for each unique ON(40,6,CH) value in the standard format containing:
  - The characters from positions 40-45 of the SOURCE data set for the unique value
  - The count of occurrences in the SOURCE data set of the unique value

The VOLSERS output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
(40,6,CH)         VALUE COUNT
ABC001            000000000000025
ABC002            000000000000011
.                               .
.                               .
.                               .
```

The heading line appears at the top of each page.

Example 2

```
OCCUR FROM(IN) LIST(LIST1) -
      TITLE('   3090 Distribution   ') -
      PAGE -
      HEADER('Data Centers')  ON(VALCNT) -
      HEADER('State') ON(1,16,CH) -
      HEADER('3090s') ON(25,3,PD) -
      BLANK
```

Prints, in the LIST1 data set:

- A title line containing the specified title and the page number
- A heading line containing the specified underlined headings
- A data line for each unique ON(1,16,CH) and ON(25,3,PD) value in the BLANK format containing:
  - The count of occurrences in the IN data set of the unique value
  - The characters from positions 1-16 of the IN data set for the unique value
  - The packed decimal values from positions 25-27 of the IN data set for the unique value

The LIST1 output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
   3090 Distribution          - 1 -

   Data Centers   State               3090s
---------------   ----------------   ------
             12   Alabama                 1
              6   Alabama                 2
              .   .                       .
              .   .                       .
              .   .                       .
```

The title line and underlined heading line appear at the top of each page.

Example 3

```
OCCURS FROM(FAILURES) LIST(CHECKIT) -
      DATE TITLE('Possible System Intruders')  PAGE -
      HEADER(' Userid ') HEADER(' Logon Failures ') -
      ON(23,8,CH)      ON(VALCNT) -
      HIGHER(4) -
      BLANK
```

Prints, in the CHECKIT data set:

- A title line containing the date, the specified title, and the page number
- A heading line containing the specified underlined headings
- A data line for each unique ON(23,8,CH) value for which there are more than 4 occurrences, in the BLANK format, containing:
  - The characters from positions 23-30 of the FAILURES data set
  - The count of occurrences of the characters from positions 23-30 of the FAILURES data set

The CHECKIT output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
10/21/92        Possible System Intruders        - 1 -

 Userid     Logon Failures
--------   ----------------
B7234510                  5
D9853267                 11
.                         .
.                         .
.                         .
```

The title line and underlined heading line appear at the top of each page.

Example 4

```
OCCUR FROM(VARIN) LIST(ONCE) -
      TITLE('Record lengths that occur only once') -
      TIME(12:) DATE(DMY.) -
      ON(VLEN) NODUPS BLANK
```

Prints, in the ONCE data set:

- A title line containing the specified title and the time and date
- A heading line containing the standard heading
- A data line for each record length for which there is only one occurrence, in the BLANK format, containing the record length

The ONCE output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
Record lengths that occur only once        09:52:17 am        21.10.92

RECORD LENGTH
           57
           61
            .
            .
            .
```

The title line and heading line appear at the top of each page.

RANGE Operator

```
►►──RANGE──FROM(indd)──┬─ON(p,m,f)─┬──┬─HIGHER(x)──────────────────────────►
                       └─ON(VLEN)──┘  ├─LOWER(y)──────────────────┤
                                      ├─HIGHER (x)──LOWER (y)──────┤
                                      ├─EQUAL(v)─────────────────┤
                                      └─NOTEQUAL(w)──────────────┘

►──┬──────────────┬──────────────────────────────────────────────────────►◄
   └─VSAMTYPE(x)──┘
```

Prints a message containing the count of values in a specified range for a specific numeric field.

DFSORT is called to copy the indd data set to ICETOOL's E35 user exit. ICETOOL prints a message containing the range count as determined by its E35 user exit.

The range can be specified as higher than x, lower than y, higher than x and lower than y, equal to v, or not equal to w, where x, y, v, and w are signed or unsigned decimal values. If the range is specified as higher than x and lower than y, it must be a valid range (for example, higher than 5 and lower than 6 is not a valid range since there is no integer value that satisfies the criteria).

You must not supply your own DFSORT MODS, INREC, or OUTREC statement since they would override the DFSORT statements passed by ICETOOL for this operator.

The operands described below can be specified in any order.

FROM(indd)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

ON(p,m,f)
Specifies the position, length, and format of the numeric field to be used for this operation.

p specifies the first byte of the field relative to the beginning of the input record. p is 1 for the first **data** byte of a fixed-length record and 5 for the first **data** byte of a variable-length record as illustrated below (RRRR represents the 4-byte record descriptor word):

```
   Fixed-length record     |    Variable-length record
   | D | A | T | A | ...  |    | R | R | R | R | D | A | T | A | ...
p=   1   2   3   4         | p=   1   2   3   4   5   6   7   8
```

m specifies the length of the field in bytes. A field must not extend beyond position 32752 or beyond the end of a record. The maximum length for a field depends on its format.

f specifies the format of the field as follows:

Format Code	Length	Description
BI	1 to 4 bytes	Unsigned binary
FI	1 to 4 bytes	Signed fixed-point
PD	1 to 8 bytes	Signed packed decimal
ZD	1 to 15 bytes	Signed zoned decimal
CSF or FS	1 to 16 bytes (15 digit limit)	Signed numeric with optional leading floating sign

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

For a CSF or FS format field:

- A maximum of 15 digits is allowed. If a CSF/FS value with 16 digits is found, ICETOOL issues an error message and terminates the operation.

For a ZD or PD format field:

- If a decimal value contains an invalid digit (A-F), ICETOOL identifies the bad value in a message and terminates the operation.
- A value is treated as positive if its sign is F, E, C, A, 8, 6, 4, 2, or 0.

- A value is treated as negative if its sign is D, B, 9, 7, 5, 3, or 1.

  For a ZD, PD or CSF/FS format field, a negative zero value is treated as a positive zero value.

ON(VLEN)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

HIGHER(x)
Values higher than x are counted as contained in the range. If only HIGHER(x) is specified, the range count is incremented when x < value. If LOWER(y) is also specified, the range count is incremented when x < value < y.

x must be specified as n, +n, or -n where n can be 1 to 15 digits.

LOWER(y)
Values lower than y are counted as contained in the range. If only LOWER(y) is specified, the range count is incremented when value < y. If HIGHER(x) is also specified, the range count is incremented when x < value < y.

y must be specified as n, +n, or -n where n can be 1 to 15 digits.

EQUAL(v)
Values equal to v are counted as contained in the range. The range count is incremented when v = value.

v must be specified as n, +n, or -n where n can be 1 to 15 decimal digits.

NOTEQUAL(w)
Values not equal to w are counted as contained in the range. The range count is incremented when w ¬= value.

w must be specified as n, +n, or -n where n can be 1 to 15 decimal digits.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

RANGE Example

```
RANGE FROM(DATA1) ON(VLEN) HIGHER(10)
RANGE FROM(DATA2) ON(11,6,ZD) LOWER(+3000)
RANGE FROM(DATA3) ON(29001,4,FI) -
       HIGHER(-10000) LOWER(27)
RANGE FROM(DATA2) ON(25,3,PD) EQUAL(-999)
RANGE FROM(DATA3) ON(40,1,BI) NOTEQUAL(199)
```

The first RANGE operator prints a message containing the count of binary values from positions 1-2 of the DATA1 data set that are higher than 10.

The second RANGE operator prints a message containing the count of zoned decimal values from positions 11-16 of the DATA2 data set that are lower than 3000.

The third RANGE operator prints a message containing the count of fixed-point values from positions 29 001-29 004 of the DATA3 data set that are higher than -10 000 but lower than 27.

The fourth RANGE operator prints a message containing the count of packed decimal values from positions 25-27 of the DATA2 data set that are equal to -999.

The fifth RANGE operator prints a message containing the count of binary values from position 40 of the DATA3 data set that are not equal to 199. This RANGE operator could be used to count the number of records that do not have 'G' in position 40, since 199 (X'C7') is the EBCDIC code for 'G'. Alternatively, the COUNT operator could be used with OMIT COND=(40,1,CH,EQ,C'G').

SELECT Operator

```
►►──SELECT──FROM(indd)──TO(outdd)──┬─┬─ON(p,m,f)─┬─┬──┬─ALLDUPS──────────────►
                                   ▼ └─ON(VLEN)──┘ │  ├─NODUPS───────────────
                                                   │  ├─HIGHER(x)────────────
                                                   │  ├─LOWER(y)─────────────
                                                   │  ├─EQUAL(v)─────────────
                                                   │  ├─FIRST────────────────
                                                   │  └─LAST─────────────────
►──┬─VSAMTYPE(x)─┬──┬─DISCARD(savedd)─┬──┬─UZERO─┬──────────────────────────►◄
   └─────────────┘  └─────────────────┘  └───────┘
```

Selects records from an input data set for inclusion in an output data set based on meeting criteria for the number of times specified numeric or character field values occur. This makes it possible to only keep records with duplicate field values, only keep records with no duplicate field values, only keep records with field values that occur more than, less than, or exactly n times, or only keep the first or last record with each unique field value. From 1 to 10 fields can be specified. At least one ON(VLEN) or ON(p,m,f) field must be specified; all such ON fields specified will be used to determine the ″value count″ (that is, the number of times the ON values occur) to be matched against the criteria.

DISCARD(savedd) can be used to save the records which do not meet the criteria (that is, the discarded records), in the savedd data set.

DFSORT is called to sort the indd data set. If the DISCARD operand is not specified, ICETOOL uses its E35 exit to determine which records to include in the outdd data set. If the DISCARD operand is specified, ICETOOL uses its E35 exit to determine whether to include each record in the outdd data set or in the savedd data set.

ICETOOL requires extra storage for SELECT processing, over and above what is normally needed by ICETOOL and DFSORT, in order to save your records until it can determine whether or not they meet your specified criteria. In most cases, only a small amount of storage is needed and can be obtained (above 16MB virtual). However, for a FROM data set with a large record length and criteria requiring many saved records, a large amount of storage is needed. For example, with a record length of 32756 and HIGHER(99), over 3 MBs of storage is needed. If ICETOOL cannot get the storage it needs, it issues a message and terminates the SELECT operation. Increasing the REGION by the amount indicated in the message may allow ICETOOL to run successfully.

The DYNALLOC option is passed to DFSORT to ensure that work space is available for the sort. If your installation defaults for dynamic allocation are inappropriate for a SELECT operator, you can take one of the following actions:

1. Override the DYNALLOC option using an OPTION control statement such as: in the DFSPARM data set (applies to all OCCUR, SELECT, SORT, and UNIQUE

```
OPTION DYNALLOC=(3390,5)
```

   operators).
2. Use SORTWKdd DD statements to override the use of dynamic allocation (applies to all OCCUR, SELECT and UNIQUE operators). Refer to “SORTWKdd DD Statement” on page 69 for details.

Tape work data sets **cannot** be used with ICETOOL.

You must not supply your own DFSORT MODS, INREC, OUTREC or OUTFIL statement since they would override the DFSORT statements passed by ICETOOL for this operator.

The operands described below can be specified in any order.

FROM(indd)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

TO(outdd)
Specifies the ddname of the output data set to which DFSORT will write the records it selects for this operation (that is, the records that meet the specified criteria). Thus, the outdd data set will contain the records selected by ALLDUPS, NODUPS, HIGHER(x), LOWER(y), EQUAL(v), FIRST or LAST.

An outdd DD statement must be present and must define an output data set that conforms to the rules for DFSORT’s SORTOUT data set (if the DISCARD operand is not specified) or OUTFIL data set (if the DISCARD operand is specified).

The ddname specified in the FROM operand must not be the same as the ddname specified in the TO operand.

Refer to “JCL Restrictions” on page 359 for more information.

ON(p,m,f)
Specifies the position, length, and format of a numeric or character field to be used for this operation.

p specifies the first byte of the field relative to the beginning of the input record. p is 1 for the first **data** byte of a fixed-length record and 5 for the first **data** byte of a variable-length record as illustrated below (RRRR represents the 4-byte record descriptor word):

```
   Fixed-length record     |    Variable-length record
   | D | A | T | A | ...  |    | R | R | R | R | D | A | T | A | ...
p=   1   2   3   4    | p=   1   2   3   4   5   6   7   8
```

m specifies the length of the field in bytes. A field must not extend beyond position 4 088, or beyond the end of a record. The maximum length for a field depends on its format.

f specifies the format of the field as shown below.

Format Code	Length	Description
BI	1 to 4 bytes	Unsigned binary
FI	1 to 4 bytes	Signed fixed-point
PD	1 to 8 bytes	Signed packed decimal
ZD	1 to 15 bytes	Signed zoned decimal
CH	1 to 80 bytes	Character
CSF or FS	1 to 16 bytes	Signed numeric with optional leading floating sign

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

For a ZD or PD format field:

- F, E, C, A, 8, 6, 4, 2, and 0 are treated as equivalent positive signs. Thus the zoned decimal values F2F3C1, F2F3F1 and 020301 are counted as only one unique value.
- D, B, 9, 7, 5, 3, and 1 are treated as equivalent negative signs. Thus the zoned decimal values F2F3B0, F2F3D0, and 020310 are counted as only one unique value.
- Digits are not checked for validity.

ON(VLEN)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

ALLDUPS
Limits the records selected to those with ON values that occur more than once (value count > 1). You can use this operand to keep just those records with duplicate field values.

ALLDUPS is equivalent to HIGHER(1).

NODUPS
Limits the records selected to those with ON values that occur only once (value count = 1). You can use this operand to keep just those records with no duplicate field values.

NODUPS is equivalent to EQUAL(1) or LOWER(2).

HIGHER(x)
Limits the records selected to those with ON values that occur more than x times (value count > x). You can use this operand to keep just those records with field values that occur more than x times.

x must be specified as n or +n where n can be 0 to 99.

LOWER(y)
Limits the records selected to those with ON values that occur less than y times (value count < y). You can use this operand to keep just those records with field values that occur less than y times.

y must be specified as n or +n where n can be 0 to 99.

EQUAL(v)
Limits the records selected to those with ON values that occur v times (value count = v). You can use this operand to keep just those records with field values that occur v times.

v must be specified as n or +n where n can be 0 to 99.

FIRST
Limits the records selected to those with ON values that occur only once (value count = 1) and the first record of those with ON values that occur more than once (value count > 1). You can use this operand to keep just the first record for each unique field value.

LAST
Limits the records selected to those with ON values that occur only once (value count = 1) and the last record of those with ON values that occur more than once (value count > 1). You can use this operand to keep just the last record for each unique field value.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

DISCARD(savedd)
Specifies the ddname of the output data set to which DFSORT will write the records it does not select for this operation (that is, the records that do not meet the specified criteria). Thus, the savedd data set will contain the records discarded by ALLDUPS, NODUPS, HIGHER(x), LOWER(y), EQUAL(v), FIRST or LAST.

A savedd DD statement must be present and must define an output data set that conforms to the rules for DFSORT’s OUTFIL data set.

The ddname specified in the DISCARD operand must not be the same as the ddname specified in the FROM or TO operand.

Refer to “JCL Restrictions” on page 359 for more information.

UZERO
See the discussion of this operand on the OCCUR statement in “OCCUR Operator” on page 398.

SELECT Examples

Although the SELECT operators in the examples below could all be contained in a single ICETOOL job step, they are shown and discussed separately for clarity.

Example 1

```
SELECT FROM(INPUT) TO(DUPS) ON(11,8,CH) ON(30,44,CH) ALLDUPS
```

Sorts the INPUT data set to the DUPS data set, selecting only the records from INPUT with characters in positions 11-18 and characters in positions 30-73 that occur more than once (that is, only records with duplicate ON field values).

The DUPS data set might look as follows (several records are shown for illustrative purposes):

```
USR002    EISSLER      12    DOC.EXAMPLES
DFSRT2    EISSLER       5    DOC.EXAMPLES
DFSRT5    MADRID       20    MYDATA
DFSRT1    MADRID       20    MYDATA
SYS003    MADRID       20    MYDATA
DFSRT2    MADRID       20    SORTST1.TEST
USR003    MADRID       20    SORTST1.TEST
.         .            .     .
.         .            .     .
.         .            .     .
```

Example 2

```
SELECT FROM(INPUT) TO(ONLYONE) ON(23,3,FS) NODUPS
```

Sorts the INPUT data set to the ONLYONE data set, selecting only the records from INPUT with floating sign values in positions 23-25 that occur just once (that is, only records with no duplicate ON field values).

The ONLYONE data set might look as follows (several records are shown for illustrative purposes):

```
DFSRT2    EISSLER       5    DOC.EXAMPLES
DFSRT1    PACKER        8    ICETOOL.SMF.RUNS
USR002    EISSLER      12    DOC.EXAMPLES
SYS003    YAEGER       32    ICETOOL.TEST.CASES
DFSRT2    MCNEILL     108    FS.TEST.CASES
.         .           .      .
.         .           .      .
.         .           .      .
```

Example 3

```
SELECT FROM(FAILURES) TO(CHECKOUT) ON(28,8,CH) ON(1,5,CH) -
  HIGHER(3)
```

Sorts the FAILURES data set to the CHECKOUT data set, selecting only the records from FAILURES with characters in positions 28-35 and characters in positions 1-5 that occur more than three times (that is only records with four or more duplicate ON field values).

The CHECKOUT data set might look as follows (several records are shown for illustrative purposes):

```
03/12/91   08:36:59        A3275647
03/12/91   09:27:32        A3275647
03/12/91   09:03:18        A3275647
03/12/91   08:56:13        A3275647
03/06/91   15:12:01        C3275647
03/06/91   14:57:00        C3275647
03/06/91   15:43:19        C3275647
03/06/91   16:06:39        C3275647
03/06/91   15:22:08        C3275647
.          .               .
.          .               .
.          .               .
```

Example 4

```
SELECT FROM(BOOKS) TO(PUBLISHR) ON(29,10,CH) FIRST
```

Sorts the BOOKS data set to the PUBLISHR data set, selecting only the records from BOOKS with characters in positions 29-38 that occur only once and the first record of those with characters in positions 29-38 that occur more than once (that is, one record for each unique ON field value).

The PUBLISHR data set might look as follows (several records are shown for illustrative purposes):

```
Banana Slugs I Have Known   Brent      Animals
Toads on Parade             Cooper     Animals
Pets Around the World       Davis      Animals
.                           .          .
.                           .          .
.                           .          .
```

Example 5

```
SELECT FROM(BOOKS) TO(PUBLISHR) ON(29,10,CH) FIRST -
  DISCARD(SAVEREST)
```

This example creates the same PUBLISHR data set as Example 4. In addition, it creates a SAVEREST data set which contains all of the records not written to the PUBLISHR data set. The SAVEREST data set might look as follows (several records are shown for illustrative purposes):

```
How to Talk to Your Amoeba  Brent    Animals
What Buzzards Want          Davis    Animals
Birds of Costa Rica         Davis    Animals
.
.
.
```

SORT Operator

```
►►─SORT─FROM(indd)─USING(xxxx)─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬─►
                               │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │
```

INCLUDE or OMIT operand is specified in the xxxxCNTL data set, the active locale's collating rules affect INCLUDE and OMIT processing as explained in the "Cultural Environment Considerations" discussion in "INCLUDE Control Statement" on page 93.

The DYNALLOC option is passed to DFSORT to ensure that work space is available for the sort. If your installation defaults for dynamic allocation are inappropriate for a SORT operator, you can take one of the following actions:

1. Override the DYNALLOC option using an OPTION control statement such as:

   ```
   OPTION DYNALLOC=(3390,5)
   ```

   in the xxxxCNTL data set (only applies to this operator) or the DFSPARM data set (applies to all OCCUR, SELECT, SORT and UNIQUE operators).
2. Use xxxxWKn DD statements to override the use of dynamic allocation only applies to this operator). Refer to "SORTWKdd DD Statement" on page 69 for details.

Tape work data sets **cannot** be used with ICETOOL.

The operands described below can be specified in any order.

FROM(indd)
See the discussion of this operand on the COPY statement in "COPY Operator" on page 360.

USING(xxxx)
Specifies the first 4 characters of the ddname for the control statement data set to be used by DFSORT for this operation. xxxx must be four characters which are valid in a ddname of the form xxxxCNTL. xxxx must not be SYSx.

An xxxxCNTL DD statement must be present, and the control statements in it must conform to the rules for DFSORT's SORTCNTL data set.

The xxxxCNTL data set must contain a SORT statement. If TO is not specified, the xxxxCNTL data set must also contain either one or more OUTFIL statements or a MODS statement for an E35 routine that disposes of all records. Other statements are optional.

Refer to "JCL Restrictions" on page 359 for more information regarding the selection of ddnames.

TO(outdd,...)
Specifies the ddnames of the output data sets to be written by DFSORT for this operation. From 1 to 10 outdd names can be specified. An outdd DD statement must be present for each outdd name specified. If a single outdd data set is specified, DFSORT is called once to sort the indd data set to the outdd data set using SORTOUT processing; the outdd data set must conform to the rules for DFSORT's SORTOUT data set. If multiple outdd data sets are specified and SERIAL is not specified, DFSORT is called once to sort the indd data set to the outdd data sets using OUTFIL processing; the outdd data sets must conform to the rules for DFSORT's OUTFIL data sets.

A ddname specified in the FROM operand must not also be specified in the TO operand.

Refer to “JCL Restrictions” on page 359 for more information regarding the selection of ddnames.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

LOCALE(name)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

LOCALE(CURRENT)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

LOCALE(NONE)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

SERIAL
Specifies that OUTFIL processing is not to be used when multiple outdd data sets are specified. DFSORT is called multiple times and uses SORTOUT processing; the outdd data sets must conform to the rules for DFSORT’s SORTOUT data set. SERIAL is not recommended because the use of serial processing (that is, multiple calls to DFSORT) instead of OUTFIL processing can degrade performance and imposes certain restrictions as detailed below. SERIAL is ignored if a single outdd data set is specified.

DFSORT is called to sort the indd data set to the first outdd data set using the DFSORT control statements in the xxxxCNTL data set. If the sort operation is successful, DFSORT is called as many times as necessary to copy the first outdd data set to the second and subsequent outdd data sets. Therefore, for maximum efficiency, use a DASD data set as the first in a list of outdd data sets on both DASD and tape. If more than one outdd data set is specified, DFSORT must be able to read the *first* outdd data set after it is written in order to copy it to the other outdd data sets. Do not use a SYSOUT or DUMMY data set as the first in a list of outdd data sets because:

- If the first data set is SYSOUT, DFSORT abends when it tries to copy the SYSOUT data set to the second outdd data set.
- If the first data set is DUMMY, DFSORT copies the empty DUMMY data set to the other outdd data sets (that is, all of the resulting outdd data sets are empty).

SORT Examples

Although the SORT operators in the examples below could all be contained in a single ICETOOL job step, they are shown and discussed separately for clarity.

Example 1

```
* Method 1
SORT FROM(MASTER) TO(PRINT,TAPE,DASD) USING(ABCD)
```

```
* Method 2
SORT FROM(MASTER) TO(DASD,TAPE,PRINT) USING(ABCD) SERIAL
```

This example shows two different methods for creating multiple sorted output data sets. Assume that the ABCDCNTL data set contains:

```
SORT FIELDS=(15,20,CH,A,1,5,PD,D)
```

Method 1 requires one call to DFSORT, one pass over the input data set, and allows the output data sets to be specified in any order. The SORT operator sorts all records from the MASTER data set to the PRINT (SYSOUT), TAPE, and DASD data sets, using the SORT statement in the ABCDCNTL data set and OUTFIL processing.

Method 2 requires three calls to DFSORT, three passes over the input data set, and imposes the restriction that the SYSOUT data set must not be the first TO data set. The SORT operator sorts all records from the MASTER data set to the DASD data set, using the SORT statement in the ABCDCNTL data set, and then copies the resulting DASD data set to the TAPE and PRINT (SYSOUT) data sets. Since the first TO data set is processed three times (written, read, read), placing the DASD data set first is more efficient than placing the TAPE data set first. PRINT must not be the first in the TO list because a SYSOUT data set cannot be read.

Example 2

```
* Method 1
SORT FROM(IN) TO(DEPT1) USING(DPT1)
SORT FROM(IN) TO(DEPT2) USING(DPT2)
SORT FROM(IN) TO(DEPT3) USING(DPT3)
```

```
* Method 2
SORT FROM(IN) USING(ALL3)
```

This example shows two different methods for creating sorted subsets of an input data set. Assume that:

- The DPT1CNTL data set contains:

  ```
  SORT FIELDS=(51,2,BI,A,18,5,CH,A,58,4,BI,A)
  INCLUDE COND=(5,3,CH,EQ,C'D01')
  ```

- The DPT2CNTL data set contains:

  ```
  SORT FIELDS=(51,2,BI,A,18,5,CH,A,58,4,BI,A)
  INCLUDE COND=(5,3,CH,EQ,C'D02')
  ```

- The DPT3CNTL data set contains:

  ```
  SORT FIELDS=(51,2,BI,A,18,5,CH,A,58,4,BI,A)
  INCLUDE COND=(5,3,CH,EQ,C'D03')
  ```

- The ALL3CNTL data set contains:

  ```
  SORT FIELDS=(51,2,BI,A,18,5,CH,A,58,4,BI,A)
  OUTFIL FNAMES=DEPT1,INCLUDE=(5,3,CH,EQ,C'D01')
  OUTFIL FNAMES=DEPT2,INCLUDE=(5,3,CH,EQ,C'D02')
  OUTFIL FNAMES=DEPT3,INCLUDE=(5,3,CH,EQ,C'D03')
  ```

Method 1 requires three calls to DFSORT and three passes over the input data set:

- The first SORT operator sorts the records from the IN data set that contain D01 in positions 5-7 to the DEPT1 data set
- The second COPY operator sorts the records from the IN data set that contain D02 in positions 5-7 to the DEPT2 data set
- The third COPY operator sorts the records from the IN data set that contain D03 in positions 5-7 to the DEPT3 data set.

Method 2 accomplishes the same result as method 1 but, because it uses OUTFIL statements instead of TO operands, requires only one call to DFSORT and one pass over the input data set.

Example 3

```
SORT FROM(IN1) TO(FRANCE) USING(SRT1) LOCALE(FR_FR)
SORT FROM(IN1) TO(CANADA) USING(SRT1) LOCALE(FR_CA)
SORT FROM(IN1) TO(BELGIUM) USING(SRT1) LOCALE(FR_BE)
```

This example shows how sorted data for three different countries can be produced. Assume that the SRT1CNTL data set contains:

```
SORT FIELDS=(5,20,CH,A,31,15,CH,A,1,4,FI,D,63,10,CH,D)
```

The first SORT operator sorts all records from the IN1 data set to the FRANCE data set, using the SORT statement in the SRT1CNTL data set. The character (CH) control fields are sorted according to the collating rules defined in locale FR_FR (French language for France).

The second SORT operator sorts all records from the IN1 data set to the CANADA data set, using the SORT statement in the SRT1CNTL data set. The character (CH) control fields are sorted according to the collating rules defined in locale FR_CA (French language for Canada).

The third SORT operator sorts all records from the IN1 data set to the BELGIUM data set, using the SORT statement in the SRT1CNTL data set. The character (CH) control fields are sorted according to the collating rules defined in locale FR_BE (French language for Belgium).

STATS Operator

```
►►──STATS──FROM(indd)──┬──┬─ON(p,m,f)─┬──┬──────────────┬──────────────────────►◄
                       ▲  └─ON(VLEN)──┘  └─VSAMTYPE(x)──┘
```

Prints messages containing the minimum, maximum, average, and total for specified numeric fields. From 1 to 10 fields can be specified.

DFSORT is called to copy the indd data set to ICETOOL's E35 user exit. ICETOOL prints messages containing the minimum, maximum, average, and total for each field as determined by its E35 exit.

The average (or mean) is calculated by dividing the total by the record count and rounding down to the nearest integer (examples: 23 / 5 = 4, -23 / 5 = - 4).

You must not supply your own DFSORT MODS, INREC, or OUTREC statement since they would override the DFSORT statements passed by ICETOOL for this operator.

The operands described below can be specified in any order.

FROM(indd)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

ON(p,m,f)
Specifies the position, length, and format of a numeric field to be used for this operation.

p specifies the first byte of the field relative to the beginning of the input record. p is 1 for the first **data** byte of a fixed-length record and 5 for the first **data** byte of a variable-length record as illustrated below (RRRR represents the 4-byte record descriptor word):

```
   Fixed-length record     |    Variable-length record
   | D | A | T | A | ...  |    | R | R | R | R | D | A | T | A | ...
p=   1   2   3   4         | p=   1   2   3   4   5   6   7   8
```

m specifies the length of the field in bytes. A field must not extend beyond position 32752 or beyond the end of a record. The maximum length for a field depends on its format.

f specifies the format of the field as follows:

Format Code	Length	Description
BI	1 to 4 bytes	Unsigned binary
FI	1 to 4 bytes	Signed fixed-point
PD	1 to 8 bytes	Signed packed decimal
ZD	1 to 15 bytes	Signed zoned decimal
CSF or FS	1 to 16 bytes	Signed numeric with optional leading floating sign

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

If the total for a field overflows, ICETOOL continues processing, but prints asterisks for the average and total for that field.

For a CSF or FS format field:

- A maximum of 15 digits is allowed. If a CSF/FS value with 16 digits is found, ICETOOL issues an error message and terminates the operation.

For a ZD or PD format field:

- If a decimal value contains an invalid digit (A-F), ICETOOL identifies the bad value in a message and prints asterisks for the minimum, maximum, average and total for that field.
- A value is treated as positive if its sign is F, E, C, A, 8, 6, 4, 2, or 0.
- A value is treated as negative if its sign is D, B, 9, 7, 5, 3, or 1.

For a ZD, PD or CSF/FS format field, a negative zero value is treated as a positive zero value.

ON(VLEN)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

STATS Example

```
STATS  FROM(DATA1)  ON(VLEN)  ON(15,4,ZD)
```

Prints messages containing the minimum, maximum, average and total of the binary values in positions 1-2 of the DATA1 data set. For variable-length records, this gives statistics about the length of the records. Prints messages containing the minimum, maximum, average and total of the zoned decimal values in positions 15-18 of the DATA1 data set.

UNIQUE Operator

```
►►─UNIQUE─FROM(indd)──┬─ON(p,m,f)─┬──┬──────────────┬──┬─────────┬──────────►◄
                      └─ON(VLEN)──┘  └─VSAMTYPE(x)──┘  └─UZERO──┘
```

Prints a message containing the count of unique values for a specified numeric or character field.

DFSORT is called to sort the indd data set to ICETOOL’s E35 user exit. ICETOOL prints a message containing the unique count as determined by its E35 user exit.

The DYNALLOC option is passed to DFSORT to ensure that work space is available for the sort. If your installation defaults for dynamic allocation are inappropriate for a UNIQUE operator, you can take one of the following actions:

1. Override the DYNALLOC option using an OPTION control statement such as: in the DFSPARM data set (applies to all OCCUR, SELECT, SORT and UNIQUE
   ```
   OPTION DYNALLOC=(3390,5)
   ```
   operators).
2. Use SORTWKdd DD statements to override the use of dynamic allocation (applies to all OCCUR, SELECT, and UNIQUE operators). Refer to “SORTWKdd DD Statement” on page 69 for details.

Tape work data sets **cannot** be used with ICETOOL.

You must not supply your own DFSORT MODS, INREC, OUTREC, SUM or RECORD statement since they override the DFSORT statements passed by ICETOOL for this operator.

The operands described below can be specified in any order.

FROM(indd)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

ON(p,m,f)
Specifies the position, length, and format of a numeric or character field to be used with this operation.

p specifies the first byte of the field relative to the beginning of the input record. p is 1 for the first **data** byte of a fixed-length record and 5 for the first **data** byte of a variable-length record as illustrated below (RRRR represents the 4-byte record descriptor word):

```
   Fixed-length record     |    Variable-length record
   | D | A | T | A | ...  |    | R | R | R | R | D | A | T | A | ...
p=   1   2   3   4         | p=   1   2   3   4   5   6   7   8
```

m specifies the length of the field in bytes. A field must not extend beyond position 32752 or beyond the end of a record. The maximum length for a field depends on its format.

f specifies the format of the field as shown below:

Format Code	Length	Description
BI	1 to 256 bytes	Unsigned binary
FI	1 to 256 bytes	Signed fixed-point
PD	1 to 32 bytes	Signed packed decimal
ZD	1 to 32 bytes	Signed zoned decimal
CH	1 to 256 bytes	Character
CSF or FS	1 to 16 bytes	Signed numeric with optional leading floating sign

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

For a ZD or PD format field:

- F, E, C, A, 8, 6, 4, 2, and 0 are treated as equivalent positive signs. Thus the zoned decimal values F2F3C1, F2F3F1, and 020301 are counted as only one unique value.
- D, B, 9, 7, 5, 3, and 1 are treated as equivalent negative signs. Thus the zoned decimal values F2F3B0, F2F3D0, and 020310 are counted as only one unique value.
- Digits are not checked for validity.

ON(VLEN)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

UZERO
See the discussion of this operand on the OCCUR statement on “OCCUR Operator” on page 398.

UNIQUE Example

```
UNIQUE FROM(DATAIN)  ON(20,40,CH)
UNIQUE FROM(DATAIN)  ON(5,3,ZD)
```

UNIQUE Operator

The first UNIQUE operator prints a message containing the count of unique character data in positions 20-59 of the DATAIN data set.

The second UNIQUE operator prints a message containing the count of unique zoned decimal values in positions 5-7 of the DATAIN data set.

VERIFY Operator

```
►►─VERIFY─FROM(indd)──┬──ON(p,m,f)──┬──────────┬──────────┬──────────────────────►
                      ▼             │          │          │
                                    └─NOSIGN─┘ └─LIMIT(n)─┘
►──┬──────────────┬──────────────────────────────────────────────────────────────►◄
   └─VSAMTYPE(x)─┘
```

Examines particular decimal fields in a data set and prints a message identifying each invalid value found for each field. From 1 to 10 fields can be specified.

DFSORT is called to copy the indd data set to ICETOOL's E35 user exit. ICETOOL uses its E35 user exit to examine the digits and sign of each value for validity, and for each invalid value found, prints an error message containing the record number and field value (in hexadecimal).

You must not supply your own DFSORT MODS, INREC, or OUTREC statement since they would override the DFSORT statements passed by ICETOOL for this operator.

Notes:

1. Values with invalid decimal digits are also identified for the DISPLAY, OCCUR, RANGE, and STATS operators.
2. The DISPLAY operator can be used to print a report identifying the relative record number, hexadecimal value, and associated fields for each invalid (and valid) decimal value, as shown in Example 9 under "DISPLAY Operator" on page 369.

The operands described below can be specified in any order.

FROM(indd)
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" on page 369.

ON(p,m,f)
Specifies the position, length, and format of a decimal field to be used for this operation.

p specifies the first byte of the field relative to the beginning of the input record. p is 1 for the first **data** byte of a fixed-length record and 5 for the first **data** byte of a variable-length record as illustrated below (RRRR represents the 4-byte record descriptor word):

```
   Fixed-length record     |    Variable-length record
   | D | A | T | A | ...  |    | R | R | R | R | D | A | T | A | ...
p=   1   2   3   4         | p=   1   2   3   4   5   6   7   8
```

m specifies the length of the field in bytes. A field must not extend beyond position 32752 or beyond the end of a record. The maximum length for a field depends on its format.

f specifies the format of the field as shown below:

Format Code	Length	Description
PD	1 to 16 bytes	Signed packed decimal
ZD	1 to 18 bytes	Signed zoned decimal

Note: See “Appendix C. Data Format Descriptions” on page 581 for detailed format descriptions.

A value is considered invalid under one of the following circumstances:

- it contains A-F as a digit (examples: a PD field of 00AF or a ZD field of F2FB)
- it contains 0-9 as a sign and the NOSIGN operand is not specified (examples: a PD field of 3218 or a ZD field of F235).

If the number of bad values reaches the LIMIT for invalid decimal values, ICETOOL terminates the operation. If the LIMIT operand is not specified, a default of 200 is used for the invalid decimal value limit.

NOSIGN
Specifies that the sign of the decimal values is not to be validity checked (overriding the default of checking for 0-9 as invalid signs).

LIMIT(n)
See the discussion of this operand on the DISPLAY statement in “DISPLAY Operator” on page 369.

VSAMTYPE(x)
See the discussion of this operand on the COPY statement in “COPY Operator” on page 360.

VERIFY Example

```
VERIFY FROM(NEW) ON(22,16,PD) ON(7,9,ZD)
VERIFY FROM(NEW) ON(22,16,PD) ON(7,9,ZD) NOSIGN LIMIT(10)
```

The first VERIFY operator checks for invalid digits (A-F) and invalid signs (0-9) in the packed decimal values from positions 22-37 and the zoned decimal values from positions 7-15 of the NEW data set. A message is printed identifying each value (if any) that contains an invalid digit or sign. If 200 invalid values are found, the operation is terminated.

The second VERIFY operator checks for invalid digits (A-F) in the packed decimal values from positions 22-37 and the zoned decimal values from positions 7-15 of the NEW data set. A message is printed identifying each value (if any) that contains an invalid digit. If 10 invalid values are found, the operation is terminated.

Note: The DISPLAY operator can be used to print a report identifying the relative record number, hexadecimal value, and associated fields for each invalid (and valid) decimal value, as shown in Example 9 under “DISPLAY Operator” on page 369.

Calling ICETOOL from a Program

ICETOOL can be called from an assembler program using the LINK, ATTACH, or XCTL system macros. Standard linkage conventions must be used. When ICETOOL finishes processing, it returns to the calling program with register 15 set to the highest operation return code encountered. See “ICETOOL Return Codes” on page 431 for an explanation of the ICETOOL return codes.

When you call ICETOOL from a program, you have a choice of two different interfaces: the TOOLIN Interface and the Parameter List Interface.

TOOLIN Interface

With the TOOLIN Interface, you supply ICETOOL statements in the TOOLIN data set. ICETOOL prints messages in the TOOLMSG data set, but does not return information directly to your program.

To use the TOOLIN interface, set a value of 0 in register 1, or place the address of a 4-byte field containing X’80000000’ in register 1, before calling ICETOOL as shown below:

```
         ...
         SLR   R1,R1            TOOLIN INTERFACE - METHOD 1
         LINK  EP=ICETOOL       CALL ICETOOL
         ...
         LA    R1,NOPLIST       TOOLIN INTERFACE - METHOD 2
         LINK  EP=ICETOOL       CALL ICETOOL
         ...
NOPLIST  DC    X'80',AL3(0)     TOOLIN INTERFACE INDICATOR
         ...
```

Parameter List Interface

The Parameter List Interface allows you to use the information derived by ICETOOL in your program. With this interface, you supply ICETOOL statements in a parameter list. ICETOOL prints messages in the TOOLMSG data set, and puts an operation status indicator and “operation-specific values” in the parameter list for use by your calling program.

Figure 40 on page 425 shows the format of the parameter list used with the Parameter List Interface. Table 55 on page 427 shows the operation-specific values returned to the calling program.

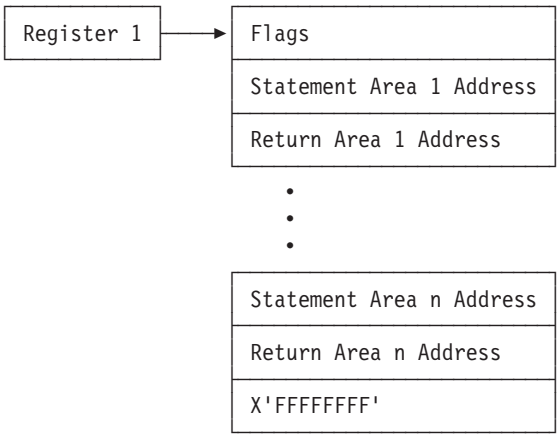


Figure 40. Parameter List for Parameter List Interface

The flags field 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 after any pair of statement/return addresses.

All addresses in the parameter list must be 31-bit addresses or clean 24-bit addresses (the first 8 bits contain zeros).

Explanation of Fields

Flags

Bit 0 = 0:
Use the Parameter List Interface. Process ICETOOL statements from this parameter list and return information to this parameter list. Ignore TOOLIN.

Bit 0 = 1:
Use the TOOLIN Interface. Process ICETOOL statements from TOOLIN. Ignore this parameter list.

Bits 1-31:
Reserved. Must be set to zero to ensure future extendibility.

Statement Area Address and Statement Area

Each statement area address gives the location of a statement area which describes an ICETOOL operation to be performed. If the statement area address is 0, ICETOOL ignores this statement area/return area pair. Otherwise, the statement area address must point to a statement area in the following format:

- A 2-byte length field containing the length of the statement area for this operation. If this field is 0, ICETOOL ignores this statement area/return area pair.
- One or more 80-character ICETOOL statement images in the format described in “ICETOOL Statements” on page 359. Each statement area must have **one** operator statement. Comment and blank statements before the operator statement are processed. Comment, blank, and additional operator statements after the end of the first operator statement are ignored.

Return Area Address and Return Area

Each return area address gives the location of a return area in which ICETOOL is to return operation-specific information for the operation described in the corresponding statement area. If the return area address is 0, ICETOOL does not return any information for this operation. Otherwise, the return area address must point to a return area in the following general format:

- A 2-byte length field containing the length of the return area for this operation. If this field is 0, ICETOOL does not return any information for this operation.
- A 1-byte operation status indicator which is set by ICETOOL as follows:

  0 = This operation was run and completed with return code 0 or 4. Operation-specific values (see below) were returned.

  4 = This operation was not run (for example, scan mode was in effect) or did not complete with return code 0 or 4. Operation-specific values (see below) were not returned.
- Operation-specific values. Each value returned by ICETOOL is an 8-bytepacked decimal value with a C for a positive sign or a D for a negative sign. If ICETOOL set the operation status to 4, it did not return any values for this operation.

Note: Programs in LPALIB which call ICETOOL must provide return areas which ICETOOL can store into.

The required return area length and the operation-specific values returned for each operator are shown in Table 55 on page 427. If the return area length is less than the length required, ICETOOL issues a message and terminates the operation.

Table 55. Return Area Lengths/Operation-Specific Values

Operator	Return Area Length (Bytes)	Operation-Specific Values Returned
COPY	1	None
COUNT	9	Count of records processed
DEFAULTS	1	None
DISPLAY	9	Count of records processed
MODE	1	None
OCCUR	17	Count of records processed, count of records resulting from criteria
RANGE	17	Count of records processed, count of values in range
SELECT	17	Count of records processed, count of records resulting from criteria
SORT	1	None
STATS	32*n+9	Count of records processed, minimum for ON field 1, maximum for ON field 1, average for ON field 1, total for ON field 1, ... minimum for ON field n, maximum for ON field n, average for ON field n, total for ON field n
UNIQUE	17	Count of records processed, count of unique values
VERIFY	9	Count of records processed

Parameter List Interface Example

The example in Figure 41 on page 428 shows a portion of an assembler language program that uses the Parameter List Interface. Table 56 on page 430 shows the JCL you might use to run the program in Figure 41 on page 428.

```
DEPTVIEW CSECT
         ...
* SET UP PARAMETER LIST AND CALL ICETOOL
         LA    R1,PARLST        LOAD ADDRESS OF PARAMETER LIST
         LINK  EP=ICETOOL       CALL ICETOOL
         LTR   R15,R15          IF ANY OPERATIONS WERE NOT SUCCESSFUL,
         BNZ   CKSTAT1            DETERMINE WHICH ONE FAILED
* ALL OPERATIONS WERE SUCCESSFUL
*  CHECK EMPLOYEES PER DEPARTMENT AGAINST ACCEPTABLE LEVEL
         CP    RT2AVG1,EMAVGCK    IF AVERAGE IS ACCEPTABLE,
         BNH   CKQUAL               NO MESSAGE IS NEEDED
* ISSUE A MESSAGE SHOWING AVERAGE, MINIMUM, MAXIMUM, AND
* TOTAL NUMBER OF EMPLOYEES PER DEPARTMENT.
         ...
*  CHECK EXPENSES PER DEPARTMENT AGAINST ACCEPTABLE LEVEL
CKQUAL   CP    RT2AVG2,TLAVGCK    IF AVERAGE IS ACCEPTABLE,
         BNH   PCTCALC              NO MESSAGE IS NEEDED
* ISSUE A MESSAGE SHOWING AVERAGE, MINIMUM, MAXIMUM, AND
* TOTAL EXPENSES PER DEPARTMENT.
         ...
*  CALCULATE THE PERCENTAGE OF DEPARTMENTS OVER/UNDER EMPLOYEE LIMIT
PCTCALC  MVC   WORK+2(4),RT3RCDS+4   COPY NUMBER OF DEPARTMENTS
         SP    WORK+2(4),RT3RNG+4(4) SUBTRACT 'NUMBER WITHIN LIMITS' TO
*                                     GET 'NUMBER OVER/UNDER LIMIT'
         CP    WORK+2(4),P0          IF NONE OVER/UNDER LIMIT,
         BE    PCTPRT                 PERCENTAGE IS ZERO
         MP    WORK+2(4),P100        MULTIPLY NUMBER OVER/UNDER BY 100
         DP    WORK(6),RT3RCDS+4(4)  DIVIDE BY NUMBER OF DEPARTMENTS
* ISSUE A MESSAGE SHOWING THE PERCENTAGE OF DEPARTMENTS THAT ARE
* OVER/UNDER EMPLOYEE LIMIT
PCTPRT   UNPK  PCTVAL,WORK(2)     CONVERT AVERAGE TO PRINTABLE EBCDIC
         OI    PCTVAL+2,X'F0'     ENSURE LAST DIGIT IS PRINTABLE
         ...
* ONE OR MORE OPERATIONS FAILED
CKSTAT1  CLI   RT1STAT,0          IF OPERATION 1 WORKED,
         BNE   CKSTAT2              CHECK OPERATION 2
* ISSUE MESSAGE: OPERATION 1 FAILED - CHECK TOOLMSG
         ...
* PARAMETER LIST
PARLST   DC    A(0)               USE PARAMETER LIST INTERFACE
         DC    A(ST1A)            STATEMENT AREA 1 ADDRESS
         DC    A(RT1A)            RETURN AREA 1 ADDRESS
         DC    A(ST2A)            STATEMENT AREA 2 ADDRESS
         DC    A(RT2A)            RETURN AREA 2 ADDRESS
         DC    A(ST3A)            STATEMENT AREA 3 ADDRESS
         DC    A(RT3A)            RETURN AREA 3 ADDRESS
         DC    F'.*-1'              END OF PARAMETER LIST* OPERATOR STATEMENT AREAS
```

Figure 41. ICETOOL Parameter List Interface Example (Part 1 of 2)

```
*   COPY OPERATION
ST1A     DC    AL2(ST1E-ST1)      LENGTH OF STATEMENT AREA 1
ST1      DC    CL80'* CREATE TWO COPIES OF THE DENVER SITE'
         DC    CL80'* DEPARTMENT RECORDS'
         DC    CL80'COPY FROM(IN) TO(OUT1,OUT2) USING(CTL1)'
ST1E     EQU   *
*   STATS OPERATION
ST2A     DC    AL2(ST2E-ST2)      LENGTH OF STATEMENT AREA 2
ST2      DC    CL80'* GET STATISTICS FOR NUMBER OF EMPLOYEES'
         DC    CL80'* AND TRAVEL EXPENSES PER DEPARTMENT'
         DC    CL80'STATS FROM(OUT1) ON(15,2,PD) ON(28,8,ZD)'
ST2E     EQU   *
*   RANGE OPERATION
ST3A     DC    AL2(ST3E-ST3)      LENGTH OF STATEMENT AREA 3
ST3      DC    CL80'* DETERMINE THE NUMBER OF DEPARTMENTS THAT ARE'
         DC    CL80'* WITHIN THE LIMIT FOR NUMBER OF EMPLOYEES'
         DC    CL80'RANGE FROM(OUT1) ON(15,2,PD) -'
         DC    CL80' HIGHER(10) LOWER(21)'
ST3E     EQU   *
* RETURN AREAS
   COPY OPERATION
RT1A     DC    AL2(RT1E-RT1STAT)  LENGTH OF RETURN AREA 1
RT1STAT  DS    C                  OPERATION STATUS
RT1E     EQU   *
*   STATS OPERATION
RT2A     DC    AL2(RT2E-RT2STAT)  LENGTH OF RETURN AREA 2
RT2STAT  DS    C                  OPERATION STATUS
RT2RCDS  DS    PL8                COUNT OF RECORDS PROCESSED
RT2MIN1  DS    PL8                FIELD 1 - MINIMUM VALUE
RT2MAX1  DS    PL8                FIELD 1 - MAXIMUM VALUE
RT2AVG1  DS    PL8                FIELD 1 - AVERAGE VALUE
RT2TOT1  DS    PL8                FIELD 1 - TOTAL VALUE
RT2MIN2  DS    PL8                FIELD 2 - MINIMUM VALUE
RT2MAX2  DS    PL8                FIELD 2 - MAXIMUM VALUE
RT2AVG2  DS    PL8                FIELD 2 - AVERAGE VALUE
RT2TOT2  DS    PL8                FIELD 2 - TOTAL VALUE
RT2E     EQU   *
*   RANGE OPERATION
RT3A     DC    AL2(RT3E-RT3STAT)  LENGTH OF RETURN AREA 3
RT3STAT  DS    C                  OPERATION STATUS
RT3RCDS  DS    PL8                COUNT OF RECORDS PROCESSED
RT3RNG   DS    PL8                COUNT OF VALUES IN RANGE
RT3E     EQU   *
* VARIABLES/CONSTANTS
WORK     DS    PL6                WORKING VARIABLE
P100     DC    P'100'             CONSTANT 100
P0       DC    P'0'               CONSTANT 0
EMAVGCK  DC    P'17'              ACCEPTABLE AVERAGE EMPLOYEE COUNT
TLAVGCK  DC    P'5000'            ACCEPTABLE AVERAGE TRAVEL EXPENSES
PCTVAL   DS    PL3                PERCENTAGE OF DEPARTMENTS THAT ARE
*                                   OVER/UNDER EMPLOYEE LIMIT
         ...
```

Figure 41. ICETOOL Parameter List Interface Example (Part 2 of 2)

Calling ICETOOL from a Program

Table 56. JCL for Parameter List Interface Program Example

```
//EXAMP JOB A402,PROGRAMMER
//INVOKE EXEC PGM=DEPTVIEW,REGION=1024K
//STEPLIB DD DSN=...  Link library containing DEPTVIEW
//TOOLMSG DD SYSOUT=A
//DFSMSG DD SYSOUT=A
//IN DD DSN=ALL.DEPTS,DISP=SHR
//OUT1 DD DSN=ALL.DEPTS.BACKUP1,DISP=OLD
//OUT2 DD DSN=ALL.DEPTS.BACKUP2,DISP=OLD
//CTL1CNTL DD *
* SELECT ONLY THE DENVER SITE DEPARTMENT RECORDS
  INCLUDE COND=(1,12,CH,EQ,C'DENVER')
/*
```

ICETOOL Notes and Restrictions

- Small REGION values can cause storage problems when ICETOOL calls DFSORT. Large REGION values give DFSORT the flexibility to use the storage it needs for best performance. We recommend that you use a REGION value of at least 1024K for ICETOOL.
- Each ICETOOL operation results in a set of ICETOOL messages in the TOOLMSG data set, and a corresponding set of DFSORT messages in the DFSMSG data set. For a particular call to DFSORT, you can relate the sets of messages in the TOOLMSG and DFSMSG data sets by using the unique identifier for that call. Just match the identifier printed in ICETOOL message ICE606I or ICE627I to the same identifier printed in DFSORT message ICE200I. This is particularly important if an ICETOOL operation fails due to an error detected by DFSORT (return code 16).
- Since ICETOOL calls DFSORT, the installation options used for DFSORT are those in effect for the appropriate program-invoked environment, that is, ICEAM2 or ICEAM4 or an ICETDx module activated for the ICEAM2 or ICEAM4 environment. The DFSORT installation options apply only to DFSORT, not to ICETOOL. For example, ICEMAC option MSGCON=ALL causes DFSORT, but not ICETOOL, to write messages to the master console. The one exception is ICEMAC option SDBMSG; the value in effect from ICEAM2 or ICEAM4 is used for ICETOOL's TOOLMSG data set.
- When ICETOOL calls DFSORT, it passes control statements and options appropriate to the specific operations being performed. You should not override the DFSORT control statements or options passed by ICETOOL unless you understand the ramifications of doing so.

  For example, ICETOOL passes the NOABEND option to DFSORT to ensure that ICETOOL will regain control if DFSORT issues an error message. If you specify: you cause DFSORT to abend when it issues an error message, thus preventing

  ```
  //DFSPARM DD *
    DEBUG ABEND
  ```

  ICETOOL from performing subsequent operators.
- Tape work data sets *cannot* be used with ICETOOL.
- An ON field must not include bytes beyond the fixed part of variable length input records. The entire field specified must be present in every input record, otherwise, DFSORT issues message ICE015A, ICE218A, or ICE027A and terminates.
- If a SmartBatch pipe data set is used for FROM, TO, LIST, or DISCARD and an error is detected by ICETOOL or DFSORT, an ABEND is generated in order to allow appropriate error propagation by the system to other applications that may

be accessing the same SmartBatch pipe data set. See “SmartBatch Pipe Considerations” on page 15 for information about special user abend processing in conjunction with SmartBatch pipe data sets.

If DFSORT detects the error, it issues the appropriate ABEND as directed by the ABCODE installation option (see *Installation and Customization*).

If ICETOOL detects the error, it issues ABEND 2222.

ICETOOL Return Codes

ICETOOL sets a return code for each operation it performs in STOP or CONTINUE mode and passes back the highest return code it encounters to the operating system or the invoking program.

For successful completion of all operations, ICETOOL passes back a return code of 0 or 4 to the operating system or the invoking program.

For unsuccessful completion due to an unsupported operating system, ICETOOL passes back a return code of 24 to the operating system or invoking program.

For unsuccessful completion of one or more operations, ICETOOL passes back a return code of 12, 16, or 20 to the operating system or the invoking program.

The meanings of the return codes that ICETOOL passes back (in register 15) are:

0 **Successful completion**. All operations completed successfully.

4 **Successful completion**. All operations completed successfully. DFSORT passed back a return code of 4 for one or more operations. See “DFSORT Messages and Return Codes” on page 22 for details.

12 **Unsuccessful completion**. ICETOOL detected one or more errors that prevented it from completing successfully. Messages for these errors were printed in the TOOLMSG data set.

16 **Unsuccessful completion**. DFSORT detected one or more errors that prevented ICETOOL from completing successfully. Messages for these errors were printed in the DFSMSG data set.

20 **Message data set error**. The TOOLMSG DD statement was not present or the TOOLMSG data set was not opened.

24 **Unsupported operating system**. This operating system is not supported by this release of DFSORT. Only current or subsequent releases of the following systems are supported:

- OS/390
- MVS/ESA

Chapter 7. Using Symbols for Fields and Constants

Field and Constant Symbols Overview . . . 433
DFSORT Example . . . 434
SYMNAMES DD Statement . . . 436
SYMNOUT DD Statement . . . 436
SYMNAMES Statements . . . 436
Comment and Blank Statements . . . 436
Symbol Statements . . . 437
Keyword Statements . . . 443
Using SYMNOUT to Check Your SYMNAMES Statements . . . 445
Using Symbols in DFSORT Statements . . . 446
SORT and MERGE . . . 447
SUM . . . 447
INCLUDE and OMIT . . . 447
INREC and OUTREC . . . 448
OUTFIL . . . 449
Using Symbols in ICETOOL Operators . . . 450
DISPLAY . . . 451
OCCUR . . . 451
RANGE . . . 451
SELECT . . . 451
STATS, UNIQUE and VERIFY . . . 451
ICETOOL Example . . . 451
Notes for Symbols . . . 453

Field and Constant Symbols Overview

This chapter describes DFSORT's simple and flexible method for using symbols in DFSORT and ICETOOL statements. You can define and use a symbol for any field or constant that is recognized in a DFSORT control statement or ICETOOL operator. This makes it easy to create and reuse collections of symbols (that is, mappings) representing information associated with various record layouts.

In addition, you can obtain and use collections of DFSORT symbols created specifically for records produced by other products (for example, RACF, DFSMSrmm and DCOLLECT) or by your site. Visit the DFSORT home page at the following URL to obtain information about downloading DFSORT symbol mappings for records produced by other products, and examples that use these symbols:

```
http://www.ibm.com/storage/dfsort/
```

Symbols can increase your productivity by automatically providing the positions, lengths and formats of the fields, and the values of the constants, associated with the particular records you are processing with DFSORT or ICETOOL.

To use symbols for DFSORT or ICETOOL, you just:

1. Create or obtain the DFSORT symbol data sets you need. Symbol data sets contain symbols that map the fields in your records, and constants used for comparisons, titles, headings and so on. The symbols are specified in DFSORT's simple but flexible SYMNAMES statement format. Symbols can be easily added or modified using an editor, such as ISPF EDIT.
2. Include a SYMNAMES DD statement in your job. SYMNAMES specifies one or more symbol data sets (sequential, partitioned member, DD *) to be used for

© Copyright IBM Corp. 1973, 2000

your DFSORT or ICETOOL application. SYMNAMES can be used to concatenate as many symbol data sets as you like.

3. Use the symbols from SYMNAMES where appropriate in DFSORT control statements or ICETOOL operators. You can mix symbols (for example, Last_Name) with regular fields (for example, p,m,f) and constants (for example, C'string').

DFSORT or ICETOOL will read SYMNAMES and use the symbols it contains to ″transform″ your statements by performing symbol substitution. DFSORT or ICETOOL will then use the transformed statements as if you had specified them directly.

If your record layout changes, just make a corresponding change to your DFSORT symbol data set. DFSORT will use the new mapping to ″transform″ your symbols correctly, even if positions change, so you won't have to change your statements. Be sure that your symbol definitions match your record layout before you attempt to use them.

DFSORT Example

The example below shows the JCL and control statements for a simple DFSORT job that uses symbols.

Let's say you created a symbols data set named MY.SYMBOLS that contains the following SYMNAMES statements:

```
* Fields
First_Name,6,20,CH
Last_Name,*,=,=
Account_Number,53,3,PD
SKIP,2
Balance,*,6,ZD
Type,*,8,CH

* Constants
Loan,'LOAN'
Check,'CHECKING'
Level1,50000
Level2,-100
```

Here's the JCL and control statements for the example:

```
//EXAMP JOB A402,PROGRAMMER
//RUNIT EXEC PGM=ICEMAN
//SYMNAMES  DD  DSN=MY.SYMBOLS,DISP=SHR
//SYMNOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SORTIN DD ...
//SORTOUT DD ...
//SYSIN    DD    *
  INCLUDE COND=((Type,EQ,Loan,AND,Balance,GT,Level1),OR,
                (Type,EQ,Check,AND,Balance,LE,Level2))
  SORT FIELDS=(Last_Name,A,First_Name,A,
               Type,A,Account_Number,D)
/*
```

This example is only meant to give you a quick overview of how symbols can be used. The rest of this chapter will explain all of the details, but here are a few important things to take note of:

- The SYMNAMES DD indicates you want DFSORT or ICETOOL to do symbol processing. The SYMNAMES data set contains the symbols for fields and constants.

- DFSORT or ICETOOL will print your original symbols and the symbol table constructed from them in the SYMNOUT data set, if you specify it. You might want to use SYMNOUT while debugging a set of symbols and then remove it, or you might want to keep SYMNOUT permanently so you can always see your original symbols and the symbol table.
- The simple, yet flexible, format for SYMNAMES statements is:

  ```
  symbol,value remark
  ```

  where value can represent a field (p,m,f or p,m or p) or a constant (C'string', c'string', 'string', X'string', x'string', B'string', b'string', n, +n or -n). Leading blanks are allowed before symbol so indentation can be used. For example, the following SYMNAMES statements could be specified:

  ```
  Div1_Department,8,1,BI      Division 1 Department
    Research,B'0001....'        Research Departments
    Marketing,B'0010....'       Marketing Departments
    Development,B'0100....'     Development Departments
  ```
- Symbols are case-sensitive: Frank, FRANK and frank are three **different** symbols.
- An asterisk (*) can be used to assign the *next position* to p. For example:

  ```
  Symbola,6,20,CH
  Symbolb,*,5,BI
  Symbolc,*,12,ZD
  ```

  is the same as specifying:

  ```
  Symbola,6,20,CH
  Symbolb,26,5,BI
  Symbolc,31,12,ZD
  ```

  By using * for p, you can map consecutive fields in your records without having to compute their actual positions.
- SKIP,n can be used to advance the *next position* by n bytes so it can be used for *. For example:

  ```
  Symbola,6,20,CH
  SKIP,2
  Symbolb,*,5,BI
  ```

  is the same as specifying:

  ```
  Symbola,6,20,CH
  Symbolb,28,5,BI
  ```

  SKIP,n gives you an easy way to skip unused bytes. Other mapping aids allow you to reset the *next position* (POSITION,q or POSITION,symbol), or align the *next position* on a halfword (ALIGN,H), fullword (ALIGN,F) or doubleword (ALIGN,D).
- An equal sign (=) can be used for p, m or f to assign the previous position, length or format to p, m, or f, respectively. For example:

  ```
  Symbola,6,20,CH
    Symbola1,=,8,=
    Symbola2,*,12,=
  Symbold,*,=,ZD
  ```

  is the same as specifying:

  ```
  Symbola,6,20,CH
    Symbola1,6,8,CH
    Symbola2,14,12,CH
  Symbold,26,12,ZD
  ```

By using = and *, you can easily map fields onto other fields.

- Symbols for fields and constants can be specified in any order. However, the use of * and = imposes order dependencies on symbols for fields.
- Comment statements and blank statements are allowed in SYMNAMES.

SYMNAMES DD Statement

A SYMNAMES DD statement indicates you want DFSORT or ICETOOL to do symbol processing. It specifies the SYMNAMES data set (SYMNAMES for short), which can consist of one DFSORT symbol data set or many concatenated symbol data sets.

A symbol data set can be a sequential data set, a partitioned member or a DD * data set; all three types can be concatenated together for the SYMNAMES DD. Each symbol data set must contain SYMNAMES statements describing the symbols for fields and constants to be used for the DFSORT or ICETOOL application. Each symbol data set must have the following attributes: RECFM=F or RECFM=FB and LRECL=80.

For best performance, use a large block size, such as the system determined optimum block size, for all DFSORT symbol data sets.

If a SYMNAMES DD statement is not present, or SYMNAMES is empty, symbol processing is not performed.

SYMNOUT DD Statement

A SYMNOUT DD statement specifies a data set in which you want DFSORT or ICETOOL to print your original SYMNAMES statements and the symbol table constructed from them. DFSORT or ICETOOL uses RECFM=FBA, LRECL=121 and the specified BLKSIZE for the SYMNOUT data set (SYMNOUT for short).

If the BLKSIZE you specify is not a multiple of 121, or you do not specify the BLKSIZE:

- the system determined optimum block size is used, if supported
- otherwise, BLKSIZE=121 is used.

For best performance, use a large block size, such as the system determined optimum block size, for the SYMNOUT data set.

SYMNAMES Statements

Each symbol in SYMNAMES must be described using a SYMNAMES statement. A SYMNAMES statement can be a symbol statement, keyword statement, comment statement or blank statement.

Comment and Blank Statements

A statement with an asterisk (*) in column 1 is treated as a comment statement. It is printed in SYMNOUT (if specified), but otherwise not processed. A statement with blanks in columns 1 through 80 is treated as a blank statement. It is printed in SYMNOUT (if specified), but otherwise not processed.

Symbol Statements

The general format for a symbol statement is:

```
symbol,value remark
```

The general coding rules are as follows:

- Columns 1 through 80 are scanned.
- The symbol can start in column 1 or in any column after 1.
- A remark is optional, but if specified, must be separated from the value by at least one blank. A remark is printed in SYMNOUT (if specified), but otherwise not processed.
- A semicolon (;) can be used instead of a comma (,) between the symbol and the value.
- Continuation is not allowed. Each symbol and value must be completely specified on one line.

The specific syntax for symbol statements is:

```
►►─symbol,─┬─constant─┬──────────────────────────────────────────►◄
           └─field────┘
```

Symbol: A symbol can be 1 to 50 EBCDIC characters consisting of uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), the number sign (#), the dollar sign ($), the commercial at sign (@) and the underscore(_). The first character of a symbol must not be a number. Symbols are treated as case-sensitive: Frank, FRANK and frank are three **different** symbols.

The following DFSORT/ICETOOL reserved words (uppercase only as shown) are not allowed as symbols: A, AC, ALL, AND, AQ, ASL, AST, BI, CH, CLO, COPY, COUNT, COUNT15, CSF, CSL, CST, CTO, D, DATE, D1, D2, E, F, FI, FL, FS, H, HEX, LS, Mn, Mnn, NONE, NUM, OL, OR, OT, PAGE, PAGEHEAD, PD, PD0, SEQNUM, SS, SUBCOUNT, SUBCOUNT15, TIME, TS, VALCNT, VLEN, X, Y2x, Y2xx, Z and ZD where n is 0-9 and x is any character. Lower case and mixed case forms of these words, such as None and page, can be used as symbols.

POSITION, SKIP and ALIGN (uppercase only) are treated as keywords as discussed in “Keyword Statements” on page 443 and thus are not recognized as symbols. However, lowercase and mixed case forms of these words, such as Position and skip, can be used as symbols.

Some examples of valid symbols are: Account_Number, CON12, PHONE# and count.

Some examples of invalid symbols are: 123_Account (starts with a number), COUNT (reserved word) and Account-Number (dash is invalid).

Constant: A constant can be a character string, hexadecimal string, bit string or decimal number.

A symbol for a constant value must be used only where such a constant is allowed and has the desired result. Otherwise, substitution of the constant for the symbol

will result in an error message or unintended processing. For example, if the following SYMNAMES statement is specified:

```
SYMB,B'10110001'
```

SYMB can be used in this INCLUDE statement:

```
INCLUDE COND=(12,1,BI,EQ,SYMB)
```

since a bit string can be compared to a binary field. However, SYMB will result in an error message if used in this INCLUDE statement:

```
INCLUDE COND=(12,1,CH,EQ,SYMB)
```

since a bit string cannot be compared to a character field.

Make sure the constants that will be substituted for your symbols are appropriate. If in doubt, check the rules for constants given in the description of the relevant operand.

A symbol can represent one of the following types of constants:

- A character string in the format ’xx...x’, C’xx...x’ or c’xx...x’.

  The value x may be any EBCDIC character. You can specify up to 64 characters for the string. c’xx...x’ will be treated like C’xx...x’.

  If you want to include a single apostrophe in the character string, you must specify it as two single apostrophes (each pair of apostrophes counts as two characters towards the 64 character limit for the string). Thus:

  ```
  Required:  O'Neill         Specify:  C'O''Neill'
  ```

  Double-byte data may be used in a character string (each pair of shift-in/shift-out characters and each double-byte character counts as two characters towards the 64 character limit for the string). See “INCLUDE Control Statement” on page 93 for details on double-byte data.

  Some examples of valid character strings are: ’+0.193’, c’Title’, C’O’’Neil’, C’J62,J82,M72’ and ’’.

  Some examples of invalid character strings are: C’AB’’ (apostrophes not paired), c’title (ending apostrophe missing) and C’O’NEIL’ (one apostrophe after O instead of two).

  You can use C’xx...x’ and ’xx...x’ interchangeably. ’xx...x’ will be substituted for symbols where appropriate even if C’xx...x’ is specified in SYMNAMES. Likewise, C’xx...x’ will be substituted for symbols where appropriate even if ’xx...x’ is specified in SYMNAMES. For example, if these SYMNAMES statements are specified:

  ```
  My_Title,c'My Report'
  My_Heading,C'January'
  DEPT1,'J82'
  DEPT2,c'M72'
  ```

  the ICETOOL operator:

  ```
  DISPLAY TITLE(My_Title) HEADER(My_Heading) ...
  ```

  will be transformed to:

  ```
  DISPLAY TITLE('My Report') HEADER('January') ...
  ```

  and the INCLUDE statement:

```
INCLUDE COND=(5,3,EQ,DEPT1,OR,5,3,EQ,DEPT2),FORMAT=CH
```

will be transformed to:

```
INCLUDE COND=(5,3,EQ,C'J82',OR,5,3,EQ,C'M72'),FORMAT=CH
```

Although the rules for character strings used as symbols generally follow the rules for INCLUDE/OMIT character strings, keep in mind that the same rules do not apply for character strings in all DFSORT and ICETOOL operands, so use symbols representing character strings appropriately. For example, ICETOOL only allows up to 50 characters for a TITLE string, so TITLE(MYCON) would result in an error message if MYCON is a 64-character string, even though MYCON could be used without error in an INCLUDE statement. As another example, double-byte characters would be recognized in a character string substituted for a symbol in an INCLUDE statement, but would not be recognized in a character string substituted in an OUTREC statement.

- A hexadecimal string in the format X'yy...yy' or x'yy...yy'.

  The value yy represents any pair of hexadecimal digits. Each hexadecimal digit must be 0-9, A-F or a-f. You can specify up to 32 pairs of hexadecimal digits. x'yy...yy' will be treated like X'yy...yy'. a-f will be treated like A-F.

  Some examples of valid hexadecimal strings are: X'F2C3', x'2fa71e', and X'07'.

  Some examples of invalid hexadecimal strings are: X'F2G301' (G is not a valid hexadecimal digit), x'bf3' (unpaired hexadecimal digits) and X'' (no hexadecimal digits).
- A bit string in the format B'bbbbbbbb...bbbbbbbb' or b'bbbbbbbb...bbbbbbbb'.

  The value bbbbbbbb represents 8 bits that constitute a byte. Each bit must be 1, 0 or . (period). You can specify up to 8 groups of 8 bits. b'bbbbbbbb...bbbbbbbb' will be treated like B'bbbbbbbb...bbbbbbbb'.

  Some examples of valid bit strings are: B'01100100', b'11..00..000..111' and B'11.....'.

  Some examples of invalid bit strings are: B'0101' (not a multiple of eight bits), b'00..11....' (not a multiple of eight bits), b'00000002' (2 is not a valid bit) and B'' (no bits).
- A decimal number in the format n, +n or -n. You can specify from 1 to 15 significant digits.

  Some examples of valid decimal numbers are: +270, 270, 000036, +0 and -2000000.

  Some examples of invalid decimal numbers are: ++15 (too many plus signs), 280- (sign in wrong place) and 2.8 (period is not allowed).

Field: A field can be specified as p,m,f (position, length and format), p,m (position and length) or p (position only).

A symbol for a field value must be used only where such a field is allowed and has the desired result. Otherwise, substitution of the field for the symbol will result in an error message or unintended processing. For example, if the following SYMNAMES statement is specified:

```
Field1,15,2,CH
```

Field1 can be used in a SORT statement such as:

```
SORT FIELDS=(Field1,A)
```

since a character field is allowed in a SORT statement. However, Field1 will result in an error message if used in a SUM statement such as:

```
SUM FIELDS=(Field1)
```

since a character field is not allowed in a SUM statement.

Make sure the fields that will be substituted for your symbols are appropriate. If in doubt, check the rules for p, m and f given in the description of the relevant operand.

You can specify p,m,f for your field symbols and then use them where p,m is required because DFSORT or ICETOOL will substitute just p,m when appropriate. For example, if you specify the following in SYMNAMES:

```
First_Field,12,2,BI
Second_Field,18,6,CH
Third_Field,28,5,PD
Fourth_Field,36,3
Fifth_Field,52,4,PD
Max,200000
```

These DFSORT control statements:

```
OMIT COND=(Fifth_Field,GT,Max)
SORT FIELDS=(First_Field,A,Fourth_Field,A),FORMAT=CH
SUM FIELDS=(Second_Field,ZD)
OUTFIL OUTREC=(First_Field,2X,Third_Field,M11,Fourth_Field)
```

will be transformed to:

```
OMIT COND=(52,4,PD,GT,200000)
SORT FIELDS=(12,2,A,36,3,A),FORMAT=CH
SUM FIELDS=(18,6,ZD)
OUTFIL OUTREC=(12,2,2X,28,5,PD,M11,36,3)
```

Note that DFSORT did the following substitutions:

- OMIT statement: p,m,f for Fifth_Field as required by COND without FORMAT.
- SORT statement: p,m for First_Field and Fourth_Field as required by FIELDS with FORMAT.
- SUM statement: p,m for Second_Field as required for symbol,f (that is, Second_Field,ZD).
- OUTFIL statement: p,m for First_Field as required by the OUTREC operand for an unedited field (that is, First_Field), but p,m,f for Third_Field as required by the OUTREC operand for an edited field (that is, Third_Field,M11).

The general rules for using p, m and f in symbol statements are as follows:

- **p** can be a number, an asterisk (*) or an equal sign (=). A number from 1 to 32752 is allowed in p,m or p,m,f. Since p (position only) cannot be distinguished from the constant n, 1 to 15 significant digits are allowed for p (position only).

  An asterisk (*) can be used to assign the *next position* to p. Each time a symbol for p,m,f or p,m is read, the *next position* is set to p+m. Additionally, the *next position* can be modified by keyword statements (see “Keyword Statements” on page 443). When * is specified for p, the *next position* is assigned to p. If the *next position* has not been set when * is used for p (for example, * is used in the first symbol), p is set to 1.

  The symbol table printed in the SYMNOUT data set (if specified) will show you the actual positions assigned when you specify * for p.

  As an example of how * can be used, if you specify the following SYMNAMES statements:

```
Sym1,*,5,ZD
Con1,27
Sym2,*,2,BI
Field1,8,13,CH
Field2,*,5,PD
Field3,*,2,FI
```

SYMNOUT will show the following symbol table:

```
Sym1,1,5,ZD
Con1,27
Sym2,6,2,BI
Field1,8,13,CH
Field2,21,5,PD
Field3,26,2,FI
```

By using * for p, you can map consecutive fields in your records without having to compute their actual positions. You can also map fields added between other fields without having to change the p values for the original or inserted fields. * is also useful for creating mappings of contiguous fields using concatenated symbol data sets. As a simple example, if you specify:

```
//SYMNAMES DD DSN=MY.SYMPDS(RDW),DISP=SHR
//         DD DSN=MY.SYMPDS(SECTION1),DISP=SHR
//         DD DSN=MY.SYMPDS(SECTION2),DISP=SHR
```

and the RDW member contains:

```
RDW,1,4,BI
```

the SECTION1 member contains:

```
Flag_Byte,*,1,BI
  Error1,X'80'
  Error2,X'40'
Count_of_Parts,*,5,ZD
```

and the SECTION2 member contains:

```
New_Parts,*,5,ZD
Old_Parts,*,5,ZD
Variable_Fields,*
```

SYMNOUT will show the following symbol table:

```
RDW,1,4,BI
Flag_Byte,5,1,BI
Error1,X'80'
Error2,X'40'
Count_of_Parts,6,5,ZD
New_Parts,11,5,ZD
Old_Parts,16,5,ZD
Variable_Fields,21
```

You might use these symbols in the following statements:

```
  OPTION COPY
  OUTFIL FNAMES=ERR1,INCLUDE=(Flag_Byte,EQ,Error1),
     OUTREC=(RDW,Count_of_Parts,Variable_Fields)
  OUTFIL FNAMES=ERR2,INCLUDE=(Flag_Byte,EQ,Error2),
     OUTREC=(RDW,New_Parts,Old_Parts,Variable_Fields)
```

An equal sign (=) can be used to assign the *previous position* to p. Each time a symbol for p,m,f or p,m is read, the *previous position* is set to p. Additionally, the *previous position* can be modified by a POSITION keyword statement (see

below). When = is specified for p, the *previous position* is assigned to p. If the *previous position* has not been set when = is used for p, an error message is issued.

The symbol table printed in the SYMNOUT data set (if specified) will show you the actual positions assigned when you specify = for p.

As an example of how = can be used for p, if you specify the following SYMNAMES statements:

```
Sym1,5,4,CH
  Sym2,=,2,CH
  Sym3,*,2,CH
```

SYMNOUT will show the following symbol table:

```
Sym1,5,4,CH
Sym2,5,2,CH
Sym3,7,2,CH
```

By using = and * for p, you can easily map fields onto other fields.

Whenever you use = for p, you must ensure that the *previous position* is the one you want. In particular, if you insert a new field symbol with the wrong position before a symbol that uses = for p, you will need to change = to the actual position you want.

- **m** can be an equal sign (=) or a number from 1 to 32752. An equal sign (=) can be used to assign the *previous length* to m. Each time a symbol for p,m,f or p,m is read, the *previous length* is set to m. When = is specified for m, the *previous length* is assigned to m. If the *previous length* has not been set when = is used for m, an error message is issued.

  The symbol table printed in the SYMNOUT data set (if specified) will show you the actual lengths assigned when you specify = for m.

  As an example of how = can be used for m, if you specify the following SYMNAMES statements:

  ```
  Flags1,5,1,BI
    Error1,X'08'
  Flags2,15,=,BI
    Error2,X'04'
  Flags3,22,=,BI
    Error3,X'23'
  ```

  SYMNOUT will show the following symbol table:

  ```
  Flags1,5,1,BI
  Error1,X'08'
  Flags2,15,1,BI
  Error2,X'04'
  Flags3,22,1,BI
  Error3,X'23'
  ```

  Whenever you use = for m, you must ensure that the *previous length* is the one you want. In particular, if you insert a new field symbol with the wrong length before a symbol that uses = for m, you will need to change = to the actual length you want.
- **f** can be an equal sign (=) or one of the following formats: AC, AQ, ASL, AST, BI, CH, CLO, CSF, CSL, CST, CTO, D1, D2, FI, FL, FS, LS, OL, OT, PD, PD0, SS, TS, Y2B, Y2C, Y2D, Y2DP, Y2P, Y2PP, Y2S, Y2T, Y2TP, Y2U, Y2UP, Y2V, Y2VP, Y2W, Y2WP, Y2X, Y2XP, Y2Y, Y2YP, Y2Z or ZD.

You can specify f using uppercase letters (for example, CH), lowercase letters (for example, ch) or mixed case letters (for example, Ch). f specified in any case will be treated like uppercase.

An equal sign (=) can be used to assign the *previous format* to f. Each time a symbol for p,m,f is read, the *previous format* is set to f. When = is specified for f, the *previous format* is assigned to f. If the *previous format* has not been set when = is used for f, an error message is issued.

The symbol table printed in the SYMNOUT data set (if specified) will show you the actual formats assigned when you specify = for f.

As an example of how = can be used for f, if you specify the following SYMNAMES statements:

```
Field1,5,8,CH
Field1a,=,3
Field2,*,12,=
Field3,*,20,=
```

SYMNOUT will show the following symbol table:

```
Field1,5,8,CH
Field1a,5,3
Field2,8,12,CH
Field3,20,20,CH
```

Whenever you use = for f, you must ensure that the *previous format* is the one you want. In particular, if you insert a new field symbol with the wrong format before a symbol that uses = for f, you will need to change = to the actual format you want.

Keyword Statements

The general format for a keyword statement is:

```
keyword,value remark
```

The general coding rules are as follows:

- Columns 1 through 80 are scanned.
- The keyword can start in column 1 or in any column after 1.
- The keyword must be specified in all **uppercase** letters. Otherwise, it will be treated as a symbol.
- A remark is optional, but if specified, must be separated from the value by at least one blank. A remark is printed in SYMNOUT (if specified), but otherwise not processed.
- A semicolon (;) can be used instead of a comma (,) between the keyword and the value.
- Continuation is not allowed. Each keyword and value must be completely specified on one line.

The specific syntax for keyword statements is:

```
►►─┬─POSITION,q──────────┬──────────────────────────────────────────►◄
   ├─POSITION,symbol─────┤
   ├─SKIP,n──────────────┤
   ├─ALIGN,H─────────────┤
   ├─ALIGN,F─────────────┤
   └─ALIGN,D─────────────┘
```

Using Symbols for Fields and Constants

Keyword statements can help you map the fields in your records by letting you set a starting position, skip unused bytes and align fields on specific boundaries.

- **POSITION,q** can be used to set the *next position* and the *previous position* to q. As discussed under p above, the *next position* is used when an asterisk (*) is specified for p in a symbol statement, and the *previous position* is used when an equal sign (=) is specified for p in a symbol statement. q can be a number from 1 to 32752. When you use POSITION,q you can use either * or = interchangably for p of the next symbol.

  As an example of how POSITION,q can be used, if you specify the following SYMNAMES statements:

  ```
  POSITION,27
  Account_Balance,*,5,PD
  Account_Id,*,8,CH
  POSITION,84
  New_Balance,=,20
  ```

  SYMNOUT will show the following symbol table:

  ```
  Account_Balance,27,5,PD
  Account_Id,32,8,CH
  New_Balance,84,20
  ```

- **POSITION,symbol** can be used to set the *next position* and the *previous position* to the position established for the indicated symbol. As discussed under p above, the *next position* is used when an asterisk (*) is specified for p in a symbol statement, and the *previous position* is used when an equal sign (=) is specified for p in a symbol statement. When you use POSITION,symbol you can use either * or = interchangeably for p of the next symbol.

  symbol can be any previously defined field symbol. Thus, POSITION,symbol can be used like the Assembler ORG instruction to map different fields onto the same area.

  As an example of how POSITION,symbol can be used, if you specify the following SYMNAMES statements:

  ```
  Workarea,21,100     Use workarea for volsers
    volser1,=,6,CH
    volser2,*,6,CH
  POSITION,Workarea   Reuse workarea for status and dsname
    status,=,1,BI
    dsname,*,44,CH
  ```

  SYMNOUT will show the following symbol table:

  ```
  Workarea,21,100
  volser1,21,6,CH
  volser2,27,6,CH
  status,21,1,BI
  dsname,22,44,CH
  ```

- **SKIP,n** can be used to add n bytes to the *next position*. As discussed under p above, the *next position* is used when an asterisk (*) is specified for p in a symbol statement. n can be a number from 1 to 32752.

  As an example of how SKIP,n can be used, if you specify the following SYMNAMES statements:

  ```
  Field#1,15,6,FS
   SKIP,4   Unused bytes
  Field#2,*,5,=
   SKIP,2   Unused bytes
  Field#3,*,8,CH
  ```

  SYMNOUT will show the following symbol table:

```
Field#1,15,6,FS
Field#2,25,5,FS
Field#3,32,8,CH
```

- **ALIGN,H** can be used to align the *next position* on a halfword boundary, that is, 1, 3, 5 and so on. As discussed under p above, the *next position* is used when an asterisk (*) is specified for p in a symbol statement. ALIGN,h will be treated like ALIGN,H.

  As an example of how ALIGN,H can be used, if you specify the following SYMNAMES statements:

  ```
  A1,7,3,CH
  ALIGN,H
  A2,*,2,BI
  ```

  SYMNOUT will show the following symbol table:

  ```
  A1,7,3,CH
  A2,11,2,BI
  ```

- **ALIGN,F** can be used to align the *next position* on a fullword boundary, that is, 1, 5, 9 and so on. As discussed under p above, the *next position* is used when an asterisk (*) is specified for p in a symbol statement. ALIGN,f will be treated like ALIGN,F.

  As an example of how ALIGN,F can be used, if you specify the following SYMNAMES statements:

  ```
  B1,7,3,CH
  ALIGN,f
  B2,*,4,BI
  ```

  SYMNOUT will show the following symbol table:

  ```
  B1,7,3,CH
  B2,13,4,BI
  ```

- **ALIGN,D** can be used to align the *next position* on a doubleword boundary, that is, 1, 9, 17 and so on. As discussed under p above, the *next position* is used when an asterisk (*) is specified for p in a symbol statement. ALIGN,d will be treated like ALIGN,D.

  As an example of how ALIGN,D can be used, if you specify the following SYMNAMES statements:

  ```
  C1,7,3,CH
  ALIGN,D
  C2,*,8,BI
  ```

  SYMNOUT will show the following symbol table:

  ```
  C1,7,3,CH
  C2,17,8,BI
  ```

Using SYMNOUT to Check Your SYMNAMES Statements

To avoid surprises, it's a good idea to check for errors and incorrect positions, lengths and formats in any SYMNAMES statements you create before you use them in DFSORT or ICETOOL statements.

The following simple job will cause DFSORT to issue messages in SYSOUT for any errors it detects in your SYMNAMES statements, allowing you to correct these errors before proceeding. Once your SYMNAMES statements are free of errors, the job will cause DFSORT to show the Symbol Table in SYMNOUT, allowing you to correct any incorrect positions, lengths or formats for your symbols (for example, those caused by incorrect use of *, =, SKIP, and so on).

```
//CHECK JOB A402,PROGRAMMER
//DOIT EXEC PGM=ICEMAN
//SYMNAMES DD ...        SYMNAMES statements to be checked
//SYMNOUT DD SYSOUT=*
//SORTIN DD *
//SORTOUT DD DUMMY
//SYSIN DD *
  OPTION COPY
/*
```

Once you've ″debugged″ your SYMNAMES statements, you can use them in DFSORT and ICETOOL statements.

Using Symbols in DFSORT Statements

You can use symbols in the following DFSORT control statements: INCLUDE, INREC, MERGE, OMIT, OUTFIL, OUTREC, SORT and SUM. In general, you can use symbols in these DFSORT statements where you can use constants ('string', C'string', X'string', B'string', n, +n or -n) and fields (p,m,f or p,m or p). See the discussion of each control statement in "Chapter 3. Using DFSORT Program Control Statements" on page 77 for a description of its syntax.

You can use symbols in these control statements in any source (that is, DFSPARM, SYSIN, SORTCNTL, and parameter lists).

When DFSORT transforms these control statements, it removes labels and remarks, and continues statements by placing an asterisk in column 72 and beginning the next line in column 16. DFSORT will list the original control statements as specified (with labels, remarks, comment statements and blank statements) by source, as well as the transformed statements.

Details and examples of the use of symbols for each applicable DFSORT control statement is given below. The examples are meant to illustrate variations in how symbols can be used and how they will be transformed. Therefore, the examples do not necessarily correspond to how symbols would be used in real applications.

The examples use these SYMNAMES statements:

```
C_Field1,6,5,CH
Any_Format,12,3
Z_Field1,22,8,ZD
P_Field1,30,4,PD
C_Field2,4,2,ch
SubString,16,3,SS
LIMIT,+12500
Depts,'J82,L92,M72'
Code_1,c'86A4Z'
QCON,C'Carrie''s Constant'
Stopper,X'FFFFFF'
Flags,35,1,BI
  Error,B'11010000'
  Empty,B'......01'
  Full,X'FF'
Lookup,52,1,BI
  Entry1,X'05'
  Value1,'Read'
  Entry2,X'20'
  Value2,'Update'
RDW,1,4              Record Descriptor Word
Variable_Fields,451  Variable fields at end of variable-length record
* Constants for report
Div_Title,'Division:  '
```

```
  BO_Title,'Branch Office'
BO_Hyphens,'-------------'
 BO_Equals,'=============='
  PL_Title,'       Profit/(Loss)'
PL_Hyphens,'--------------------'
 PL_Equals,'====================='
Total,'Total'
Lowest,'Lowest'
* Fields for report
Division,3,10,CH
Branch_Office,16,13,CH
Profit_or_Loss,31,10,ZD
```

SORT and MERGE

FIELDS operand: You can use symbols where you can use fields (p,m,f and p,m). A symbol for p,m,f results in substitution of p,m if FORMAT=f or symbol,f is specified.

Example 1

```
SORT FIELDS=(C_Field1,A,Z_Field1,D,
             C_Field2,ZD,A),EQUALS
```

will be transformed to:

```
SORT FIELDS=(6,5,CH,A,22,8,ZD,D,4,2,ZD,A),EQUALS
```

Example 2

```
MERGE FIELDS=(Any_Format,A,C_Field1,A),FORMAT=CH
```

will be transformed to:

```
MERGE FIELDS=(12,3,A,6,5,A),FORMAT=CH
```

SUM

FIELDS operand: You can use symbols where you can use fields (p,m,f and p,m). A symbol for p,m,f results in substitution of p,m if FORMAT=f or symbol,f is specified.

Example 1

```
SUM FIELDS=(Z_Field1,C_Field1,ZD)
```

will be transformed to:

```
SUM FIELDS=(22,8,ZD,6,5,ZD)
```

Example 2

```
SUM FORMAT=ZD,FIELDS=(C_Field1,Any_Format)
```

will be transformed to:

```
SUM FORMAT=ZD,FIELDS=(6,5,12,3)
```

INCLUDE and OMIT

COND operand: You can use symbols where you can use fields (p1,m1,f1 and p1,m1 and p2,m2,f2 and p2,m2) and constants (n, +n, -n, C’xx...x’, X’yy...yy’, Y’yyx...x’ and B’bbbbbbbb...bbbbbbbb’). A symbol for p,m,f results in substitution of p,m if FORMAT=f or symbol,f is specified. A symbol for ’string’ always results in substitution of C’string’.

Using Symbols for Fields and Constants

Example 1

```
INCLUDE COND=((Z_Field1,GT,LIMIT,AND,Any_Format,CH,EQ,C_Field2),OR,
              (SubString,NE,Depts),OR,
              (Flags,ALL,Error,AND,Flags,NE,Empty))
```

will be transformed to:

```
INCLUDE COND=((22,8,ZD,GT,+12500,AND,12,3,CH,EQ,4,2,CH),OR,(16,3,SS,NE*
              ,C'J82,L92,M72'),OR,(35,1,BI,ALL,B'11010000',AND,35,1,BI*
              ,NE,B'......01'))
```

Example 2

```
 OMIT FORMAT=BI,COND=(C_Field1,EQ,Code_1,OR,
                      Any_Format,EQ,Stopper,OR,
                      Flags,EQ,Full)
```

will be transformed to:

```
OMIT FORMAT=BI,COND=(6,5,EQ,C'86A4Z',OR,12,3,EQ,X'FFFFFF',OR,35,1,EQ,X*
              'FF')
```

INREC and OUTREC

FIELDS operand: You can use symbols where you can use fields (p,m,f and p,m and p) and non-repeated constants (C'xx...x' and X'yy...yy', but not nC'xx...x' or nX'yy...yy'). You cannot use symbols for edit patterns ('pattern').

In the CHANGE and NOMATCH suboperands, you can use symbols where you can use fields (q,n) and constants (C'xx...x', X'yy...yy' and B'bbbbbbbb').

A symbol for p,m,f or p,m,Y2x results in substitution of p,m,f or p,m,Y2x respectively if symbol,TO=fo is specified, but results in substitution of p,m if symbol,fo is specified because fo cannot be distinguished from f. For example, if Sym1 is defined as 5,4,ZD, SYM1,TO=PD is transformed to 5,4,ZD,TO=PD whereas SYM1,PD is transformed to 5,4,PD. Thus, you should always use symbol,TO=fo rather than symbol,fo.

A symbol for p,m,Y2x results in substitution of p,m,Y2x unless symbol,f or symbol,HEX is specified. A symbol for p,m,Y2xP results in substitution of p,m,Y2xP unless symbol,f or symbol,HEX is specified. A symbol for p,m,f where f is not Y2x or Y2xP results in substitution of p,m unless symbol,edit or symbol,to is specified.

A symbol for 'string' always results in substitution of C'string'.

Example 1

```
INREC FIELDS=(11:C_Field2,2X,C_Field1,F,Stopper,5C'*',
   Z_Field1,55:Depts,X,P_Field1,TO=FS,X,Z_Field1,M10)
```

will be transformed to:

```
    INREC FIELDS=(11:4,2,2X,6,5,F,X'FFFFFF',5C'*',22,8,55:C'J82,L92,M72',X*
                  ,30,4,PD,TO=FS,X,22,8,ZD,M10)
```

Example 2

```
OUTREC FIELDS=(RDW,  ** Record Descriptor Word **
                   Z_Field1,2Z,
                   3C'Symbol cannot be used for a repeated constant',
                   Code_1,Flags,
                   Variable_Fields)  ** Variable part of input record
```

will be transformed to:

```
OUTREC FIELDS=(1,4,22,8,2Z,3C'Symbol cannot be used for a repeated con*
              stant',C'86A4Z',35,1,451)
```

OUTFIL

INCLUDE and OMIT operands: You can use symbols where you can use fields (p1,m1,f1 and p1,m1 and p2,m2,f2 and p2,m2) and constants (n, +n, -n, C'xx...x', X'yy...yy', Y'yyx...x' and B'bbbbbbbb...bbbbbbbb'). A symbol for p,m,f results in substitution of p,m if symbol,f is specified. A symbol for 'string' always results in substitution of C'string'.

OUTREC operand: You can use symbols where you can use fields (p,m,f and p,m and p) and non-repeated constants (C'xx...x' and X'yy...yy', but not nC'xx...x' or nX'yy...yy'). You cannot use symbols for edit patterns ('pattern').

In the CHANGE and NOMATCH suboperands, you can use symbols where you can use fields (q,n) and constants (C'xx...x', X'yy...yy' and B'bbbbbbbb').

A symbol for p,m,f or p,m,Y2x results in substitution of p,m,f or p,m,Y2x respectively if symbol,TO=fo is specified, but results in substitution of p,m if symbol,fo is specified because fo cannot be distinguished from f. For example, if Sym1 is defined as 5,4,ZD, SYM1,TO=PD is transformed to 5,4,ZD,TO=PD whereas SYM1,PD is transformed to 5,4,PD. Thus, you should always use symbol,TO=fo rather than symbol,fo.

A symbol for p,m,Y2x results in substitution of p,m,Y2x unless symbol,f or symbol,HEX is specified. A symbol for p,m,Y2xP results in substitution of p,m,Y2xP unless symbol,f or symbol,HEX is specified. A symbol for p,m,f where f is not Y2x or Y2xP results in substitution of p,m unless symbol,edit or symbol,to is specified.

A symbol for 'string' always results in substitution of C'string'.

VLFILL operand: You can use symbols where you can use constants (C'x' and X'yy'). A symbol for 'string' always results in substitution of C'string'.

HEADER1 and HEADER2 operands: You can use symbols where you can use fields (p,m) and non-repeated constants ('xx...x' and C'xx...x', but not n'xx...x' or nC'xx...x'). A symbol for p,m,f always results in substitution of p,m. A symbol for 'string' always results in substitution of C'string'.

TRAILER1 and TRAILER2 operands' You can use symbols where you can use fields (p,m) and non-repeated constants ('xx...x' and C'xx...x', but not n'xx...x' or nC'xx...x'). A symbol for p,m,f results in substitution of p,m if symbol,f is specified or if the symbol is specified outside of the suboperands TOTAL, TOT, MIN, MAX, AVG, SUBTOTAL, SUBTOT, SUB, SUBMIN, SUBMAX and SUBAVG. A symbol for 'string' always results in substitution of C'string'.

SECTIONS operand: The ″HEADER1 and HEADER2 operands″ discussion above also applies to the HEADER3 suboperand of SECTIONS. The ″TRAILER1 and TRAILER2 operands″ discussion above also applies to the TRAILER3 suboperand of SECTIONS.

Outside of the HEADER3 and TRAILER3 suboperands, you can use symbols where you can use fields (p,m). A symbol for p,m,f always results in substitution of p,m.

Example 1

```
 OUTFIL FNAMES=OUT1,
       INCLUDE=(Z_Field1,GT,LIMIT,AND,Any_Format,CH,EQ,C_Field2),
       OUTREC=(12:P_Field1,M0,2X,Any_Format,BI,LENGTH=3,2X,QCON,2X,
                          C_Field2,HEX,2X,Z_Field1,EDIT=('I III IIT.T'),2X,
*  Lookup Table
               Lookup,CHANGE=(6,Entry1,Value1,Entry2,Value2),
                    NOMATCH=(Lookup))
```

will be transformed to:

```
 OUTFIL FNAMES=OUT1,INCLUDE=(22,8,ZD,GT,+12500,AND,12,3,CH,EQ,4,2,CH),O*
               UTREC=(12:30,4,PD,M0,2X,12,3,BI,LENGTH=3,2X,C'Carrie''s *
               Constant',2X,4,2,HEX,2X,22,8,ZD,EDIT=('I III IIT.T'),2X,*
               52,1,CHANGE=(6,X'05',C'Read',X'20',C'Update'),NOMATCH=(5*
               2,1))
```

Example 2

```
 OUTFIL FNAMES=REPORT,
    OUTREC=(6:Branch_Office,24:Profit_or_Loss,M5,LENGTH=20,75:X),
    SECTIONS=(Division,SKIP=P,
       HEADER3=(2:Div_Title,Division,5X,'Page:',&PAGE;,2/,
                   6:BO_Title,24:PL_Title,/,
                   6:BO_Hyphens,24:PL_Hyphens),
      TRAILER3=(6:BO_Equals,24:PL_Equals,/,
                   6:Total,24:TOTAL=(Profit_or_Loss,M5,LENGTH=20),/,
                   6:Lowest,24:MIN=(Profit_or_Loss,M5,LENGTH=20)))
```

will be transformed to:

```
 OUTFIL FNAMES=REPORT,OUTREC=(6:16,13,24:31,10,ZD,M5,LENGTH=20,75:X),SE*
               CTIONS=(3,10,SKIP=P,HEADER3=(2:C'Division:  ',3,10,5X,'P*
               age:',&PAGE;,2/,6:C'Branch Office',24:C'      Profit/(Lo*
               ss)',/,6:C'-------------',24:C'--------------------'),TR*
               AILER3=(6:C'=============',24:C'====================',/,*
               6:C'Total',24:TOTAL=(31,10,ZD,M5,LENGTH=20),/,6:C'Lowest*
               ',24:MIN=(31,10,ZD,M5,LENGTH=20)))
```

Using Symbols in ICETOOL Operators

You can use symbols in the following ICETOOL operators: DISPLAY, OCCUR, RANGE, SELECT, STATS, UNIQUE and VERIFY. In general, you can use symbols in these ICETOOL operators where you can use constants (’string’, n, +n or -n) and fields (p,m,f or p,m). See the discussion of each operator in “Chapter 6. Using ICETOOL” on page 349 for a description of its syntax.

ICETOOL reads the SYMNAMES data set once and uses it for all operators and DFSORT control statements for the run. You can use symbols in operators from the TOOLIN data set or your parameter list. You can also use symbols in DFSORT control statements in xxxxCNTL data sets or in the DFSPARM data set (see “Using Symbols in DFSORT Statements” on page 446 for details).

ICETOOL will list the original operator statements as well as the transformed operator statements.

Details of the use of symbols for each applicable ICETOOL operator is given below followed by a complete ICETOOL example. The example is meant to illustrate variations in how symbols can be used and how they will be transformed. Therefore, the example does not necessarily correspond to how symbols would be used in real applications.

DISPLAY

ON operand:: You can use symbols where you can use fields (p,m,f and p,m). A symbol for p,m,f results in substitution of p,m if symbol,f or symbol,HEX is specified.

BREAK operand: You can use symbols where you can use fields (p,m,f and p,m). A symbol for p,m,f results in substitution of p,m if symbol,f is specified.

TITLE, HEADER, TOTAL, MAXIMUM, MINIMUM, AVERAGE, BTITLE, BTOTAL, BMAXIMUM, BMINIMUM and BAVERAGE operands: You can use symbols where you can use constants ('string'). A symbol for C'string' always results in substitution of 'string'.

OCCUR

ON operand: You can use symbols where you can use fields (p,m,f and p,m). A symbol for p,m,f results in substitution of p,m if symbol,f or symbol,HEX is specified.

TITLE and HEADER operands: You can use symbols where you can use constants ('string'). A symbol for C'string' always results in substitution of 'string'.

HIGHER, LOWER and EQUAL operands: You can use symbols where you can use constants (x, y and v).

RANGE

ON operand: You can use symbols where you can use fields (p,m,f and p,m). A symbol for p,m,f results in substitution of p,m if symbol,f is specified.

HIGHER, LOWER, EQUAL and NOTEQUAL operands: You can use symbols where you can use constants (x, y, v and w).

SELECT

ON operand: You can use symbols where you can use fields (p,m,f and p,m). A symbol for p,m,f results in substitution of p,m if symbol,f is specified.

HIGHER, LOWER and EQUAL operands: You can use symbols where you can use constants (x, y and v).

STATS, UNIQUE and VERIFY

ON operand: You can use symbols where you can use fields (p,m,f and p,m). A symbol for p,m,f results in substitution of p,m if symbol,f is specified.

ICETOOL Example

```
//TOOLSYM JOB A402,PROGRAMMER
//DOIT  EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//SYMNOUT DD SYSOUT=*
//SYMNAMES  DD  *
Rdw,1,4,BI
Account_Code,12,1,CH
Dept_Code,*,=,=
Customer_Name,*,20,CH
SKIP,2
Customer_Balance,*,10,ZD
Customer_Flags,*,1,BI
* Department Codes
```

```
Research,'R'
Marketing,'M'
* Balance Cutoffs
Cancel,+10000       100.00
Gift,+1000000    10,000.00
Stop_Check,-500      -5.00
* Headings and Titles
Title,'Customer Report for'
Head1,'Customer Name'
Head2,'Customer Balance'
Head3,'Customer Flags'
/*
//IN DD DSN=MY.CUSTOMER.INPUT,DISP=SHR
//OUT DD DSN=&O,UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
// DISP=(,PASS)
//LIST1 DD SYSOUT=*
//TOOLIN DD *
  RANGE FROM(IN) ON(Customer_Balance) LOWER(Stop_Check)
  SORT FROM(IN) TO(OUT) USING(CTL1)
  DISPLAY FROM(OUT) LIST(LIST1) BLANK  WIDTH(133) -
    TITLE(Title) DATE(4MD/) PAGE -
    HEADER(Head1) ON(Customer_Name) -
    HEADER(Head2) ON(Customer_Balance,C1) -
    HEADER(Head3) ON(Customer_Flags,HEX)
/*
//CTL1CNTL DD *
  SORT FIELDS=(Customer_Balance,D,Customer_Name,A)
  INCLUDE COND=((Dept_Code,EQ,Research,OR,Dept_Code,EQ,Marketing),
    AND,Customer_Balance,GT,Gift)
/*
```

SYMNOUT will show the following:

```
------- ORIGINAL STATEMENTS FROM SYMNAMES -------
Rdw,1,4,BI
Account_Code,12,1,CH
Dept_Code,*,=,=
Customer_Name,*,20,CH
SKIP,2
Customer_Balance,*,10,ZD
Customer_Flags,*,1,BI
* Department Codes
Research,'R'
Marketing,'M'
* Balance Cutoffs
Cancel,+10000       100.00
Gift,+1000000    10,000.00
Stop_Check,-500      -5.00
* Headings and Titles
Title,'Customer Report for'
Head1,'Customer Name'
Head2,'Customer Balance'
Head3,'Customer Flags'

------------------ SYMBOL TABLE -----------------
Rdw,1,4,BI
Account_Code,12,1,CH
Dept_Code,13,1,CH
Customer_Name,14,20,CH
Customer_Balance,36,10,ZD
Customer_Flags,46,1,BI
Research,C'R'
Marketing,C'M'
Cancel,+10000
Gift,+1000000
Stop_Check,-500
```

```
Title,C'Customer Report for'
Head1,C'Customer Name'
Head2,C'Customer Balance'
Head3,C'Customer Flags'
```

The ICETOOL operators will be transformed to:

```
RANGE FROM(IN) ON(36,10,ZD) LOWER(-500)

  SORT FROM(IN) TO(OUT) USING(CTL1)

DISPLAY FROM(OUT) LIST(LIST1) BLANK WIDTH(133)-
TITLE('Customer Report for') DATE(4MD/) PAGE-
HEADER('Customer Name') ON(14,20,CH)-
HEADER('Customer Balance') ON(36,10,ZD,C1)-
HEADER('Customer Flags') ON(46,1,HEX)
```

The DFSORT control statements in CTL1CNTL will be transformed to:

```
 SORT FIELDS=(36,10,ZD,D,14,20,CH,A)
 INCLUDE COND=((13,1,CH,EQ,C'R',OR,13,1,CH,EQ,C'M'),AND,36,10,ZD,GT,+10*
               00000)
```

Notes for Symbols

- EFS programs cannot be used with symbol processing.
- DFSORT or ICETOOL scans each SYMNAMES statement for errors, and prints an error message for the first error detected. A marker ($) is printed directly below the SYMNAMES statement near the error, if appropriate.

  Scanning stops at the first error, and then continues with the next SYMNAMES statement. However, once an error is detected, positions generated by using an asterisk (*) for p or POSITION,symbol in subsequent statements will not be checked for errors. DFSORT and ICETOOL terminate after all SYMNAMES statements are scanned if an error is detected in any statement.
- If DFSORT or ICETOOL detects an error in a control statement or operator statement during the substitution phase (that is, while attempting to substitute values for symbols), it may either:
  - print the original statement in error followed by a $ marker (if appropriate) and an error message, continue the substitution phase with the next statement and terminate when the substitution phase is complete, or
  - stop performing substitution for the statement in error, continue with the next statement and let processing after the substitution phase handle the error. It is possible in this case for a symbol, rather than a substituted value, to appear in a transformed statement.
- If the substitution phase is successful, DFSORT and ICETOOL will substitute values for symbols wherever symbols are allowed. Substituted values which are invalid for a particular statement or operand will be detected after the substitution phase. This makes it easier to determine the cause of the error. For example, if SYMNAMES contains:

  ```
  Sym1,5,4,ZD
  Con1,'1234'
  Con2,1234
  ```

  the statement:

  ```
   INCLUDE COND=(Sym1,EQ,Con1)
  ```

  will be transformed to the following during the substitution phase:

  ```
   INCLUDE COND=(5,4,ZD,EQ,C'1234')
  ```

An ICE114A message with a $ marker under C'1234' will then be issued for the statement because a ZD field cannot be compared to a character constant. In this example, the error could be fixed by using Con2 (a decimal constant) in the statement instead of Con1 or by redefining Con1 as a decimal constant.

- If you use a temporary or permanent message data set, it is best to specify a disposition of MOD to ensure you see all messages and control statements in the message data set. In particular, if you use symbols processing and do not use MOD, you will not see the original control statements unless Blockset is selected.
- If you rearrange your records in any way (for example, using E15, E35, INREC, OUTREC or OUTFIL) and want to use symbols for the rearranged records, be sure to use symbols that map to the new positions of your fields. For example, if you use a SYMNAMES data set with the following statements:

  ```
  Field1,1,5,ZD
  Field2,*,6,ZD
  Field3,*,3,ZD
  Field4,*,4,ZD
  ```

  for this INREC statement:

  ```
   INREC FIELDS=(Field2,Field4)
  ```

  the resulting records will only contain Field2 and Field4. If you want to use symbols for the rearranged records (for example, in a SORT statement), you will need to use a SYMNAMES data set with symbols that map to the rearranged records, such as:

  ```
  New_Field2,1,6,ZD
  New_Field4,*,4,ZD
  ```

  If you use unique symbols for the rearranged fields, as in the example above, you can concatenate the old and new symbol data sets together and use the old and new symbols where appropriate, as in this example:

  ```
   INREC FIELDS=(Field2,Field4)
   SORT FIELDS=(New_Field2,A,New_Field4,A)
  ```

Chapter 8. Using Extended Function Support

Using EFS 456
Addressing and Residence Mode of the EFS Program 456
How EFS Works 457
DFSORT Program Phases 457
DFSORT Calls to Your EFS Program 458
Initialization Phase 460
Input Phase 462
Termination Phase 462
What You Can Do with EFS. 463
Opening and Initializing Data Sets 464
Examining, Altering, or Ignoring Control Statements 464
Processing User-Defined Data Types with EFS Program User Exit Routines 466
Supplying Messages for Printing to the Message Data Set 466
Terminating DFSORT 466
Closing Data Sets and Housekeeping 466
Structure of the EFS Interface Parameter List 466
Action Codes 468
Control Statement Request List 469
Control Statement String Sent to the EFS program 469
Special Handling of OPTION and DEBUG Control Statements 471
Control Statement String Returned by the EFS Program 471
Rules for Parsing 471
EFS Formats for SORT, MERGE, INCLUDE, and OMIT Control Statements 472
D1 Format on FIELDS Operand 473
D2 Format on COND Operand 473
Length of Original Control Statement 475
Length of the Altered Control Statement 475
EFS Program Context Area 475
Extract Buffer Offsets List 475
Record Lengths List 476
Information Flags 476
Message List 478
EFS Program Exit Routines. 478
EFS01 and EFS02 Function Description 479
EFS01 User Exit Routine. 479
EFS01 Parameter List. 480
EFS02 User Exit Routine. 480
EFS02 Parameter List. 482
Addressing and Residence Mode of EFS Program User Exit Routines 482
EFS Program Return Codes You Must Supply 483
Record Processing Order 483
How to Request a SNAP Dump 486
EFS Program Example 486
DFSORT Initialization Phase: 486
Major Call 1 486
Major Call 2 487
Major Call 3 488
DFSORT Termination Phase 488
Major Call 4 488

© Copyright IBM Corp. 1973, 2000

Major Call 5 488

Using EFS

Like the user exits described in Chapter 4. Using Your Own User Exit Routines, the DFSORT Extended Function Support (EFS) interface is a means by which you can pass run-time 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 5665-360, Release 2.0, or you can use locale processing with the appropriate locale.

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.

Notes:

1. DFSORT does not support EFS programs for Conventional merge or tape work data set sort applications.
2. VLSHRT is not allowed if EFS processing is in effect and an EFS01 or EFS02 exit routine is provided by the EFS program.
3. If you use locale processing for SORT, MERGE, INCLUDE, or OMIT fields, you must not use an EFS program. DFSORT's locale processing may eliminate the need for an EFS program. See "OPTION Control Statement" on page 134 for information related to locale processing.
4. If you use symbol processing, you must not use an EFS program.

The DFSORT target library, SICEUSER, contains a mapping macro called ICEDEFS, which provides a separate Assembler DSECT for the EFS parameter list.

Addressing and Residence Mode of the EFS Program

You can design the EFS program to reside and run above or below 16MB 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 run below 16MB virtual, 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 run-time, and communicates information back and forth across the interface as your EFS program runs.

You can activate your EFS program during run-time 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 16)
- On the PARM parameter of your EXEC statement when you use job control language to invoke DFSORT (see “Specifying EXEC/DFSPARM PARM Options” on page 29)
- On the OPTION program control statement (see “OPTION Control Statement” on page 134).

See “Appendix B. Specification/Override of DFSORT Options” on page 549 for override information. Figure 42 illustrates the role of the EFS interface in linking DFSORT’s processing capabilities to the EFS program you write.

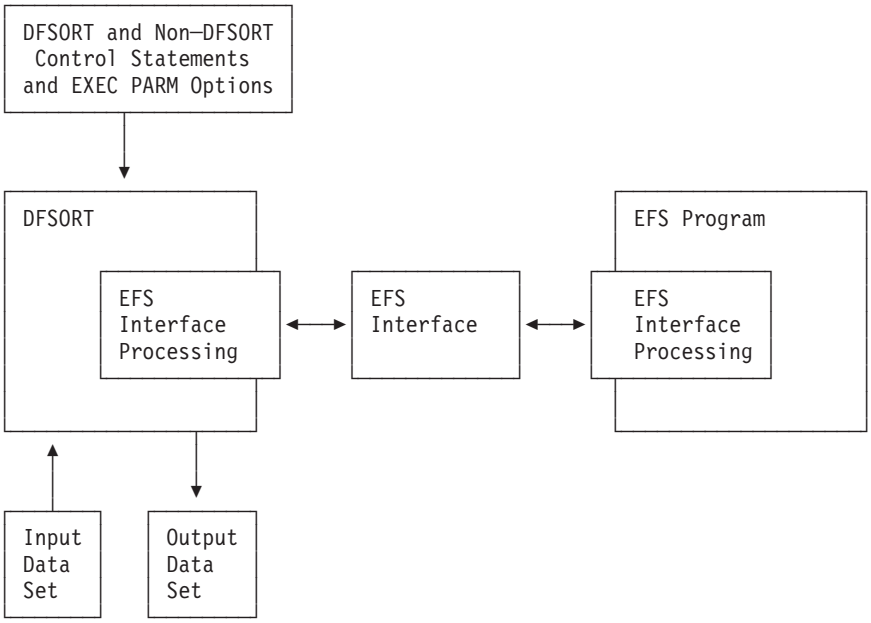


Figure 42. Relationship Between DFSORT and an EFS Program

DFSORT Program Phases

A DFSORT program phase is a large DFSORT component designed to perform a specific task such as writing the output file. An EFS program is called at various points during run-time of DFSORT program phases in performing the variety of tasks capable with an EFS program. When the termination phase is completed, DFSORT returns control to the operating system or invoking program.

How EFS Works

EFS processing can be invoked during the initialization, input, and termination phases of DFSORT. DFSORT always calls the EFS program during the initialization phase.

During the input phase, DFSORT reads input records, and performs any INCLUDE or OMIT statement 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 2 and 3. Refer to Figure 43 on page 459 and Figure 44 on page 460 as you read this section for illustrations of the relationships between program phases and calls during run-time.

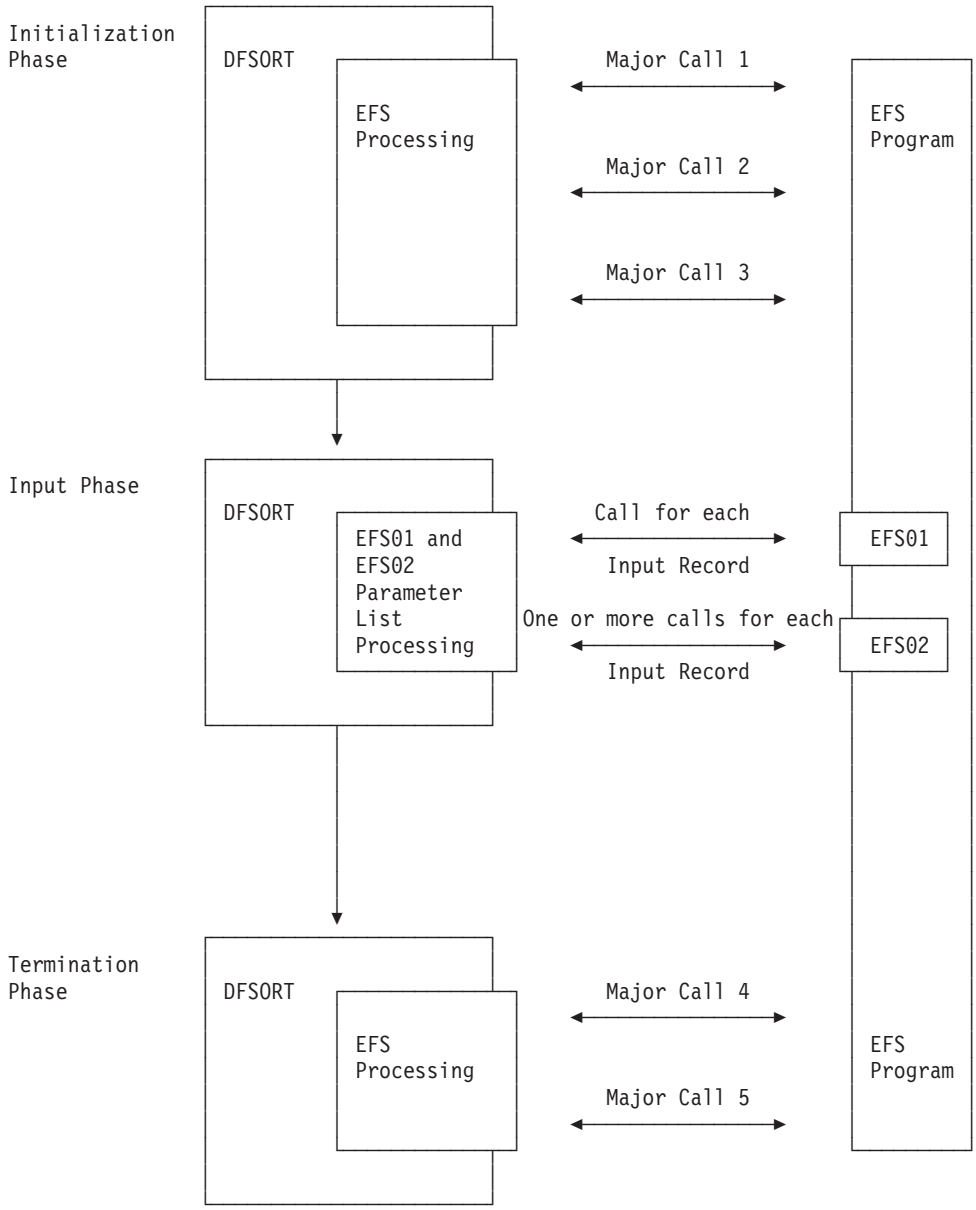


Figure 43. EFS Program Calls for a Sort. The figure also shows the calls to the EFS program EFS01 and EFS02 exit routines.

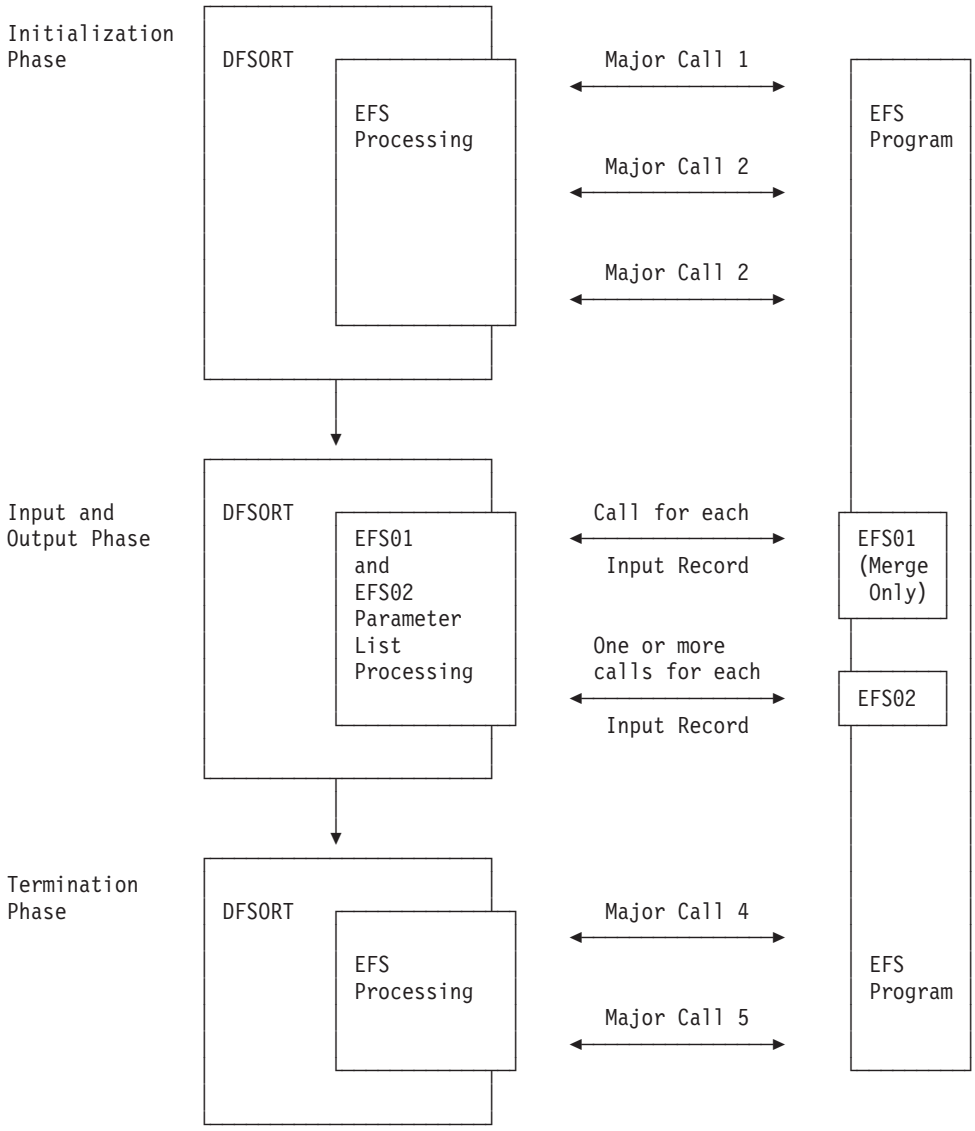


Figure 44. EFS Program Calls for a Merge or Copy. The figure also shows the calls to the EFS program EFS01 and EFS02 exit routines.

Initialization Phase

DFSORT runs 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.

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

  Note: OUTFIL statements cannot be requested by an 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: 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 calls your EFS program once for each control statement or EXEC PARM you request.

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
- 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 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, or include or omit user-defined data types, 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.

  Note: OUTFIL statements cannot be passed to an EFS program or returned from an EFS program to be parsed.
- 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 can provide 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 runs 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 for each record, according to the 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 to which the EFS01 routine must move all EFS control fields. See “EFS01 User Exit Routine” on page 479 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 “EFS02 User Exit Routine” on page 480 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 runs Major Calls 4 and 5 during the termination phase. Only one call is made at each of these Major Calls.

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 run, the ESTAE routine runs them. If an EFS abend occurs during Major Call 1, the ESTAE routine does not run Major Calls 4 and 5. See “Appendix E. DFSORT Abend Processing” on page 595 for more information about ESTAE.

Major Call 4: The EFS program provides any final 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 4 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 user exit routines EFS01 and EFS02.

Table 57. 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	
Supplying messages to DFSORT for printing to the message data set	EFS Program		EFS Program

Table 57. Functions of an Extended Function Support (EFS) Program (continued)

EFS Program Functions	Initialization Phase	Input Phase	Termination Phase
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. OUTFIL statements cannot be requested by an EFS program.

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. OUTFIL statements cannot be passed to an EFS program or returned from an EFS program to be parsed.

Refer to Figure 45 on page 465 for an illustration of the control statement processing sequence used when an EFS program is activated.

The same override rules apply to control statements and parameters returned from an EFS program as apply to the original control statements and parameters.

For example, a STOPAFT parameter added to the SORT statement by an EFS program is overridden by a STOPAFT parameter in an OPTION statement in the same way as if the SORT statement originally contained the STOPAFT parameter.

See “Appendix B. Specification/Override of DFSORT Options” on page 549 for full override details.

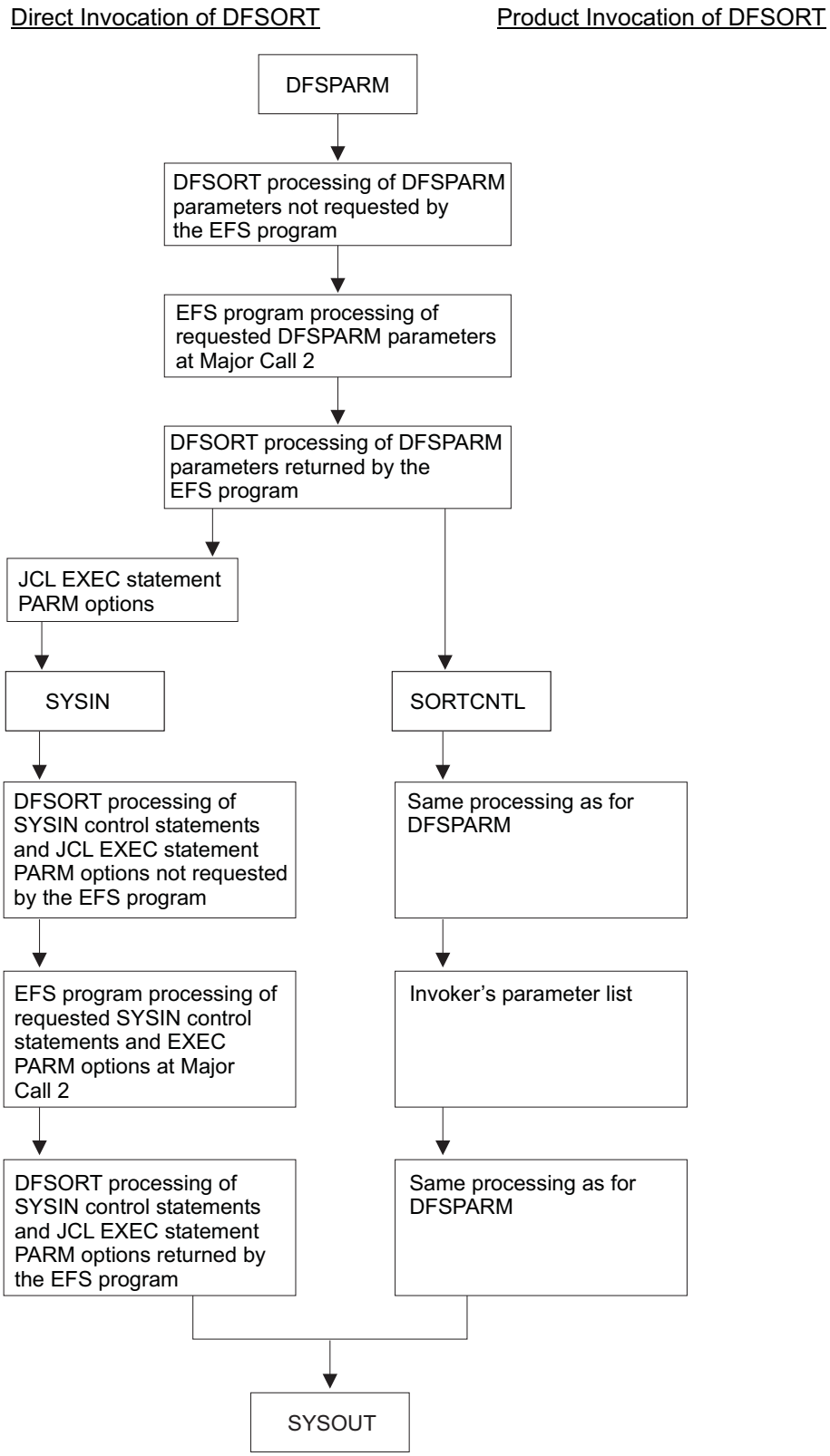


Figure 45. Control Statement Processing Sequence

Processing User-Defined Data Types with EFS Program User Exit Routines

You can write your EFS program to provide two user exit routines to perform various tasks during run-time.

Your EFS program user exit routines can:

- Process user-defined data types. Your EFS program can provide an EFS01 routine to alter any control field of an input record.
- Include or omit records based on user-defined data types. Your EFS program can provide an 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 to the message data set with:

- The LISTX operator of the ICEMAC macro (see “Installation Defaults” on page 16)
- The LISTX or NOLISTX operators in the PARM field of the JCL EXEC statement (see “Specifying EXEC/DFSPARM PARM Options” on page 29)
- 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 16MB virtual, and remains accessible while the EFS program is active, although DFSORT might change its storage location during run-time 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 46 on page 467 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          │   │ Control statement    │
      │                                  │   │ request list         │
      │──────────── 4 bytes ─────────────│   └── ** bytes ──────────┘
      │ Address of                       │
      │  original Control Statement      │   ┌──────────────────────┐
      │  including all keywords and      │   │ Original control     │
      │  corresponding subparameters     │──►│ statement string     │
      │                                  │   └── * bytes ───────────┘
      │──────────── 4 bytes ─────────────│
      │ Address of                       │
      │  modified Control Statement      │   ┌──────────────────────┐
      │  including all keywords and      │   │ Modified control     │
      │  corresponding subparameters     │──►│ statement string     │
      │                                  │   └── * bytes ───────────┘
      │──────────── 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 ─────────────┘
```

Figure 46. EFS Interface Parameter List (Part 1 of 2)

Structure of the EFS Interface Parameter List

```
Address of
 Extract buffer offsets
 (zeros if no EFS fields exist)
------------- 4 bytes -------------
Address of                          ---> Record lengths
 Record lengths list                      list
------------- 4 bytes -------------        -- 8 bytes --
RESERVED
------------- 4 bytes -------------
RESERVED
------------- 4 bytes -------------
RESERVED
------------- 4 bytes -------------
Information flags
------------- 4 bytes -------------
Address of
 message list
 (zeros if none)
------------- 4 bytes -------------
RESERVED
------------- 4 bytes -------------
RESERVED
------------- 4 bytes -------------
RESERVED
------------- 4 bytes -------------
    | Address of
f   |  EFS01 extract routine
    |  (zeros if none)
------------- 4 bytes -------------
    | Address of
f   |  EFS02 INCLUDE/OMIT routine
    |  (zeros if none)
------------- 4 bytes -------------
List end indicator (X'FFFFFFFF')
------------- 4 bytes -------------

**  — Length determined by length fields in list
*   — Length determined by corresponding length field
```

Figure 46. EFS Interface Parameter List (Part 2 of 2)

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 several 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.

OUTFIL statements cannot be requested by an EFS program.

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

The asterisk (*) 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 81, 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 *JCL User's Guide* and *JCL Reference* 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 471).

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 471), 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 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 476 for further details).

OUTFIL statements cannot be returned from an EFS program to be parsed.

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 77, 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 472).

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 on). 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.
- Comment statements, blank statements, and remarks are not 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/DFSPARM PARM Options” on page 29.

The following operands will be ignored by DFSORT if returned by an EFS program on the OPTION control statement:

EFS
LIST
NOLIST
LISTX
NOLISTX
LOCALE
MSGDDN
MSGDD
MSGPRT
SMF
SORTDD
SORTIN
SORTOUT
USEWKDD

The following EXEC PARM options will be ignored by DFSORT if returned by an EFS program:

EFS
LIST
NOLIST
LISTX
NOLISTX
LOCALE
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 FIELDS parameter of the SORT or MERGE statement
- D2 with the COND parameter of the INCLUDE or OMIT statement.

Use D1 and D2 to reflect data types that require special processing by EFS program exit routines EFS01 and EFS02, respectively. You cannot specify D2 format with the INCLUDE or OMIT parameters of the OUTFIL statement.

D1 Format on FIELDS Operand

The syntax for the SORT and MERGE statements usingthe 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.

Table 58 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.

Table 58. D1 Format Returned by an EFS Program

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

D2 Format on COND Operand

Following is the syntax for the INCLUDE or OMIT statements usingthe D2 format on the COND operand.

```
                                  ┌──────────────────,──┬─AND─┬──,──────────────────┐
                                  │                     └─OR──┘                     │
                                  │   ┌──(──┐  ┌──────,──┬─AND─┬──,──────────┐ ┌─)─┐│
                                  ↓↓  │     ↓  │        └─OR──┘             ↓ │   ↓│
►►──┬─INCLUDE─┬─COND=─(──────────┴──mc,mm,mf,─operator─,─┬─mc,mm,mf─┬──────────┴──)─►◄
   └─OMIT────┘                                          ├─constant─┤
                                                        └─mask─────┘
```

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

operator
a valid DFSORT comparison or bit logic operator

constant
a valid DFSORT decimal, character, hexadecimal or bit constant.

mask a valid DFSORT hexadecimal or bit string

Table 59 on page 475 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 Table 59 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 might exceed the record length.
  - Any valid DFSORT constant or mask is allowed.
  - If COND=(mc1,mm1,mf1,operator,mc2,mm2,mf2) is used, both mf1 and mf2 must be D2.
  - CHALT has no effect on EFS fields.
  - FORMAT=D2 is not allowed.

Table 59. Correlator Identifier and D2 Format Returned by an EFS Program

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.

Length of Original Control Statement

The control statement includes the first byte of thecontrol 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 areathat 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 toyour 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:

Structure of the EFS Interface Parameter List

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 inputand output record lengths. You must be aware of the possible change in record size during run-time (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 in the figure that follows:

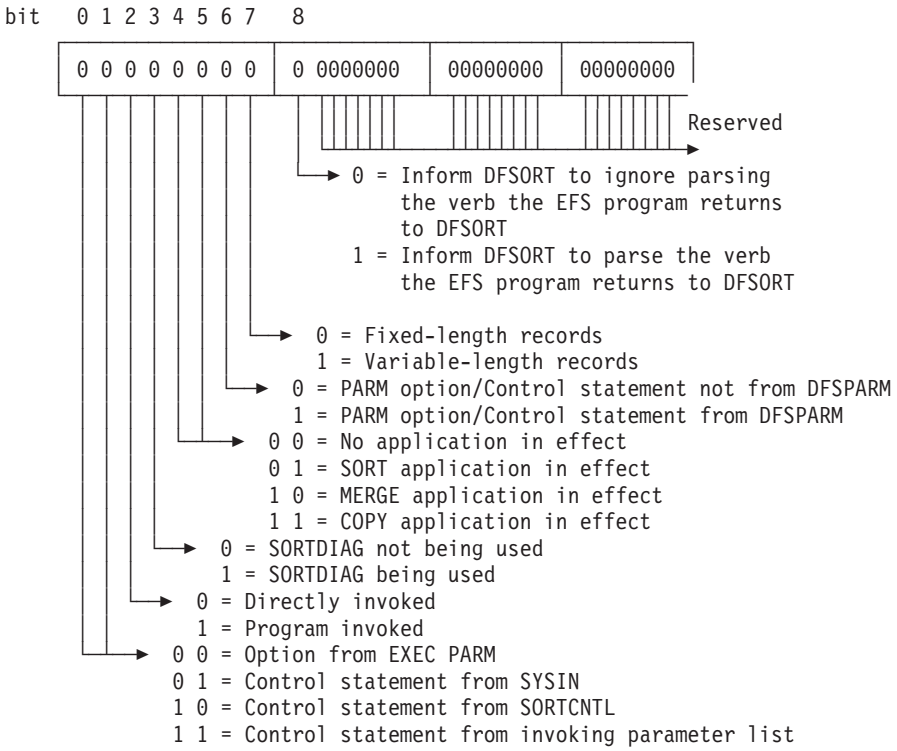


Figure 47. Information Flags

Bit **Description**

Bits 0 and 1
Indicate the source of the control statement being processed. Information flags 0 and 1 are set by DFSORT before a call 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 is set by DFSORT before Major Call 1 to the EFS program.

Bits 4 and 5
Indicate the DFSORT function being run. Information flags 4 and 5 are set by DFSORT before each call 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 is set by DFSORT before each call 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 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 is managed by the LISTX/NOLISTX parameters (see “OPTION Control Statement” on page 134 for further details). Information flag 8 is set by the EFS program before returning to DFSORT from each call 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 first prints the control statement in error and then prints 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:

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

An asterisk (*) indicates that the length is determined by the corresponding length field.

DFSORT imposes no restrictions on the format of the messages 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 *Messages, Codes and Diagnosis*

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 buffer area 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 enables an EFS program to provide user exit routines to perform functions beyond the capabilities of DFSORT control statements.

The EFS program can provide user exit routine EFS01 to supplement the function of the DFSORT SORT/MERGE control statements and can provide user 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 48).
- The routines must use the described interfaces (see “EFS01 Parameter List” on page 480 and “EFS02 Parameter List” on page 482).

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 48. DFSORT Register Convention

EFS01 User Exit Routine

Processing of user-defined data types with the EFS01 exit routinerequires using the function that alters 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 469 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 EFS01 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 472 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 routine move the data indicated by the EFS control fields. See “Extract Buffer Offsets List” on page 475 for further details. At Major Call 3, the EFS program must return the address of the EFS01 routine to DFSORT.

During the input phase, DFSORT calls the EFS01 routine for each input record. The EFS01 exit routine must move all data indicated by the EFS control fields, specified 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 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.

DFSORT will do sort or merge processing using the data in the extract buffer, and will treat the data as binary data.

EFS01 Parameter List

DFSORT sends three words to the EFS01 user 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 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 User Exit Routine

Including or omitting records based on user-defined data typeswith the EFS02 user 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 469 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 EFS02 routine (the fields defined with the D2 format are also known as EFS compare fields). See “EFS Formats for SORT, MERGE, INCLUDE, and OMIT Control Statements” on page 472 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 472 for further details.

At Major Call 3, the EFS program must return the address of the EFS02 routine to DFSORT.

The EFS02 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 one or more times 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 49 repeats Table 59 on page 475 to illustrate an example of the calling sequences to an EFS02 by DFSORT.

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: the calling sequence to EFS02 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 49. 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.

Addressing and Residence Mode of EFS Program User Exit Routines

DFSORT supplies the following features to allow residence above or below 16MB 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 user 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 user 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 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 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 user exit routines returns a return code of 16, DFSORT will skip interim Major Calls, where applicable, to the EFS program or user 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 usingEFS is similar to processing without it. Figure 50 on page 484 illustrates the record processing sequence for a sort or merge and Figure 51 on page 485 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 8 with the following exceptions:

- When record processing is done for an INCLUDE or OMIT control statement, an EFS02 user 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 user exit routine is called to perform the extraction process for EFS control fields.

Record Processing Order

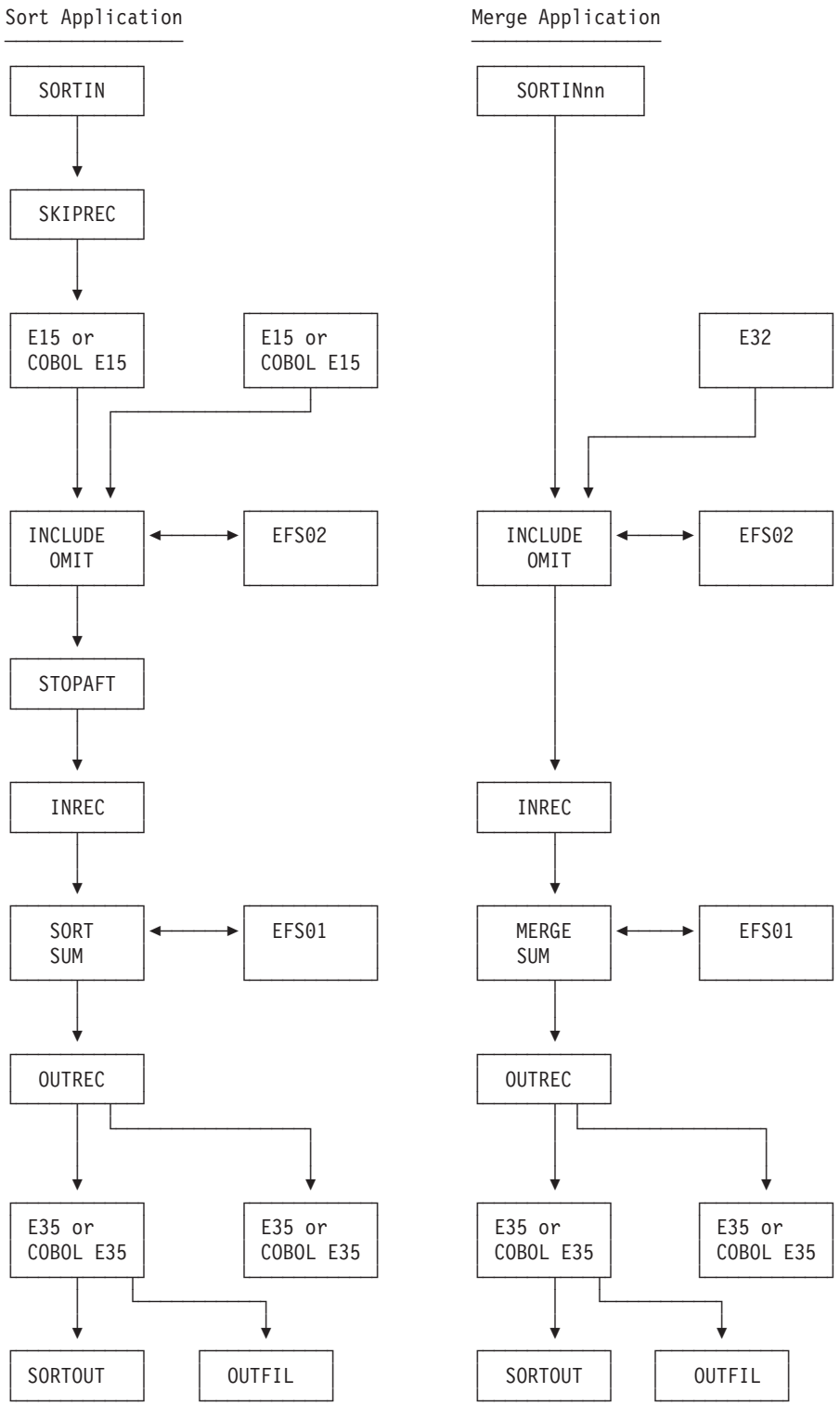


Figure 50. EFS Record Processing Sequence for a Sort or Merge

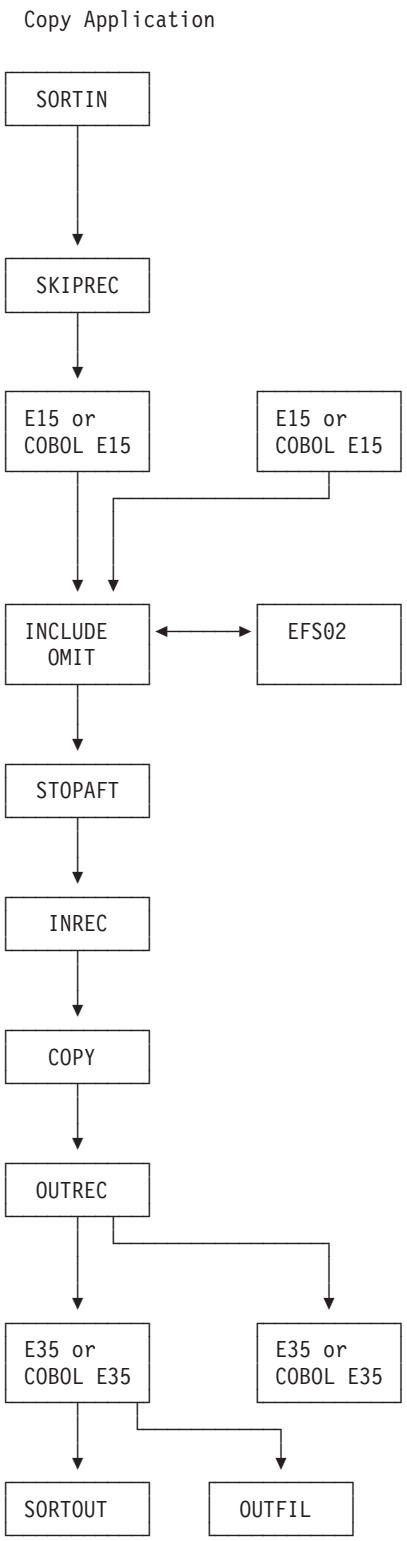


Figure 51. EFS Record Processing Sequence for a Copy

How to Request a SNAP Dump

You can request a SNAP dump for diagnostic purposes before or afterany 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 87 for the correct syntax.

EFS Program Example

The following example shows how an EFS program can be used to change control statements at run-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 application 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=3390
/*
```

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 calls EFS program EFSPGM at Major Call 1, 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 EFS program EFSPGM 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=3390
- The length of the *original* OPTION control statement, including all operands and corresponding subparameters

  The original control statement string is 31 bytes long.

DFSORT calls EFS program EFSPGM at Major Call 2, 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.

Table 60 shows the original control statement sent to EFS program EFSPGM and the altered control statement returned by EFS program EFSPGM.

Table 60. Original and Altered Control Statements

Original OPTION control statement sent to EFSPGM
OPTION STOPAFT=30,DYNALLOC=3390
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

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 calls EFS program EFSPGM at Major Call 3, 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 user exit routine is not used.

  Because no INCLUDE control statement was supplied (with EFS fields), the EFS02 user 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 calls EFS program EFSPGM at Major Call 4, 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 calls EFS program EFSPGM at Major Call 5, and EFSPGM does not set any fields in the EFS interface parameter list but sets general register 15 to zero.

Chapter 9. Improving Efficiency

Improving Performance 490
Design Your Applications to Maximize Performance 490
Directly Invoke DFSORT Processing 490
Plan Ahead When Designing New Applications. 491
Efficient Blocking. 491
Specify Efficient Sort/Merge Techniques 491
Sorting Techniques 491
Merging Techniques 492
Specify Input/Output Data Set Characteristics Accurately 492
Input File Size. 492
Variable-Length Records 492
Direct Access Storage Devices 492
Use Sequential Striping 493
Use Compression 493
Use DFSMSrmm-Managed Tapes 493
Use SmartBatch Pipes 493
Use VIO in Expanded Storage. 493
Specify Devices that Improve Elapsed Time. 494
Use Options that Enhance Performance 494
CFW 494
COBEXIT 494
DSA 494
DSPSIZE 494
FASTSRT 494
SDB 495
HIPRMAX 495
Use DFSORT's Fast, Efficient Productivity Features. 496
INCLUDE or OMIT, STOPAFT, and SKIPREC 496
OUTFIL 496
LOCALE. 496
SUM 496
ICETOOL 497
Avoid Options that Degrade Performance. 497
CKPT 497
EQUALS. 497
EQUCOUNT 497
LOCALE. 497
NOCINV 497
NOBLKSET 497
VERIFY 497
Tape Work Data Sets 497
User Exit Routines 497
Dynamic Link-Editing 497
EFS Programs 497
Use Main Storage Efficiently 498
Tuning Main Storage 498
Releasing Main Storage 500
Allocate Temporary Work Space Efficiently 501
Direct Access Work Storage Devices 501
Virtual I/O for Work Data Sets 502
Tape Work Storage Devices 502
Use Hipersorting. 503
Sort with Data Space 503

© Copyright IBM Corp. 1973, 2000

Use ICEGENER Instead of IEBGENER 504
ICEGENER Return Codes 506
Use DFSORT's Performance Booster for The SAS System 506
Use DFSORT's BLDINDEX Support. 507

Improving Performance

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 applications to maximize performance:
  - Directly invoke DFSORT processing
  - Plan ahead when designing new applications
  - Specify efficient sort/merge techniques
  - Specify input/output data set characteristics accurately
  - Use sequential striping
  - Use compression
  - Use DFSMSrmm-managed tapes
  - Use SmartBatch pipes
  - Use VIO in expanded storage
  - Specify devices that improve elapsed time
  - Use options that enhance performance
  - Use DFSORT's fast, efficient productivity features
  - Avoid options that degrade performance.
- Use main storage efficiently
- Allocate temporary work space efficiently
- Use Hipersorting
- Sort with data space
- Use ICEGENER instead of IEBGENER
- Use DFSORT's Performance Booster for The SAS System
- Use DFSORT's BLDINDEX support.

The DFSORT *Tuning Guide* provides additional information related to many of the topics covered in this chapter.

Design Your Applications to Maximize Performance

Even though DFSORT automatically optimizes performance when your application is run, you can still improve efficiency by using specifications and options that permit DFSORT to make the best possible use of available resources.

Directly Invoke DFSORT Processing

You can enhance performance by invoking DFSORT withJCL 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, SUM, and OUTFIL. You can eliminate records from input files, reformat records to eliminate unwanted fields, combine records arithmetically, and create reports, without requiring routines from other programs.

Plan Ahead When Designing New Applications

You should consider several factors when designing newapplications. 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
- If you know that a fixed-point control field always contains positive values, specify it as a binary field.
- If you know that a packed decimal or zoned decimal control field always contains positive values with the same sign (for example, X'C'), specify it as a binary field.
- 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.
- Avoid using locale processing if your SORT, MERGE, INCLUDE, or OMIT character fields can be processed using the binary encoding of the data.

Efficient Blocking

You can improve the performance of DFSORT significantly by blocking your input and output records efficiently. Whenever possible, use system-determined optimum block sizes for your data sets.

For more information about letting DFSORT select system-determined optimum block sizes for your output data sets, see the discussion of the SDB option in “OPTION Control Statement” on page 134.

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 isthe 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 502.

The Blockset and Peerage/Vale techniques can be used only with DASD work data sets. These techniques are discussed below.

Blockset 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 “Allocate Temporary Work Space Efficiently” on page 501.

- 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 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 Blocksetand 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 Merging Technique: When the conditions for use of the Blockset merging technique are not met, DFSORT uses the Conventional merge technique, which is less efficient.

Specify Input/Output Data Set Characteristics Accurately

DFSORT uses the information given it (aboutthe 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.

Input File Size

When DFSORT has accurate information about the input file size, it can make the most efficient use of both main storage and intermediate work storage. See "File Size and Dynamic Allocation" on page 543 for information about when and how to specify the input file size.

Variable-Length Records

When the input data set consists of variable-length recordsand dynamic allocation of intermediate data sets is used, specify the average record length as accurately as possible using AVGRLEN=n in the OPTION statement.

Direct Access Storage Devices

System performance is improved if storage is specified in cylindersrather than tracks or blocks. Storage on sort work data sets will be readjusted to cylinders if possible. The number of tracks per cylinder for direct access devices is shown in Table 61.

Table 61. Number of Tracks per Cylinder for Direct Access Devices

Device	Tracks per Cylinder
3380	15
3390	15
9345	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 sets 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
- Low ratio of available storage to input file size.

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.

Use Sequential Striping

The use of sequential striping can significantly reduce the elapsed time DFSORT spends reading and writing data. We recommend using sequential striping for your DFSORT input and output data sets as a way to improve elapsed time performance.

Use Compression

The use of compression can significantly reduce the DASD storage required for many types of data and the resulting time DFSORT spends reading and writing that data. We recommend using compression for your DFSORT input and output data sets as a way to improve elapsed time performance.

Use DFSMSrmm-Managed Tapes

The use of tapes managed by DFSMSrmm allows DFSORT to obtain accurate information about input file size and data set characteristics. This can result in improved performance and more efficient use of both main storage and intermediate work storage.

Use SmartBatch Pipes

The use of SmartBatch input and output pipes can significantly reduce elapsed time as a result of the parallelism inherent in piping the data from “writer” to concurrent “reader” jobs. For example, if a SORTOUT data set is piped, DFSORT output processing can be overlapped with the receiving job’s input processing. In addition, because a pipe is a virtual storage queue rather than a DASD or tape data set, data transfer time and elapsed time can be reduced significantly.

We recommend using SmartBatch pipes for your DFSORT input and output data sets, when appropriate, as a way to improve elapsed time and data transfer time.

Use VIO in Expanded Storage

Temporary non-VSAM input and output data sets can be held in expanded storage using Virtual I/O (VIO). Because expanded storage is used rather than a DASD or tape data set, data transfer time and elapsed time can be reduced significantly (provided that the entire data set can fit in the available expanded storage). For example, if a SORTOUT data set is allocated as a temporary VIO data set in expanded storage, DFSORT’s output processing as well as the receiving job’s input processing can show improved performance.

We recommend using VIO in expanded storage for your DFSORT input and output data sets, when appropriate, as a way to improve elapsed time and data transfer time.

Note that VIO is generally not recommended for work data sets, as discussed in “Virtual I/O for Work Data Sets” on page 502.

Specify Devices that Improve Elapsed Time

To get the best elapsed time improvement when using DASD with DFSORT, you should use 3390s for SORTIN, SORTWKdd, SORTOUT, and OUTFIL data sets. A mixture of 3380s and 3390s for these data sets might not result in the same elapsed time improvement you would get with all 3390s; this is indirectly affected by DFSORT processing techniques, but is primarily due to the lower performance characteristics of the 3380 in relation to the 3390.

The exact elapsed time improvement you see when using 3390s depends on the processing techniques used by DFSORT for the particular run, and which data sets (SORTIN, SORTWKdd, SORTOUT, OUTFIL) reside on 3390s. We recommend that if you cannot use all 3390s, you use 3390s for SORTIN, SORTOUT, and OUTFIL data sets in preference to SORTWKdd data sets.

Use Options that Enhance Performance

To obtain optimum performance, you can fine-tune theoptions specified during installation and at run time. Several options that can enhance performance are described below.

CFW

To improve Blockset sorting performance by taking advantage of the cached 3990 Storage Controls, specify CFW on the DEBUG control statement or CFW=YES as the installation default (CFW=YES is the IBM-supplied default).

COBEXIT

To take advantage of the COBOL II interface with DFSORT andenhance performance, specify COBEXIT=COB2 on the OPTION control statement or define it as the installation default when you run user exits compiled with VS COBOL II, COBOL for MVS & VM or COBOL for OS/390 & VM.

DSA

Performance can be improved for Blockset sort applications by using Dynamic Storage Adjustment (DSA).

The DSA installation parameter sets the maximum amount of storage available to DFSORT for dynamic storage adjustment of a Blockset sort application when SIZE/MAINSIZE=MAX is in effect. If you specify a DSA value greater than the TMAXLIM value, you allow DFSORT to use more storage than the TMAXLIM value if doing so should improve performance. DFSORT only tries to obtain as much storage as needed to improve performance up to the DSA value.

DSPSIZE

Performance can be improved for sort applications that use the Blockset technique by using dataspace sorting.

The DSPSIZE parameter sets the maximum amount of data space that will be used for dataspace sorting. The default, DSPSIZE=MAX, permits DFSORT to select the maximum amount of data space that it uses based on the size of the file being sorted and the paging activity of your system.

FASTSRT

By specifying the VS COBOL II or later FASTSRT compiler 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/output (see Figure 52 on page 495). For more details, see the VS COBOL II or later 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, OUTFIL, 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.

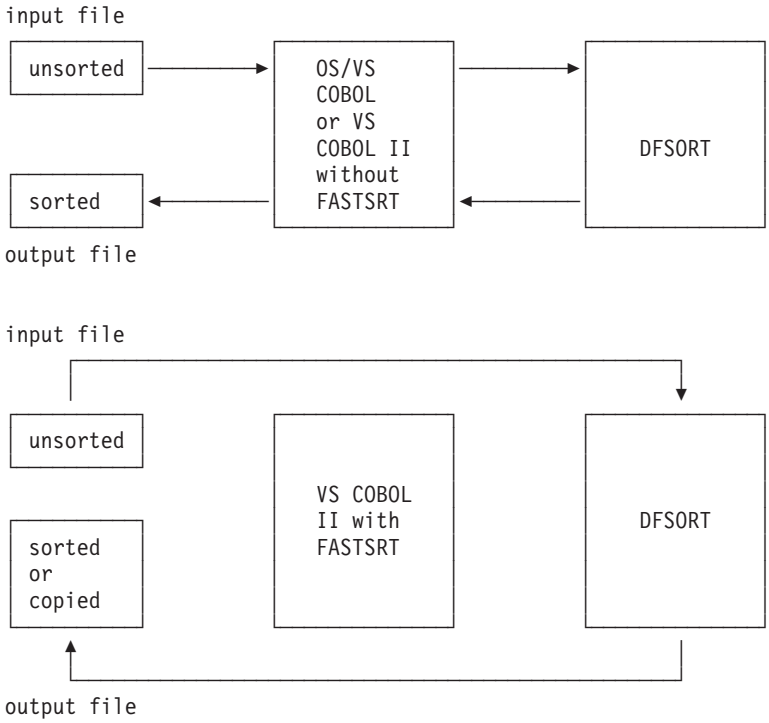


Figure 52. Faster Sorting with VS COBOL II

SDB

To improve Blockset elapsed time, and DASD and tape utilization, specify SDB=LARGE as your site's installation default (SDB=INPUT is the IBM-supplied default). SDB=LARGE allows DFSORT to select the system-determined optimum block size for your DASD and tape output data sets, when appropriate.

HIPRMAX

Blockset sorting performance can be improved by using Hiperspace along with DASD for temporary storage.

The HIPRMAX parameter sets the maximum amount of Hiperspace to be committed during a run. Specifying HIPRMAX=OPTIMAL allows DFSORT to optimize the maximum amount of Hiperspace to be committed during a run, subject to other system and concurrent Hipersorting activity throughout the run. Total Hipersorting activity on a system can be further limited by the DFSORT installation

options EXPMAX, EXPOLD, and EXPRES. See the description of HIPRMAX in “OPTION Control Statement” on page 134 for more information.

Use DFSORT’s Fast, Efficient Productivity Features

DFSORT offers a rich set of fast, efficient productivity features. These features can eliminate the up-front costs of writing and debugging your own code to perform various tasks, and will perform those tasks more efficiently. The functional capabilities of each of the features listed below is described in detail elsewhere in this book. This section highlights the performance aspects of these features.

INCLUDE or OMIT, STOPAFT, and SKIPREC

You can use the INCLUDE or OMIT statement and the STOPAFTand SKIPREC options to reduce the number of records to be processed, which can reduce processor and data transfer time.

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

OUTFIL

If you need to create multiple output data setsfrom the same input data set, you can use OUTFIL to read the input data set only once, thus improving performance. OUTFIL can be used for sort, merge, and copy applications to provide sophisticated filtering, editing, conversion, lookup and replace, and report features.

If you are creating only a single output data set and do not need the features of OUTFIL, use SORTOUT rather than OUTFIL for best performance.

LOCALE

You can use the LOCALE option to sort and merge character data basedon collating rules in an active locale; this enables you to obtain results with DFSORT that were previously possible only through pre- and/or post-processing of your data. By eliminating such costly processing, you can save time and processing resources.

SUM

You can improve performance by using SUM to add the contentsof fields. The SUM statement adds the contents of specified SUM fields in records with identical control fields. The result is placed in one record while the other record is deleted, thus reducing the number of records to be output by DFSORT.

You can use the ZDPRINT=YES installation option or the ZDPRINT run-time option to specify that positive zoned decimal fields that result from summing are to be printable. That is, you can tell DFSORT to change the last digit of the zone from hex C to hex F.

Eliminating Duplicate Records: You can eliminate records with duplicate keys by specifying

```
SUM FIELDS=NONE
```

when using the SUM control statement.

For a diagram of the processing sequence for record handling statements, user exits, and options, see Figure 2 on page 8.

ICETOOL

ICETOOL is a multi-purpose utility that allows you to use DFSORT's highly efficient I/O and processing to perform multiple operations on one or more data sets in a single job step. ICETOOL's twelve operators allow you to perform sort, copy, statistical, and report operations quickly and efficiently.

Avoid Options that Degrade Performance

Certain options can adversely affect performance, and shouldbe 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 moreefficient 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 optionincreases the time needed for comparison of records and for data transfer.

EQUCOUNT

The EQUCOUNT option takes additional time to count the number ofrecords with equal keys.

LOCALE

The LOCALE option may increase the time required to run anapplication.

NOCINV

The NOCINV option precludes the use of control interval accessfor more efficient VSAM processing.

NOBLKSET

The NOBLKSET option precludes the use of the more efficient Blocksettechnique.

VERIFY

The VERIFY option degrades performance, becauseit involves extra processing.

Tape Work Data Sets

Use of tape work data for intermediate storage precludes the useof the much more efficient disk techniques.

User Exit Routines

When user exit routines are included in an application, thetime required to run the application is usually increased.

The run time required by most user exit routines is generally small, but the routines at user exits E15, E32, and E35 are entered for each record of the data sets. For large input data sets, the total run 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 timerequired to run the application might increase.

Use Main Storage Efficiently

In general, the more main storage you make available to DFSORT, the better the performance for large applications. To prevent excessive paging, ensure that sufficient real storage is available to back up the amount of main storage used. This is especially important with main storage sizes greater than 32MB. The default amount of main storage that will be made available to DFSORT is defined when it is installed.

DFSORT requires a minimum of 88KB, but to get better performance, use a much larger amount of storage. The recommended amount is about 4MB. Improved performance will be most noticeable with large input files.

Note: When Blockset is selected, DFSORT can place selected buffers above 16MB virtual. This frees more storage for DFSORT without having to increase the REGION size. A REGION size of atleast 440KB must be available to allow DFSORT to use storage effectively.

Tuning Main Storage

Either the REGION value or the MAINSIZE/SIZE value candetermine how much storage is available to DFSORT. See *Installation and Customization* for details.

Generally, the most efficient way to allocate (virtual) main storage is to use MAINSIZE/SIZE=MAX explicitly or by default. However, problems can arise if the values for the TMAXLIM or MAXLIM installation options have been set excessively high (or low). Guidelines for setting these values are given in *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 a user 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.

When SIZE/MAINSIZE=MAX is in effect, DFSORT will use its Dynamic Storage Adjustment (DSA) feature, when appropriate, to improve performance. See *Installation and Customization* for details of the DSA installation parameter.

If the MINLIM value is greater than thatspecified 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 run, provided at least 88KB (below 16MB virtual) are available to DFSORT.

Although DFSORT requires a minimum of 88KB (below 16MB virtual), 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
- CHALT or SMF options

- ALTSEQ, INCLUDE, OMIT, SUM, OUTREC, or INREC control statements
- Very large blocks or logical records
- VSAM data sets
- An Extended Function Support (EFS) program
- An ICETEXIT routine
- A large ICEIEXIT routine
- OUTFIL control statements (especially if many OUTFIL data sets are specified or if the data sets have large block sizes)
- Locale processing.
- A large number of JCL or dynamically allocated work data sets.

Storage used for OUTFIL processing will be adjusted automatically, depending upon several factors, including:

- Total available storage
- Non-OUTFIL processing storage requirements
- Number of OUTFIL data sets and their attributes (for example, block size).

OUTFIL processing is subject to the ODMAXBF limit and your system storage limits (for example, IEFUSI) but not to DFSORT storage limits, that is, SIZE/MAINSIZE, MAXLIM, and TMAXLIM. DFSORT attempts to use storage above 16MB virtual for OUTFIL processing whenever possible.

Notes:

1. In some cases, this release of DFSORT may use more storage than prior releases for comparable applications. This might affect the operation of some applications. For example, some applications that run as in-storage sorts (with no SORTWKdd 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 440KB 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 16MB virtual.
2. For extremely large sorts (for example, 500MB or more of data), make sure that Hipersorting and dataspace sorting are enabled, or make sure that 16MB or more of main storage is available to DFSORT.

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, increasing storage (using REGION or SIZE/MAINSIZE=n) can give you some relief from the time constraint. If you are concerned with processor time, 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 to allow other programs to have the storage they need to operate simultaneously; by increasing the allocation, you can run large DFSORT applications efficiently at the risk of decreasing the efficiency of other applications sharing the multiprogramming environment.

Releasing Main Storage

Under some circumstances, DFSORT uses all the availablestorage in your REGION. This normally will not occur for storage above 16MB virtual (if it does, use the ARESINV 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 effectand n is greater than the REGION parameter or the default REGION value, or when SIZE/MAINSIZE=MAX and TMAXLIM is greater than your REGION, specify the storage you need released in the following way:

- For applications with user exits:
  - For directly 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 IEFUSI value,this difference is available. (OVERRGN is an installation option that can be modified only by your system programmer.)
  - For program invoked DFSORT, you can choose one of the following:
    - If the user 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 user exit address is passed in the parameter list,and SIZE/MAINSIZE=MAX is in effect, use the RESINV option.
    - If the user 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 user exit address inthe parameter list and SIZE/MAINSIZE=n is in effect, then consider having the OVERRGN value changed by your system programmer to less than your IEFUSI value.
- For applications without user exits:
  – For directly 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 IEFUSI value.
  – For program 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 IEFUSI 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.

Allocate 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.

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=3390), DFSORT assigns areas, and some optimization occurs automatically. You can get the best performance using direct access intermediate storage devices when you:

- Use two or more work data sets.
- 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 work data sets the same size, or as nearly the same size as possible.
- Make sure the SORTWKdd data sets do not share devices or channels with the SORTIN, SORTOUT, or OUTFIL data sets. When appropriate, use SmartBatch pipes for input and output data sets, to avoid contention.
- Specify contiguous space for work data sets, and make sure there is enough primary space so that the automatic secondary allocation is not needed.
- Provide adequate virtual storage when work data sets are allocated on non-synchronous devices, as explained in “Non-Synchronous Storage Subsystems” on page 541.

Elapsed time is decreased when DFSORT can both read input while writing to SORTWKdd and write output while reading from SORTWKdd. If, for example, you

have two channels, the best allocation of them is to have SORTIN, SORTOUT, and OUTFIL data sets on one and the SORTWKdd data sets on the other.

Storage requirements for different disk techniques can be estimated by using the guidelines found in “Appendix A. Using Work Space” on page 539.

Virtual I/O for Work Data Sets

Although VIO data sets can provide performance improvements in many applications, these work data sets are generally not as effective as DASD work data sets for DFSORT.

DFSORT temporary work data sets allocated to virtual devices (VIO) can provide reduced elapsed time at the cost of increased CPU time for DFSORT applications. In general, this is not a good trade-off and VIO should not be used for DFSORT work data sets unless:

- The system supports VIO in expanded storage, and
- Elapsed time is of primary concern.

If work data sets are allocated to VIO, the ICEMAC option VIO tells DFSORT how to handle to VIOs:

- VIO=YES causes DFSORT to accept the use of VIOs for work data sets.
- VIO=NO causes DFSORT to reallocate work data sets from virtual devices to real devices. Note that in order for re-allocation to be successful, a real device with the same device type as the virtual device must be available.

Re-allocation of VIO data sets (VIO=NO) is recommended over no re-allocation (VIO=YES). However, it is better to avoid using VIO work data sets in the first place, since re-allocation wastes time and resources.

Tape Work Storage Devices

The use of tape work storage devices prevents the use of themore 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 88KB 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 100KB available.

For information on how to determine storage requirements when using different tape techniques, see “Appendix A. Using Work Space” on page 539.

Notes:

1. The frequency with which the tape direction changes during DFSORT work file operations has more of an impact on the effective data rate of the IBM 3480/3490/3590 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.

2. Devices using the Improved Data Recording Capability (IDRC) feature are not recommended as intermediate storage devices, as they do not perform well with the read backward command.

Use Hipersorting

Hipersorting uses Hiperspace. 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. Hipersorting is recommended when the input or output is a compressed sequential or VSAM data set.

You can control the maximum amount of Hiperspace for a Hipersortingapplication with the HIPRMAX parameter. HIPRMAX can direct DFSORT to dynamically determine the maximum amount of Hiperspace, subject to the available expanded storage at the start of the run. You can also use HIPRMAX to suppress Hipersorting when optimizing CPU time is your major concern because Hipersorting can slightly degrade CPU time.

The actual amount of Hiperspace a Hipersorting application uses depends upon several factors. See the HIPRMAX description in “OPTION Control Statement” on page 134 for more details. Most important, throughout the run, DFSORT determines the amount of available expanded storage as well as the amount of expanded storage needed by other concurrent Hipersorting applications. Based on this information, DFSORT switches dynamically from using Hiperspace to using DASD work data sets when either an expanded storage shortage is predicted or the total Hipersorting activity on the system reaches the limits set by the DFSORT installation options EXPMAX, EXPOLD, and EXPRES. See *Installation and Customization* for a complete description of these installation options.

Sort with Data Space

Dataspace sorting uses data space to improve the performance of sort applications that use DFSORT’s Blockset Technique.

Dataspace sorting allows DFSORT to sort large pieces of data at a time. This helps to reduce CPU time and elapsed time.

The maximum amount of data space used for dataspace sorting can be controlled with the DSPSIZE option. DSPSIZE=MAX allows DFSORT to select the maximum data space to use. In this case, the amount used would depend on the size of the file being sorted and the paging activity of your system. DSPSIZE=0 means that DFSORT will not use dataspace sorting.

The following functions and types of data sets are not supported for dataspace sorting:

- Spool, dummy, or pipe data set, or HFS file, as input.
- User exits
- INREC, OUTFIL, OUTREC, and SUM
- EQUCOUNT

Sort with Data Space

Dataspace sorting is seldom used for very small data sets of a few MB or so because it is more efficient to sort small amounts of data entirely in main storage.

In order for dataspace sorting to be used, you need sufficient available central storage, that is, unused or not recently used, as reported by SRM at the start of the sort. Such storage is needed to back the corresponding data space required by DFSORT. The amount of data space required varies. Typically, it grows as the amount of data to sort increases, and, it shrinks as the amount of main storage specified increases.

The following are actions you can take which might increase the use of dataspace sorting:

- Specify sufficient main storage. The default is 4MB, the recommended minimum for dataspace sorting. If you increase the amount of main storage specified, more dataspace sorting is possible, especially when sorting large amounts of data (multiple hundred MBs). Specifying more than 12MB or so will have no significant impact on DFSORT's decision to use dataspace sorting; it will, however, improve the performance of large non-dataspace sort applications.
- Specify generous extent sizes for work data sets, especially for secondary extents. Dataspace sorting is frequently used in conjunction with DASD work space but never with Hiperspace or with tape work data sets.
- Specify DSPSIZE=MAX.
- Verify that IEFUSI does not place any restrictions on the size of the data spaces that a single address space can create.
- Ensure that DFSORT has accurate information about the input file size. DFSORT can automatically estimate the file size for DASD input data sets and DFSMSrmm-managed tape data sets. But in situations where DFSORT cannot determine the file size accurately as described in "File Size and Dynamic Allocation" on page 543, specify a FILSZ=Un estimate on the OPTION control statement, where n is the number of records to be sorted.

Use ICEGENER Instead of IEBGENER

You can achieve more efficient processing for applications set up touse the IEBGENER system utility by using DFSORT's ICEGENER facility. Qualifying IEBGENER jobs are processed by the equivalent (though not identical), 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.

ICEGENER, like IEBGENER, will use an SDB=value parameter you supply, when appropriate. If you do not supply an SDB=value parameter, ICEGENER will use your site's DFSORT installation default for SDB, when appropriate (the IBM-supplied default is SDB=INPUT). If ICEGENER transfers control to IEBGENER, IEBGENER will use the SDB=value parameter you supply, if any, or its normal default for SDB.

ICEGENER can transfer control to IEBGENER due to DFSPARM or SORTCNTL statement errors or other errors detected by DFSORT. Therefore, we recommend that ICEGENER not be used for any application for which IEBGENER cannot be used, to avoid unwanted IEBGENER processing. For example, if ICEGENER is used with an INCLUDE statement in DFSPARM, IEBGENER could be used and complete successfully, but the INCLUDE statement would be ignored. Instead, DFSORT copy should be used directly so that IEBGENER cannot be called.

Use ICEGENER Instead of IEBGENER

If your site has installed ICEGENER to be invoked by the name IEBGENER, you need not make any changes to your applications 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 directly invoked) or LINK macro (when DFSORT is program-invoked) in any applications you choose. Program-invoked applications must be recompiled.

Following is an example of how an IEBGENER application 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:

1. The SYSUT2 data set should not be the same as the SYSUT1 data set because this can cause lost or incorrect data or unpredictable results.
2. Whether ICEGENER is invoked from a program or not, DFSORT will be invoked from ICEGENER using an extended parameter list. Therefore, the installation options for the program-invoked environment (that is, ICEAM2 or ICEAM4 or an ICETDx module activated for the ICEAM2 or ICEAM4 environment) apply and SORTCNTL or DFSPARM can be used to provide additional control statements for the copy application; for example, OPTION. However, ICEGENER can transfer control to IEBGENER due to DFSPARM or SORTCNTL statement errors or other errors detected by DFSORT. Therefore, DFSORT copy should be used directly rather than ICEGENER if DFSORT processing statements such as INCLUDE, OUTREC, SUM and so on are required.
3. For most error conditions that prevent the use of DFSORT copy, control will be transferred to the IEBGENER system utility. 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.
4. 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.
5. DFSORT copy can perform some functions not provided by IEBGENER, such as certain padding and truncation operations. ICEGENER processing is not identical to IEBGENER processing in all cases, since DFSORT copy uses methods to enhance performance (EXCP, for example) that are not used by IEBGENER.
6. In some cases, IEBGENER terminates when the SYSUT2 LRECL is different from the SYSUT1 LRECL. ICEGENER takes one of three actions depending on ICEMAC option GNPAD (LRECL padding) or GNTRUNC (LRECL truncation), as appropriate.

Use ICEGENER Instead of IEBGENER

If you want ICEGENER to transfer control to IEBGENER when the SYSUT2 LRECL is larger than the SYSUT1 LRECL, use ICEMAC option GNPAD=IEB. If you want ICEGENER to handle LRECL padding, use GNPAD=RC0 (the supplied default) or GNPAD=RC4.

If you want ICEGENER to transfer control to IEBGENER when the SYSUT2 LRECL is smaller than the SYSUT1 LRECL, use ICEMAC option GNTRUNC=IEB. If you want ICEGENER to handle LRECL truncation, use GNTRUNC=RC0 (the supplied default) or GNTRUNC=RC4.

ICEGENER Return Codes

ICEGENER can use either IEBGENER or the DFSORT copy function. However, for unsuccessful completion due to an unsupported operating system, ICEGENER passes back a return code of 24 to the operating system or the invoking program, without using either IEBGENER or DFSORT copy.

If ICEGENER transfers control to IEBGENER, IEBGENER passes back its return code to the operating system or the invoking program.

If ICEGENER uses the DFSORT copy function:

- For successful completion, ICEGENER passes back a return code of 0 or 4 to the operating system or the invoking program.
- For unsuccessful completion with NOABEND in effect, ICEGENER passes back a return code of 12 (changed from 16) to the operating system or the invoking program.
- For unsuccessful completion with ABEND in effect, DFSORT issues a user abend with the appropriate code as specified by the ICEMAC option ABCODE (either the error message number or a number between 1 and 99).

The meanings of the return codes that ICEGENER passes back (in register 15) are:

0 **Successful completion**. ICEGENER completed successfully.

4 **Successful completion**. ICEGENER completed successfully, and:

- ICEMAC option GNPAD=RC4 was specified and the SYSUT2 LRECL was larger than the SYSUT1 LRECL (LRECL padding) or
- ICEMAC option GNTRUNC=RC4 was specified and the SYSUT2 LRECL was smaller than the SYSUT1 LRECL (LRECL truncation), or
- SPANINC=RC4 was in effect and one or more incomplete spanned records was detected.

12 **Unsuccessful completion**. DFSORT detected an error that prevented ICEGENER from completing successfully.

24 **Unsupported operating system**. This operating system is not supported by this release of ICEGENER. Only current or subsequent releases of the following systems are supported:

- OS/390
- MVS/ESA

Use DFSORT's Performance Booster for The SAS System

DFSORT provides significant CPU time improvements for SAS applications. To take advantage of this feature, contact SAS Institute Inc. for details of the support they provide to enable this enhancement.

Use DFSORT's BLDINDEX Support

DFSORT provides support that enables IDCAMS BLDINDEX to automatically use DFSORT to improve the performance of most BLDINDEX jobs that require BLDINDEX external sorting.

Chapter 10. Examples of DFSORT Job Streams

Summary of Examples 509
Storage Administrator Examples 510
REXX Examples 510
Sort Examples. 511
Example 1. Sort with ALTSEQ 511
Example 2. Sort with OMIT, SUM, OUTREC, DYNALLOC and ZDPRINT 512
Example 3. Sort with ISCII/ASCII Tapes 513
Example 4. Sort with E15, E35, FILSZ, AVGRLEN and DYNALLOC 514
Example 5. Called sort with SORTCNTL, CHALT, DYNALLOC and FILSZ 516
Example 6. Sort with VSAM Input/Output, DFSPARM and Option Override 517
Example 7. Sort with COBOL E15, EXEC PARM, COBEXIT and MSGDDN 518
Example 8. Sort with Dynamic Link-Editing of Exits 520
Example 9. Sort with the Extended Parameter List Interface 521
Example 10. Sort with OUTFIL 524
Example 11. Sort with SmartBatch Pipes and OUTFIL SPLIT 526
Example 12. Sort with INCLUDE and LOCALE. 528
Example 13: Sort with HFS Files 529
Merge Examples 530
Example 1. Merge with EQUALS 530
Example 2. Merge with LOCALE and OUTFIL 531
Copy Examples 532
Example 1. Copy with EXEC PARMs, SKIPREC, MSGPRT and ABEND 532
Example 2. Copy with INCLUDE and VLSHRT. 533
ICEGENER Example 534
ICETOOL Example 535

Summary of Examples

The table below summarizes the examples provided in this chapter.

Application	No.	Input	Output	Functions/Options
Sort	1	DASD	Tape	ALTSEQ
Sort	2	DASD	DASD	OMIT, SUM, OUTREC, DYNALLOC, ZDPRINT
Sort	3	Tape	Tape	ISCII/ASCII Tapes
Sort	4	Tape	DASD	E15, E35, FILSZ, AVGRLEN, DYNALLOC
Sort	5	DASD	DASD	Program-invoked, SORTCNTL, CHALT, DYNALLOC, FILSZ
Sort	6	DASD	DASD	VSAM Input/Output, DFSPARM, Option Override
Sort	7	DASD	DASD	COBOL E15, EXEC PARM, COBEXIT, MSGDDN
Sort	8	DASD	DASD	Dynamic Link-editing of Exits
Sort	9	E15	DASD	Extended Parameter List Interface
Sort	10	DASD	DASD and SYSOUT	OUTFIL
Sort	11	Pipe	Pipes	SmartBatch Pipes, OUTFIL SPLIT, FILSZ, DYNALLOC
Sort	12	DASD	DASD	INCLUDE, LOCALE
Sort	13	HFS files	HFS file	
Merge	1	DASD	DASD	EQUALS

© Copyright IBM Corp. 1973, 2000

Summary of Examples

Application	No.	Input	Output	Functions/Options
Merge	2	DASD	DASD	LOCALE, OUTFIL
Copy	1	Tape	DASD	EXEC PARMs, SKIPREC, MSGPRT, ABEND
Copy	2	DASD	DASD	INCLUDE, VLSHRT
ICEGENER	1	DASD	DASD	
ICETOOL	1	DASD	DASD	OCCUR, COPY, SORT, MODE, VERIFY, STATS, DISPLAY

Storage Administrator Examples

DFSORT provides a set of sample jobs that demonstrate techniques of interest to Storage Administrators and others who analyze data collected from DFSMShsm, DFSMSrmm, DCOLLECT and SMF. These sample jobs can be found in the ICESTGEX member of the SICESAMP library (contact your System Programmer for details). You can also download these sample jobs from the DFSORT FTP site. These sample jobs show some of the many ways DFSORT features such as ICETOOL and OUTFIL can be used to analyze data and generate reports:

DCOLEX1
DCOLLECT Example 1: VSAM report

DCOLEX2
DCOLLECT Example 2: Conversion reports

DCOLEX3
DCOLLECT Example 3: Capacity planning analysis and reports

DFHSMEX1
DFHSM Example 1: Deciphering Activity Logs

DFHSMEX2
DFHSM Example 2: Recover a DFHSM CDS with a broken index

RMMEX1
DFSMSrmm Example 1: SMF audit report

RMMEX2
DFSMSrmm Example 2: Create ADDVOLUME commands

REXX Examples

Both DFSORT and ICETOOL can be called from REXX. The key is to specify ALLOCATE statements for the data sets you need and then use an ADDRESS statement like this:

```
ADDRESS LINKMVS name
```

which says to fetch the named program using the standard system search list.

Here is an example of a REXX CLIST to call DFSORT:

```
/* Simple REXX CLIST to call DFSORT */
  "FREE FI(SYSOUT SORTIN SORTOUT SYSIN)"
  "ALLOC FI(SYSOUT)   DA(*)"
  "ALLOC FI(SORTIN)   DA('Y897797.INS1') REUSE"
  "ALLOC FI(SORTOUT)  DA('Y897797.OUTS1') REUSE"
  "ALLOC FI(SYSIN)    DA('Y897797.SORT.STMTS') SHR REUSE"
  ADDRESS LINKMVS ICEMAN
```

Here are the DFSORT control statements that might appear in the Y897797.SORT.STMTS data set:

```
SORT FIELDS=(5,4,CH,A)
INCLUDE COND=(21,3,SS,EQ,C'L92,J82,M72')
```

Here is an example of a REXX CLIST to call ICETOOL:

```
/* Simple REXX CLIST to call ICETOOL */
  "FREE FI(TOOLMSG DFSMSG VLR LENDIST TOOLIN)"
  "ALLOC FI(TOOLMSG)   DA(*)"
  "ALLOC FI(DFSMSG)   DUMMY"
  "ALLOC FI(VLR)   DA('Y897797.VARIN') REUSE"
  "ALLOC FI(LENDIST)   DA(*)"
  "ALLOC FI(TOOLIN)    DA('Y897797.TOOLIN.STMTS') SHR REUSE"
  ADDRESS LINKMVS ICETOOL
```

Here are the ICETOOL statements that might appear in the Y897797.TOOLIN.STMTS data set:

```
 OCCURS FROM(VLR) LIST(LENDIST) -
   TITLE('LENGTH DISTRIBUTION REPORT') BLANK -
   HEADER('LENGTH')  HEADER('NUMBER OF RECORDS') -
   ON(VLEN)          ON(VALCNT)
```

Sort Examples

This section includes 12 sort examples.

Example 1. Sort with ALTSEQ

INPUT Blocked variable-length records on DASD

OUTPUT
Blocked variable-length records on 3490

WORK DATA SETS
Two 3390 data sets

USER EXITS
None

FUNCTIONS/OPTIONS
ALTSEQ

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//S1       EXEC PGM=SORT                                        02
//SYSOUT   DD SYSOUT=A                                          03
//SORTIN   DD DSN=A123456.IN5,DISP=SHR                          04
//SORTOUT  DD DSN=OUT1,UNIT=3490,DISP=(,KEEP),VOL=SER=VOL001    05
//SORTWK01 DD UNIT=3390,SPACE=(CYL,(10,10))                     06
//SORTWK02 DD UNIT=3390,SPACE=(CYL,(10,10))                     07
//SYSIN    DD *                                                 08
* COLLATE $, # and @ AFTER Z                                   09
  SORT FIELDS=(7,5,AQ,A)                                        10
  ALTSEQ CODE=(5BEA,7BEB,7CEC)                                  11
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias SORT.

03 SYSOUT DD statement. Directs DFSORT messages and control statements to system output class A.

04 SORTIN DD statement. The input data set is named A123456.IN5 and is cataloged. DFSORT determines from the data set label that the RECFM is VB, the maximum LRECL is 120, and the BLKSIZE is 2200.

05 SORTOUT DD statement. The output data set is named OUT1 and is to be allocated on 3490 volume VOL001 and kept. DFSORT sets the RECFM and LRECL from SORTIN and selects an appropriate BLKSIZE for this standard labeled tape.

06 SORTWK01 DD statement. The first work data set is allocated on 3390.

07 SORTWK02 DD statement. The second work data set is allocated on 3390.

08 SYSIN DD statement. DFSORT control statements follow.

09 Comment statement. Printed but otherwise ignored.

10 SORT statement. FIELDS specifies an ascending 5-byte character control field starting at position 7 (the third data byte, since the RDW occupies the first 4 bytes). The control field is to be collated according to the modified sequence described in the ALTSEQ statement.

11 ALTSEQ statement. CODE specifies that the three characters $, # and @ are to collate in that order after Z.

Example 2. Sort with OMIT, SUM, OUTREC, DYNALLOC and ZDPRINT

INPUT Blocked fixed-length records on 3380 and 3390

OUTPUT
Blocked fixed-length records on 3390

WORK DATA SETS
Dynamically allocated

USER EXITS
None

FUNCTIONS/OPTIONS
OMIT, OUTREC, SUM, DYNALLOC, ZDPRINT

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//STEP1    EXEC PGM=SORT                                        02
//SYSOUT   DD SYSOUT=H                                          03
//SORTIN   DD DSN=INP1,DISP=SHR,UNIT=3380,VOL=SER=SCR001        04
//         DD DSN=INP2,DISP=SHR,UNIT=3390,VOL=SER=SYS351        05
//SORTOUT  DD DSN=&&OUTPUT,DISP=(,PASS),UNIT=3390,              06
//   SPACE=(CYL,(5,1)),DCB=(LRECL=22)                           07
//SYSIN    DD *                                                 08
  OMIT COND=(5,1,CH,EQ,C'M')                                    09
  SORT FIELDS=(20,8,CH,A,10,3,FI,D)                             10
  SUM FIELDS=(16,4,ZD)                                          11
  OPTION DYNALLOC,ZDPRINT                                       12
  OUTREC FIELDS=(10,3,20,8,16,4,2Z,5,1,C' SUM')                 13
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias SORT.

03 SYSOUT DD statement. Directs DFSORT messages and control statements to system output class H.

04-05 SORTIN DD statement. Consists of a concatenation of two data sets. The first input data set is named INP1 and resides on 3380 volume SCR001. The second input data set is named INP2 and resides on 3390 volume SYS351. DFSORT determines from the data set labels that the record format is FB, the LRECL is 80 and the largest BLKSIZE is 27920.

06-07 SORTOUT DD statement. The output data set is temporary and is to be allocated on a 3390. Since the OUTREC statement results in a reformatted output record length of 22 bytes, LRECL=22 must be specified. DFSORT sets the RECFM from SORTIN and selects an appropriate BLKSIZE.

08 SYSIN DD statement. DFSORT control statements follow.

09 OMIT statement. COND specifies that input records with a character M in position 5 are to be omitted from the output data set.

10 SORT statement. FIELDS specifies an ascending 8-byte character control field starting at position 20 and a descending 3-byte fixed-point control field starting at position 10.

11 SUM statement. FIELDS specifies a 4-byte zoned-decimal summary field starting at position 16. Whenever two records with the same control fields (specified in the SORT statement) are found, their summary fields (specified in the SUM statement) are to be added and placed in one of the records, and the other record is to be deleted.

12 OPTION statement. DYNALLOC specifies that work data sets are to be dynamically allocated using the installation defaults for the type of device and number of devices. ZDPRINT specifies that positive ZD SUM fields are to be printable.

13 OUTREC statement. FIELDS specifies how the records are to be reformatted for output. The reformatted records are 22 bytes long and look as follows:

Position	Content
1-3	Input positions 10 through 12
4-11	Input positions 20 through 27
12-15	Input positions 16 through 19
16-17	Zeros
18	Input position 5
19-22	The character string ’ SUM’

Example 3. Sort with ISCII/ASCII Tapes

INPUT Variable-length ISCII/ASCII records on 3590

OUTPUT
Variable-length ISCII/ASCII records on 3590

WORK DATA SETS
One SYSDA data set

USER EXITS
None

FUNCTIONS/OPTIONS
None

```
//EXAMP    JOB A400,PROGRAMMER                                    01
//RUNIT    EXEC SORTD                                             02
//SORTIN   DD DSN=SRTFIL,DISP=(OLD,DELETE),UNIT=3590,             03
//  DCB=(RECFM=D,LRECL=400,BLKSIZE=404,OPTCD=Q,BUFOFF=L),         04
//  VOL=SER=311500,LABEL=(1,AL)                                   05
//SORTOUT  DD DSN=OUTFIL,UNIT=3590,LABEL=(,AL),DISP=(,KEEP),      06
//  DCB=(BLKSIZE=404,OPTCD=Q,BUFOFF=L),VOL=SER=311501            07
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(4))                          08
//SYSIN    DD *                                                   09
  SORT FIELDS=(10,8,AC,D)                                         10
  RECORD TYPE=D,LENGTH=(,,,20,80)                                 11
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Uses the SORTD cataloged procedure to call DFSORT directly.
03-05	SORTIN DD statement. The input data set is named SRTFIL and resides on 3590 volume 311500. It is to be deleted after this job step. It has a RECFM of D (variable-length ISCII/ASCII records), a maximum LRECL of 400, a BLKSIZE of 404 and 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.
06-07	SORTOUT DD statement. The output data set is named OUTFIL and is to be allocated on 3590 volume 311501 and kept. It is to be written with an ISCII/ASCII label. DFSORT sets the RECFM and LRECL from SORTIN and sets the BLKSIZE to 404 as indicated in the DD statement. For this job, the buffer offset is the block length indicator. The records are to be translated from EBCDIC to ISCII/ASCII.
08	SORTWK01 DD statement. The work data set is allocated on SYSDA.
09	SYSIN DD statement. DFSORT control statements follow.
10	SORT statement. FIELDS specifies a descending 8-byte ISCII/ASCII control field starting at position 10.
11	RECORD statement. TYPE specifies ISCII/ASCII variable-length records. LENGTH specifies that the minimum record length is 20 and the average record length is 80.

Example 4. Sort with E15, E35, FILSZ, AVGRLEN and DYNALLOC

INPUT Variable-length records on 3490

OUTPUT
Blocked variable-length records on SYSDA

WORK DATA SETS
Dynamically allocated

USER EXITS
E15 and E35

FUNCTIONS/OPTIONS
FILSZ, AVGRLEN, DYNALLOC

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//STEP1    EXEC PGM=ICEMAN                                      02
//SYSOUT   DD SYSOUT=A                                          03
//SORTIN   DD DSN=INPUT,VOL=SER=FLY123,                         04
//  UNIT=3490,DISP=OLD                                          05
//SORTOUT  DD DSN=&&OUT,DISP=(,PASS),SPACE=(CYL,(10,12)),       06
//  UNIT=SYSDA,DCB=(RECFM=VB)                                   07
//MODLIB   DD DSN=EXIT1.RTNS,DISP=SHR                           08
//         DD DSN=EXIT2.RTNS,DISP=SHR                           09
//SYSIN    DD *                                                 10
  SORT FIELDS=(23,4,PD,A,10,6,FS,A)                             11
  OPTION DYNALLOC=(3390,3),AVGRLEN=75,FILSZ=E50000              12
  MODS E15=(MODREC,1024,MODLIB),E35=(ADDREC,1200,MODLIB)        13
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly.
03	SYSOUT DD statement. Directs DFSORT messages and control statements to system output class A.
04-05	SORTIN DD statement. The input data set is named INPUT and resides on 3490 volume FLY123. DFSORT determines from the data set label of this standard labeled tape that the RECFM is V, the LRECL is 120 and the BLKSIZE is 124.
06-07	SORTOUT DD statement. The output data set is temporary and is to be allocated on SYSDA. Since the input is unblocked and the output is to be blocked, RECFM=VB must be specified. DFSORT sets the LRECL from SORTIN and selects an appropriate BLKSIZE.
08-09	MODLIB DD statement. Specifies the load libraries that contain the exit routines. When exit routines reside in more than one library, the libraries must be concatenated using a single DD statement.
10	SYSIN DD statement. DFSORT control statements follow.
11	SORT statement. FIELDS specifies an ascending 4-byte packed-decimal control field starting at position 23 and an ascending 6-byte floating-sign control field starting at position 10.
12	OPTION statement. DYNALLOC=(3390,3) specifies that three 3390 work data sets are to be allocated. AVGRLEN=75 specifies an average record length of 75. AVGRLEN helps DFSORT optimize work space for variable-length record input. FILSZ=E50000 specifies an estimate of 50000 records. Since the 3490 input data set is compacted, DFSORT might not be able to determine the file size accurately unless the data set is

DFSMSrmm-managed; specifying FILSZ can make a significant difference in work space optimization when tape input data sets are not DFSMSrmm-managed.

13 MODS statement. E15 specifies a user exit routine named MODREC. Approximately 1024 bytes are required for MODREC and the system services (for example, GETMAIN and OPEN) it performs. E35 specifies a user exit routine named ADDREC. Approximately 1200 bytes are required for ADDREC and the system services it performs. MODREC and ADDREC reside in the libraries defined by the MODLIB DD statement.

Example 5. Called sort with SORTCNTL, CHALT, DYNALLOC and FILSZ

INPUT Blocked fixed-length records on DASD

OUTPUT
Blocked fixed-length records on DASD

WORK DATA SETS
Dynamically allocated

USER EXITS
None

FUNCTIONS/OPTIONS
CHALT, DYNALLOC, FILSZ

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//RUNSORT  EXEC PGM=MYPGM                                       02
//STEPLIB  DD DSN=M999999.LOAD,DISP=SHR                         03
//SYSOUT   DD SYSOUT=A                                          04
//SYSPRINT DD SYSOUT=A                                          05
//SORTIN   DD DSN=M999999.INPUT(MASTER),DISP=OLD                06
//SORTOUT  DD DSN=M999999.OUTPUT.FILE,DISP=OLD                  07
//SORTCNTL DD *                                                 08
   OPTION  CHALT,DYNALLOC=(,3),FILSZ=U25000                     09
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls a program named MYPGM that in turn calls DFSORT.
03	STEPLIB DD statement. Specifies the load library that contains MYPGM.
04	SYSOUT DD statement. Directs DFSORT messages and control statements to system output class A.
05	SYSPRINT DD statement. Directs MYPGM output to system output class A.
06	SORTIN DD statement. The input data set is member MASTER in the cataloged partitioned data set M999999.INPUT. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.
07	SORTOUT DD statement. The output data set is named M999999.OUTPUT.FILE and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.

08 SORTCNTL DD statement. DFSORT control statements follow. Statements in SORTCNTL override or supplement statements passed by MYPGM in the DFSORT parameter list it uses.

09 OPTION statement. CHALT specifies that character format control fields (specified in the SORT statement passed by MYPGM) are to be sorted using the installation default ALTSEQ table. DYNALLOC=(,3) specifies that three work data sets are to be dynamically allocated using the installation default for the type of device. FILSZ=U25000 specifies a file size of 25000 records is to be used by DFSORT to determine the amount of work space needed. Since the input data set is a member of a PDS, specifying FILSZ helps DFSORT optimize work data set space.

Example 6. Sort with VSAM Input/Output, DFSPARM and Option Override

INPUT VSAM TYPE=V records

OUTPUT
VSAM TYPE=V records

WORK DATA SETS
Dynamically allocated

USER EXITS
None

FUNCTIONS/OPTIONS
Override of Various Options

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//S1       EXEC PGM=SORT                                        02
//SYSOUT   DD SYSOUT=A                                          03
//SORTIN   DD DSN=TEST.SORTIN.FILE,DISP=SHR                     04
//SORTOUT  DD DSN=TEST.SORTOUT.FILE,DISP=SHR                    05
//DFSPARM  DD *                                                 06
  RECORD TYPE=V                                                 07
  SORT FIELDS=(30,4,BI,A)                                       08
  OPTION HIPRMAX=10,DYNALLOC=3390,MAINSIZE=3M,                  09
    MSGPRT=CRITICAL,NOLIST                                      10
```

For purposes of illustration, assume that none of the standard installation defaults for batch direct invocation of DFSORT have been changed by the site.

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias SORT.
03	SYSOUT DD statement. Directs DFSORT messages and control statements to system output class A.
04	SORTIN DD statement. The input data set is TEST.SORTIN.FILE. DFSORT determines that it is a VSAM data set and obtains its attributes from the catalog.
05	SORTOUT DD statement. The output data set is TEST.SORTOUT.FILE. DFSORT determines that it is a VSAM data set and obtains its attributes from the catalog.

06 DFSPARM DD statement. DFSORT control statements follow. DFSPARM can be used for both direct-invocation and program-invocation of DFSORT and overrides options and statements from all other sources. Certain operands, such as MSGPRT and LIST/NOLIST, are used if supplied in DFSPARM, the EXEC PARM or the invocation parameter list, but not used if supplied in SYSIN or SORTCNTL.

07 RECORD statement. TYPE=V specifies that DFSORT is to treat the VSAM records as variable-length. In this case, the RECORD statement could be omitted, since DFSORT would automatically set a record type of V due to the use of VSAM data sets for SORTIN and SORTOUT.

08 SORT statement. FIELDS specifies an ascending 4-byte binary control field starting at position 30. This position corresponds to a specification of KEYS(4 25) for the VSAM CLUSTER (4 bytes at offset 25, which is equivalent to position 26 with 4 bytes added for the RDW that DFSORT supplies at input and removes at output for VSAM TYPE=V records).

09-10 OPTION statement. HIPRMAX=10 specifies that up to 10 MBs of Hiperspace can be committed for Hipersorting, overriding the standard installation default of HIPRMAX=OPTIMAL. DYNALLOC=3390 specifies that work data sets are to be allocated on 3390s, overriding the standard installation default of SYSDA. The standard installation default of four work data sets is not overridden. MAINSIZE=3M specifies that up to 3 MBs of storage can be used, overriding the standard installation default of MAINSIZE=MAX. MSGPRT=CRITICAL specifies that only error messages are to be printed, overriding the standard installation default of MSGPRT=ALL. NOLIST specifies that control statements are not to be printed, overriding the standard installation default of LIST=YES.

Example 7. Sort with COBOL E15, EXEC PARM, COBEXIT and MSGDDN

INPUT Fixed-length records on DASD

OUTPUT
Fixed-length records on SYSDA

WORK DATA SETS
None

USER EXITS
COBOL E15

FUNCTIONS/OPTIONS
COBEXIT, MSGDDN

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//STEP1    EXEC PGM=SORT,PARM='MSGDDN=DFSOUT'                   02
//STEPLIB  DD DSN=SYS1.COBLIB,DISP=SHR                          03
//SYSOUT   DD SYSOUT=A                                          04
//DFSOUT   DD SYSOUT=A                                          05
//EXITC    DD DSN=COBEXITS.LOADLIB,DISP=SHR                     06
//SORTIN   DD DSN=SORT1.IN,DISP=SHR                             07
//SORTOUT  DD DSN=&&OUT,DISP=(,PASS),SPACE=(CYL,(5,5)),         08
//  UNIT=SYSDA,DCB=(LRECL=120)                                  09
//SYSIN    DD *                                                 10
  SORT FIELDS=(5,4,A,22,2,A),FORMAT=BI                          11
  MODS E15=(COBOLE15,37000,EXITC,C)                             12
  RECORD LENGTH=(,120)                                          13
  OPTION COBEXIT=COB2                                           14
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias name SORT. MSGDDN=DFSOUT specifies an alternate message data set for DFSORT messages and control statements to prevent the COBOL messages in SYSOUT from being interleaved with the DFSORT messages and control statements.
03	STEPLIB statement. Specifies the VS COBOL II library or the Language Environment library, as appropriate.
04	SYSOUT statement. Directs COBOL messages to system output class A.
05	DFSOUT statement. Directs DFSORT messages and control statements to system output class A (this is the alternate message data set specified by MSGDDN in the PARM field of the EXEC statement).
06	EXITC statement. Specifies the load library that contains the exit routine.
07	SORTIN DD statement. The input data set is named SORT1.IN and is cataloged. DFSORT determines from the data set label that the RECFM is F, the LRECL is 100 and the BLKSIZE is 100.
08-09	SORTOUT DD statement. The output data set is temporary and is to be allocated on SYSDA. Since the E15 exit changes the length of the records from 100 bytes to 120 bytes, LRECL=120 must be specified. DFSORT sets the RECFM from SORTIN and sets the BLKSIZE to the LRECL (unblocked records).
10	SYSIN DD statement. DFSORT control statements follow.
11	SORT statement. FIELDS specifies an ascending 4-byte control field

starting at position 5 and an ascending 2-byte control field starting at position 22. FORMAT specifies that the control fields are binary.

12 MODS statement. E15 specifies a user exit routine named COBOLE15 written in COBOL. Approximately 37000 bytes are required for the exit, the system services (for example, GETMAIN and OPEN) it performs, and the COBOL library subroutines. COBOLE15 resides in the library defined by the EXITC DD statement.

13 RECORD statement. LENGTH specifies that the COBOL E15 routine changes the length of the records to 120 bytes.

14 OPTION statement. COBEXIT=COB2 specifies that the COBOL E15 routine is to be run with the VS COBOL II library or with the Language Environment library, as appropriate.

Example 8. Sort with Dynamic Link-Editing of Exits

INPUT Blocked fixed-length records on DASD

OUTPUT
Blocked fixed-length records on 3380

WORK DATA SETS
One SYSDA data set

USER EXITS
E11, E15, E17, E18, E19, E31, E35, E38, E39

FUNCTIONS/OPTIONS
None

```
 //EXAMP    JOB A400,PROGRAMMER                                  01
 //STEPA    EXEC SORT                                            02
 //SORTIN   DD DSN=SMITH.INPUT,DISP=SHR                          03
 //SORTOUT  DD DSN=SMITH.OUTPUT,DISP=(NEW,CATLG),                04
 //  UNIT=3380,SPACE=(TRK,(10,2)),VOL=SER=XYZ003                 05
 //SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1,1))                      06
 //EXIT     DD DSN=SMITH.EXIT.OBJ,DISP=SHR                       07
 //EXIT2    DD DSN=SMITH.EXIT2.OBJ,DISP=SHR                      08
 //SORTMODS DD UNIT=SYSDA,SPACE=(TRK,(10,,3))                    09
 //SYSIN    DD *                                                 10
   SORT FIELDS=(1,8,CH,A,20,4,BI,D)                             11
   MODS E11=(EXIT11,1024,EXIT,S),                               12
        E15=(E15,1024,SYSIN,T),                                 13
        E17=(EXIT17,1024,EXIT2,T),                              14
        E18=(EXIT18,1024,EXIT,T),                               15
        E19=(E19,1024,SYSIN,T),                                 16
        E31=(PH3EXIT,1024,EXIT,T),                              17
        E35=(PH3EXIT,1024,EXIT,T),                              18
        E38=(PH3EXIT,1024,EXIT,T),                              19
        E39=(E39,1024,SYSIN,T)                                  20
   END                                                          21
<object deck for E15 exit here>                                 22
<object deck for E19 exit here>                                 23
<object deck for E39 exit here>                                 24
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.

02 EXEC statement. Uses the SORT cataloged procedure to call DFSORT directly and supply the DD statements (not shown) required by the linkage editor.

03 SORTIN DD statement. The input data set is named SMITH.INPUT and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.

04-05 SORTOUT DD statement. The output data set is named SMITH.OUTPUT and is to be allocated on 3380 volume XYZ003 and cataloged. DFSORT sets the RECFM and LRECL from SORTIN and selects an appropriate BLKSIZE.

06 SORTWK01 DD statement. The work data set is allocated on SYSDA.

07 EXIT DD statement. Specifies the partitioned data set containing the object decks for the E11, E18, E31, E35 and E38 exit routines.

08 EXIT2 DD statement. Specifies the partitioned data set containing the object deck for the E17 exit routine.

09 SORTMODS DD statement. The partitioned data set to hold exit routine object decks from SYSIN for input to the linkage editor is to be allocated on SYSDA.

10 SYSIN DD statement. DFSORT control statements, and object decks to be used by the linkage editor, follow.

11 SORT statement. FIELDS specifies an ascending 8-byte character control field starting at position 1 and a descending 4-byte binary control field starting at position 20.

12-20 MODS statement. Specifies the exit routines to be used, the approximate number of bytes required for each exit and that:

- The EXIT11 routine in the EXIT library is to be link-edited separately from other input phase exit routines and associated with user 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 user 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 user exits E31, E35, E38, and E39, respectively.

21 END statement. Marks the end of the DFSORT control statements and the beginning of the exit routine object decks.

22-24 Object decks. The three object decks for the E15, E19, and E39 exit routines follow the END statement.

Example 9. Sort with the Extended Parameter List Interface

INPUT Fixed-length records from E15

OUTPUT
Blocked fixed-length records on SYSDA

WORK DATA SETS
Dynamically allocated

USER EXITS
E15

FUNCTIONS/OPTIONS
OMIT, FILSZ, RESINV, DYNALLOC

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//STEP1    EXEC PGM=MYSORT                                      02
//SYSOUT   DD SYSOUT=C                                          03
//MSGOUT   DD SYSOUT=C                                          04
//STEPLIB  DD DSN=A123456.LOAD,DISP=SHR                         05
//SORTOUT  DD DSN=&&OUTPUT,DISP=(,PASS),UNIT=SYSDA,             06
//  SPACE=(CYL,(8,4))                                           07
//SORTCNTL DD *                                                 08
* Update file size estimate                                    09
  OPTION FILSZ=E30000                                           10
-----------------------------------------------------------------------

-----------------------------------------------------------------------
MYSORT CSECT                                                    11
       .
       .
       .
       LA   R1,PL1            SET ADDRESS OF PARAMETER LIST     12
*                              TO BE PASSED TO DFSORT           13
       ST   R2,PL4            SET ADDRESS OF GETMAINED AREA     14
*                              TO BE PASSED TO E15              15
       LINK EP=SORT           INVOKE DFSORT                     16
       .
       .
       .
PL1    DC   A(CTLST)          ADDRESS OF CONTROL STATEMENTS     17
PL2    DC   A(E15)            ADDRESS OF E15 ROUTINE            18
PL3    DC   A(0)              NO E35 ROUTINE                    19
PL4    DS   A                 USER EXIT ADDRESS CONSTANT        20
PL5    DC   F'-1'             INDICATE END OF LIST              21
CTLST  DS   0H                CONTROL STATEMENTS AREA           22
       DC   AL2(CTL2-CTL1)    LENGTH OF CHARACTER STRING        23
CTL1   DC   C' SORT FIELDS=(5,8,CSF,A)'                         24
       DC   C' RECORD TYPE=F,LENGTH=80 '                        25
       DC   C' OPTION FILSZ=E25000,DYNALLOC,'                   26
       DC   C'RESINV=8000 '                                     27
       DC   C' OMIT FORMAT=CSF,COND=(5,8,EQ,13,8) '             28
CTL2   EQU  *                                                   29
OUT    DCB  DDNAME=MSGOUT,...                                   30
E15    DS   0H                E15 ROUTINE                       31
       .
       .
       .
       L    R2,4(,R1)         GET ADDRESS OF GETMAINED AREA     32
       .
       .
       .
       BR   R14               RETURN TO DFSORT                  33
       .
       .
       .
```

The JCL for running program MYSORT, and highlights of the code used by MYSORT to invoke DFSORT with the extended parameter list, are shown below. For purposes of illustration, assume that none of the standard installation defaults for batch program invocation of DFSORT have been changed by the site.

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls a program named MYSORT that in turn calls DFSORT.
03	SYSOUT DD statement. Directs DFSORT messages and control statements to SYSOUT class C.
04	MSGOUT DD statement. Directs MYSORT messages to SYSOUT class C.
05	STEPLIB DD statement. Specifies the load library that contains MYSORT.
06-07	SORTOUT DD statement. The output data set is temporary and is to be allocated on SYSDA. Since SORTIN is not used, DFSORT sets the RECFM and LRECL from the RECORD statement and sets the BLKSIZE to the LRECL (unblocked records).
08	SORTCNTL DD statement. DFSORT control statements follow. Statements in SORTCNTL override or supplement statements passed by MYSORT in the extended parameter list it uses.
09	Comment statement. Printed but otherwise ignored.
10	OPTION statement. FILSZ=E30000 specifies an estimate of 30000 records, overriding FILSZ=E25000 in the OPTION statement of the extended parameter list. Since the E15 routine supplies all of the input records, DFSORT will not be able to determine the file size accurately; therefore, specifying FILSZ can make a significant difference in work space optimization when an E15 routine supplies all of the input records. It's important to change the FILSZ value whenever the number of input records increases significantly.
11	This is the start of program MYSORT. Assume that it GETMAINs a work area, saves its address in register 2, and initializes it with information to be used by the E15 routine.
12-13	MYSORT places the address of the extended parameter list to be passed to DFSORT in register 1.
14-15	MYSORT places the address of the GETMAINed work area in the user exit address constant field in the extended parameter list. DFSORT will pass this address to the E15 routine (in the second word of the E15 parameter list) when it is entered.
16	MYSORT calls DFSORT by it's alias SORT.
17-21	The extended parameter list specifies: the address of the control statements area, the address of the E15 routine, that no E35 routine is present, and the address of the GETMAINed work area. F'-1' indicates the end of the extended parameter list. Subsequent parameter list fields, such as the address of an ALTSEQ table, are not used in this application. Since the address of the E15 routine is passed in the parameter list, SORTIN cannot be used; if a SORTIN DD statement were present, it would be ignored.

22-23 This is the start of the control statements area. The total length of the control statements is specified.

24 SORT statement. FIELDS specifies an ascending 8-byte floating-sign control field starting at position 5.

25 RECORD statement. TYPE=F and LENGTH=80 specify that the E15 inserts fixed-length records of 80 bytes. In this case, TYPE=F could be omitted, since DFSORT would automatically set a record type of F. However, LENGTH must be specified when an E15 supplies all of the input records.

26-27 OPTION statement. FILSZ=E25000 specifies an estimate of 25000 records, which is overridden by FILSZ=E30000 in SORTCNTL's OPTION statement. DYNALLOC specifies that work data sets are to be dynamically allocated using the installation defaults for the type of device and number of devices. RESINV=8000 specifies that approximately 8000 bytes are required for the system services (for example, GETMAIN and OPEN) that MYSORT's E15 exit routine performs.

28 OMIT statement. FORMAT specifies that the compare fields are floating-sign. COND specifies that input records with equal 8-byte floating-sign compare fields starting in position 5 (also the control field) and position 13 are to be omitted from the output data set.

29 This is the end of the control statements area.

30 This is the DCB for MYSORT's MSGOUT output.

31-33 This is MYSORT's E15 routine. The E15 routine loads the address of the GETMAINed work area from the second word of the E15 parameter list. The E15 routine must supply each input record by placing its address in register 1 and placing a 12 (insert) in register 15. When all the records have been passed, the E15 routine must place an 8 ("do not return") in register 15.

Example 10. Sort with OUTFIL

INPUT Fixed-length record data set

OUTPUT
Multiple fixed-length record data sets

WORK DATA SETS
Dynamically allocated (by default)

USER EXITS
None

FUNCTIONS/OPTIONS
OUTFIL

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//OUTFIL   EXEC  PGM=SORT                                       02
//SYSOUT   DD  SYSOUT=A                                         03
//SORTIN   DD DSN=GRP.RECORDS,DISP=SHR                          04
//ALLGPS   DD DSN=GRP.ALLGRPS,DISP=OLD                          05
//ALLBU    DD DSN=GRP.BU,DISP=(NEW,CATLG,DELETE),               06
//            UNIT=3390,SPACE=(TRK,(10,10))                     07
//G1STATS  DD SYSOUT=A                                          08
//G2STATS  DD SYSOUT=A                                          09
//SYSIN    DD    *                                              10
 SORT FIELDS=(6,5,CH,A)                                         11

 OUTFIL FNAMES=(ALLGPS,ALLBU)                                   12

 OUTFIL FNAMES=G1STATS,                                         13
   INCLUDE=(1,3,CH,EQ,C'G01'),                                  14
   HEADER2=(1:'GROUP 1 STATUS REPORT FOR ',&DATE,               15
              ' - PAGE ',&PAGE,2/,                              16
            6:'ITEM ',16:'STATUS      ',31:'PARTS',/,           17
            6:'-----',16:'------------',31:'-----'),            18
    OUTREC=(6:6,5,                                              19
            16:14,1,CHANGE=(12,                                 20
                            C'1',C'SHIP',                       21
                            C'2',C'HOLD',                       22
                            C'3',C'TRANSFER'),                  23
                    NOMATCH=(C'*CHECK CODE*'),                  24
            31:39,1,BI,M10,LENGTH=5,                            25
            120:X)                                              26

 OUTFIL FNAMES=G2STATS,                                         27
   INCLUDE=(1,3,CH,EQ,C'G02'),                                  28
   HEADER2=(1:'GROUP 2 STATUS REPORT FOR ',&DATE,               29
              ' - PAGE ',&PAGE,2/,                              30
            6:'ITEM ',16:'STATUS      ',31:'PARTS',/,           31
            6:'-----',16:'------------',31:'-----'),            32
    OUTREC=(6:6,5,                                              33
            16:14,1,CHANGE=(12,                                 34
                            C'1',C'SHIP',                       35
                            C'2',C'HOLD',                       36
                            C'3',C'TRANSFER'),                  37
                    NOMATCH=(C'*CHECK CODE*'),                  38
            31:39,1,BI,M10,LENGTH=5,                            39
            120:X)                                              40
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias name SORT.

03 SYSOUT DD statement. Directs DFSORT messages and control statements to sysout class A.

04 SORTIN DD statement. The input data set is named GRP.RECORDS and is cataloged. DFSORT determines from the data set label that the RECFM is FB, the LRECL is 80 and the BLKSIZE is 23440.

05 ALLGPS DD statement. The first OUTFIL output data set is named GRP.ALLGRPS and is catalogued. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.

06-07 ALLBU DD statement. The second OUTFIL output data set is named GRP.BU and is to be allocated on a 3390 and catalogued. DFSORT sets the RECFM and LRECL from SORTIN and selects an appropriate BLKSIZE.

08 G1STATS DD statement. The third OUTFIL output data set is directed to sysout class A. Since this is an OUTFIL report data set, DFSORT sets the RECFM to FBA (FB from SORTIN and A for ASA control characters) and the LRECL to 121 (1 byte for the ASA control character and 120 bytes for the data). DFSORT sets an appropriate BLKSIZE.

09 G2STATS DD statement. The fourth OUTFIL output data set is directed to sysout class A. Since this is an OUTFIL report data set, DFSORT sets the RECFM to FBA (FB from SORTIN and A for ASA control characters) and the LRECL to 121 (1 byte for the ASA control character and 120 bytes for the data). DFSORT sets an appropriate BLKSIZE.

10 SYSIN DD statement. DFSORT control statements follow.

11 SORT statement. FIELDS specifies an ascending 5-byte character control field starting at position 6.

12 OUTFIL statement. The sorted input records are written to the ALLGPS and ALLBU data sets.

13-26 OUTFIL statement. The subset of sorted input records containing ’G01’ in positions 1 through 3 are used to produce a report which is written to the G1STATS data set.

27-40 OUTFIL statement. The subset of sorted input records containing ’G02’ in positions 1 through 3 are used to produce a report which is written to the G2STATS data set.

Example 11. Sort with SmartBatch Pipes and OUTFIL SPLIT

INPUT SmartBatch Pipes

OUTPUT
SmartBatch pipes

WORK DATA SETS
Dynamically allocated

USER EXITS
None

FUNCTIONS/OPTIONS
FILSZ, OUTFIL, DYNALLOC

```
//EXAMP   JOB A400,PROGRAMMER                                  01
//RUNSORT EXEC PGM=ICEMAN                                      02
//SYSOUT  DD SYSOUT=H                                          03
//SORTIN  DD DSN=INPUT.PIPE,SUBSYS=PIPE,                       04
//           DCB=(LRECL=60,RECFM=FB,BLKSIZE=32760)             05
//OUT1    DD DSN=OUTPUT.PIPE1,SUBSYS=PIPE,                     06
//           DCB=(LRECL=60,RECFM=FB,BLKSIZE=32760)             07
//OUT2    DD DSN=OUTPUT.PIPE2,SUBSYS=PIPE,                     08
//           DCB=(LRECL=60,RECFM=FB,BLKSIZE=32760)             09
//SYSIN   DD *                                                 10
  OPTION DYNALLOC,FILSZ=U1000000                               11
  SORT FIELDS=(1,20,CH,A,25,4,BI,A)                            12
  OUTFIL FNAMES=(OUT1,OUT2),SPLIT                              13
```

Line	Explanation
01	Job statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly.
03	SYSOUT DD statement. Directs DFSORT messages and control statements to system output class H.
04-05	SORTIN DD statement. The SUBSYS=PIPE parameter directs the allocation to the ’PIPE’ SmartBatch subsystem for the pipe named INPUT.PIPE. The DCB statement describes the data set characteristics to subsystem PIPE.
06-07	OUT1 DD statement. The SUBSYS=PIPE parameter directs the allocation to the ’PIPE’ SmartBatch subsystem for the pipe named OUTPUT.PIPE1. The DCB statement describes the data set characteristics to subsystem PIPE.
08-09	OUT2 DD statement. The SUBSYS=PIPE parameter directs the allocation to the ’PIPE’ SmartBatch subsystem for the pipe named OUTPUT.PIPE2. The DCB statement describes the data set characteristics to subsystem PIPE.
10	SYSIN DD statement. DFSORT control statements follow.
11	OPTION statement. DYNALLOC specifies that work data sets are to be dynamically allocated using the installation defaults for type of device and number of devices. FILSZ=U1000000 specifies an estimate of one million input records.
12	SORT statement. FIELDS specifies an ascending 20-byte character control field starting at position 1 and an ascending 4 byte binary control field starting at position 25.
13	OUTFIL statement. The records from the SORTIN pipe are sorted and

written alternatively to the OUT1 and OUT2 pipes (that is, the sorted records are split evenly between the two output pipes).

Example 12. Sort with INCLUDE and LOCALE

INPUT Fixed-length record data set

OUTPUT
Fixed-length record data set

WORK DATA SETS
Dynamically allocated (by default)

USER EXITS
None

FUNCTIONS/OPTIONS
INCLUDE, LOCALE

```
//EXAMP    JOB  A400,PROGRAMMER                                01
//STEP1    EXEC PGM=SORT,PARM='LOCALE=FR_CA'                    02
//STEPLIB  DD  DSN=SYS1.SCEERUN,DISP=SHR                        03
//SYSOUT   DD  SYSOUT=A                                         04
//SORTIN   DD  DSN=INPUT.FRENCH.CANADA,DISP=SHR                 05
//SORTOUT  DD  DSN=OUTPUT.FRENCH.CANADA,DISP=OLD                06
//SYSIN    DD  *                                                07
 SORT FIELDS=(1,20,CH,A,25,1,BI,D,30,10,CH,A)                   08
 INCLUDE COND=(40,6,CH,EQ,50,6,CH)                              09
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias name SORT. LOCALE specified in EXEC PARM overrides installation default for LOCALE. The locale for the French language and the cultural conventions of Canada will be active.
03	STEPLIB DD statement. Specifies the Language Environment run-time library containing the dynamically loadable locales.
04	SYSOUT statement. Directs DFSORT messages and control statements to sysout class A.
05	SORTIN DD statement. The input data set is named INPUT.FRENCH.CANADA and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.
06	SORTOUT DD statement. The output data set is named OUTPUT.FRENCH.CANADA and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.
07	SYSIN DD statement. DFSORT control statements follow.
08	SORT statement. FIELDS specifies an ascending 20-byte character control field starting at position 1, a one-byte descending binary control field starting at position 25, and a 10-byte ascending character control field starting at position 30. The character (CH) control fields will be sorted according to the collating rules defined in locale FR_CA.
09	INCLUDE statement. COND specifies that only input records with equal

6-byte character compare fields starting in position 40 and position 50 are to be included in the output data set. The character (CH) compare fields will be compared according to the collating rules defined in locale FR_CA.

Example 13: Sort with HFS Files

INPUT Concatenated HFS Files

OUTPUT
HFS File

WORK DATA SETS
Dynamically allocated (by default)

USER EXITS
None

FUNCTIONS/OPTIONS
None

```
//EXAMP   JOB A400,PROGRAMMER                                    01
//S1      EXEC PGM=SORT                                          02
//SYSOUT  DD SYSOUT=A                                            03
//SORTIN  DD  PATH='/user/hfs.inp1.txt',PATHOPTS=ORDONLY,        04
//        LRECL=80,BLKSIZE=240,RECFM=FB,FILEDATA=TEXT            05
//        DD  PATH='/user/hfs.inp2.txt',PATHOPTS=ORDONLY,        06
//        LRECL=80,BLKSIZE=80,RECFM=F,FILEDATA=TEXT              07
//SORTOUT DD  PATH='/user/hfs.ut.txt',PATHOPTS=OWRONLY,          08
//        LRECL=80,BLKSIZE=80,RECFM=F,FILEDATA=TEXT              09
//SYSIN DD *                                                     10
  SORT FIELDS=(10,8,CH,A)                                        11
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias name SORT.
03	SYSOUT DD statement. Directs DFSORT messages and control statements to system output class A.
04-05	SORTIN DD statement. The first input file is an HFS file named /user/hfs.inp1.txt. Only read access is allowed. The file is defined as a text file. It has fixed-length records with a record size of 80 and a block size of 240.
06-07	The second input file is an HFS file named /user/hfs.inp2.txt. Only read access is allowed. The file is defined as a text file. It has fixed-length records with a record size of 80 and a block size of 80.
08-09	SORTOUT DD statement. The output file is an HFS file named /user/hfs.ut.txt. Only write access is allowed. The file is defined as a text file. It has fixed-length records with a record size of 80 and a block size of 80.
10	SYSIN DD statement. DFSORT control statements follow.
11	SORT statement. FIELDS specifies an ascending 8-byte character control field starting at position 10.

Merge Examples

This section contains two merge examples.

Example 1. Merge with EQUALS

INPUT Blocked fixed-length records on DASD

OUTPUT
Blocked fixed-length records on 3390

WORK DATA SETS
Not applicable

USER EXITS
None

FUNCTIONS/OPTIONS
EQUALS

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//STEP1    EXEC PGM=SORT                                        02
//SYSOUT   DD   SYSOUT=A                                        03
//SORTIN01 DD   DSN=M1234.INPUT1,DISP=SHR                       04
//SORTIN02 DD   DSN=M1234.INPUT2,DISP=SHR                       05
//SORTIN03 DD   DSN=M1234.INPUT3,DISP=SHR                       06
//SORTOUT  DD   DSN=M1234.MERGOUT,DISP=(,KEEP),                 07
//  SPACE=(CYL,(2,4)),UNIT=3390                                 08
//SYSIN    DD *                                                 09
    MERGE FIELDS=(1,8,CH,A,20,4,PD,A)                           10
    OPTION EQUALS                                               11
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias SORT.
03	SYSOUT DD statement. Directs DFSORT messages and control statements to sysout class A.
04	SORTIN01 DD statement. The first input data set is named M1234.INPUT1 and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.
05	SORTIN02 DD statement. The second input data set is named M1234.INPUT2 and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.
06	SORTIN03 DD statement. The third input data set is named M1234.INPUT3 and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.
07-08	SORTOUT DD statement. The output data set is named M1234.MERGOUT and is to be allocated on 3390 and kept. DFSORT sets the RECFM and LRECL from the SORTINnn data sets and selects an appropriate BLKSIZE.
09	SYSIN DD statement. DFSORT control statements follow.
10	MERGE statement. FIELDS specifies an ascending 8-byte character control

field starting at position 1 and an ascending 4-byte packed-decimal field starting at position 20. The records in each input data set must already be in the order specified.

25 OPTION statement. EQUALS specifies that the order of output records with equal control fields is to be based on the file number of the input data sets and the original order of the records within each input data set.

Example 2. Merge with LOCALE and OUTFIL

INPUT Fixed-length record data set

OUTPUT
Multiple fixed-length record data sets

WORK DATA SETS
Not applicable

USER EXITS
None

FUNCTIONS/OPTIONS
LOCALE, OUTFIL

```
//EXAMP    JOB  A400,PROGRAMMER                                01
//STEP1    EXEC PGM=SORT                                       02
//STEPLIB  DD  DSN=SYS1.SCEERUN,DISP=SHR                       03
//SYSOUT   DD  SYSOUT=A                                        04
//SORTIN01 DD  DSN=INPUT01.GERMAN.GERMANY,DISP=SHR             05
//SORTIN02 DD  DSN=INPUT02.GERMAN.GERMANY,DISP=SHR             06
//SORTIN03 DD  DSN=INPUT03.GERMAN.GERMANY,DISP=SHR             07
//GP1      DD  DSN=OUTPUT.GERMAN.GP1,DISP=OLD                  08
//GP2      DD  DSN=OUTPUT.GERMAN.GP2,DISP=OLD                  09
//GP3      DD  DSN=OUTPUT.GERMAN.SAVE,DISP=OLD                 10
//DFSPARM  DD  *                                               11
 LOCALE=De_DE.IBM-1047                                         12
 MERGE FIELDS=(25,5,CH,A,40,4,PD,D)                            13
 OUTFIL FNAMES=GP1,                                            14
   INCLUDE=(23,1,CH,LE,C'Ö')                                   15
 OUTFIL FNAMES=GP2,                                            16
   INCLUDE=(23,1,CH,GT,C'Ö',AND,                               17
           23,1,CH,LT,C'Ü')                                    18
 OUTFIL FNAMES=GP3,SAVE                                        19
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias name SORT.
03	STEPLIB DD statement. Specifies the Language Environment run-time library containing the dynamically loadable locales.
04	SYSOUT statement. Directs DFSORT messages and control statements to sysout class A.
05	SORTIN01 DD statement. The first input data set is named INPUT01.GERMAN.GERMANY and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.
06	SORTIN02 DD statement. The second input data set is named

INPUT02.GERMAN.GERMANY and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.

07 SORTIN03 DD statement. The third input data set is named INPUT03.GERMAN.GERMANY and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.

08 GP1 DD statement. The first OUTFIL output data set is named OUTPUT.GERMAN.GP1 and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.

09 GP2 DD statement. The second OUTFIL output data set is named OUTPUT.GERMAN.GP2 and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.

10 GP3 DD statement. The third OUTFIL output data set is named OUTPUT.GERMAN.GP3 and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.

11 DFSPARM DD statement. DFSORT control statements follow.

12 LOCALE parameter. Overrides installation default for LOCALE. The locale for the German language and the cultural conventions of Germany based on the IBM-1047 encoded character set will be active.

13 MERGE statement. FIELDS specifies an ascending 5-byte character control field starting at position 25, and a descending 4-byte packed decimal control field starting at position 40. The character (CH) control field will be merged according to the collating rules defined in locale De_DE.IBM-1047. The records in each input data set must already be in the order specified.

14-15 OUTFIL statement. The subset of records with character values less than or equal to 'Ö' in position 23 are written to the GP1 output data set. The character (CH) compare field and character constant will be compared according to the collating rules defined in locale De_DE.IBM-1047.

16-18 OUTFIL statement. The subset of records with character values greater than 'Ö' but less than 'Ü' in position 23 are written to the GP2 output data set. The character (CH) compare fields and character constants will be compared according to the collating rules defined in locale De_DE.IBM-1047.

19 OUTFIL statement. Any records not written to the GP1 or GP2 output data sets are written to the GP3 output data set.

Copy Examples

This section contains 2 copy examples.

Example 1. Copy with EXEC PARMs, SKIPREC, MSGPRT and ABEND

INPUT Blocked fixed-length records on multivolume 3490

OUTPUT
Blocked fixed-length records on DASD

WORK DATA SETS
Not applicable

USER EXITS
None

FUNCTIONS/OPTIONS
SKIPREC, MSGPRT, ABEND

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//STEP1    EXEC   PGM=SORT,                                     02
//  PARM='SKIPREC=500,MSGPRT=CRITICAL,ABEND'                     03
//SYSOUT   DD SYSOUT=A                                          04
//SORTIN   DD DSN=FLY.RECORDS,VOL=SER=(000333,000343),          05
//  UNIT=(3490,2),DISP=OLD,LABEL=(,NL),                         06
//  DCB=(RECFM=FB,LRECL=12000,BLKSIZE=24000)                    07
//SORTOUT  DD DSN=FLY.RECORDS.COPY,DISP=OLD                     08
//SYSIN DD *                                                    09
  SORT FIELDS=COPY                                              10
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02-03	EXEC statement. Calls DFSORT directly by its alias SORT. SKIPREC=500 specifies that the first 500 input records are not to be included in the output data set. MSGPRT=CRITICAL specifies that error messages, but not informational messages, are to be printed. ABEND specifies that DFSORT is to terminate with a user ABEND if it issues an error message.
04	SYSOUT DD statement. Directs DFSORT messages and control statements to sysout class A.
05-07	SORTIN DD statement. The input data set is named FLY.RECORDS and resides on 3490 volumes 000333 and 000343. The UNIT parameter requests two tape drives, one for each volume of the data set. Because the tape is unlabeled, DCB parameters must be supplied to indicate that the RECFM is FB, the LRECL is 12000 and the BLKSIZE is 24000.
08	SORTOUT DD statement. The output data set is named FLY.RECORDS.COPY and is cataloged. DFSORT determines the RECFM, LRECL and BLKSIZE from the data set label.
09	SYSIN DD statement. DFSORT control statements follow.
10	SORT statement. FIELDS=COPY specifies a copy application.

Example 2. Copy with INCLUDE and VLSHRT

Copy Examples

INPUT Blocked spanned records on DASD

OUTPUT
Blocked spanned records on SYSDA

WORK DATA SETS
Not applicable

USER EXITS
None

FUNCTIONS/OPTIONS
INCLUDE, VLSHRT

```
//EXAMP    JOB A400,PROGRAMMER                                  01
//COPY     EXEC PGM=SORT                                        02
//SYSOUT   DD SYSOUT=A                                          03
//SORTIN   DD DSN=SMF.DATA,DISP=SHR                             04
//SORTOUT  DD DSN=SMF.VIOL,DISP=(,KEEP),SPACE=(CYL,(2,5)),      05
//  UNIT=SYSDA                                                  06
//SYSIN    DD *                                                 07
  INCLUDE COND=(6,1,FI,EQ,80,AND,19,1,BI,EQ,B'1.......')        08
  OPTION COPY,VLSHRT                                            09
```

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls DFSORT directly by its alias SORT.
03	SYSOUT DD statement. Directs DFSORT messages and control statements to sysout class A.
04	SORTIN DD statement. The input data set is named SMF.DATA and is cataloged. DFSORT determines from the data set label that the RECFM is VBS, the LRECL is 32760 and the BLKSIZE is 23476.
05-06	SORTOUT DD statement. The output data set is named SMF.VIOL and is to be allocated on SYSDA and kept. DFSORT sets the RECFM and LRECL from SORTIN and selects an appropriate BLKSIZE.
07	SYSIN DD statement. DFSORT control statements follow.
08	INCLUDE statement. COND specifies that only input records with decimal 80 in the 1-byte fixed-point field at position 6 and bit 0 on in the 1-byte binary field at position 19 are to be included in the output data set.
09	OPTION statement. COPY specifies a copy application. VLSHRT specifies that records which are too short to contain all of the INCLUDE compare fields are not to be included in the output data set.

ICEGENER Example

This section contains an ICEGENER example.

INPUT Same as for IEBGENER job

OUTPUT
Same as for IEBGENER job

WORK DATA SETS
Not applicable

USER EXITS
None

FUNCTIONS/OPTIONS
None

```
//EXAMP    JOB A400,PROGRAMMER                                01
//GENR     EXEC PGM=ICEGENER                                  02
//SYSPRINT DD SYSOUT=A                                        03
//SYSUT1   DD DSN=CTL.MASTER,DISP=SHR                         04
//SYSUT2   DD DSN=CTL.BACKUP,DISP=OLD                         05
//SYSIN    DD DUMMY                                           06
```

This example shows how to use the ICEGENER facility for an IEBGENER job if your site has not installed ICEGENER as an automatic replacement for IEBGENER. The ICEGENER facility selects the more efficient DFSORT copy function for this IEBGENER job.

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls the ICEGENER facility. PGM=IEBGENER has been replaced by PGM=ICEGENER.
03-06	No other changes to the IEBGENER job are required.

ICETOOL Example

This section contains an example of ICETOOL with various operators

ICETOOL Example

INPUT Multiple data sets

OUTPUT
Multiple data sets

WORK DATA SETS
Dynamically allocated (automatic)

USER EXITS
ICETOOL's E35 (automatic)

FUNCTIONS/OPTIONS
OCCUR, COPY, SORT, MODE, VERIFY, STATS, DISPLAY

```
//EXAMP    JOB A400,PROGRAMMER
//TOOLRUN  EXEC PGM=ICETOOL,REGION=1024K
//TOOLMSG  DD SYSOUT=A
//DFSMSG   DD SYSOUT=A
//TOOLIN   DD *
* Print report showing departments with less than 5 employees
  OCCUR  FROM(IN1) LIST(LT5) LOWER(5) BLANK -
    TITLE('Small Departments') PAGE -
    HEADER('Department') HEADER('Employees') -
    ON(45,3,CH)          ON(VALCNT)
* Copy and reformat selected records
   COPY USING(CJ69) FROM(IN1) TO(OUTJ69D)
   COPY USING(CJ82) FROM(IN1) TO(OUTJ82D)
* Sort/save/print the resulting combined data sets
   SORT FROM(CONCAT) TO(DEPTSD,DEPTSP) USING(ABCD)
* Do following operators even if a previous operator failed,
* but stop processing if a subsequent operator fails.
 MODE STOP
* Verify decimal fields
   VERIFY FROM(IN2) ON(22,6,PD) ON(30,3,ZD)
* Print statistics for record length and numeric fields
   STATS FROM(IN2) ON(VLEN) ON(22,6,PD) ON(30,3,ZD)
* Sort and produce total for each unique key
   SORT FROM(IN2) TO(OUT4) USING(CTL1)
* Print report containing:
*   - key and total for each unique key
*   - lowest and highest of the totals
  DISPLAY FROM(OUT4) LIST(LIST1) -
    TITLE('Unique key totals report') DATE TIME -
    ON(5,10,CH) ON(22,6,PD) ON(30,3,ZD) -
    MINIMUM('Lowest') MAXIMUM('Highest') PLUS
//LT5      DD SYSOUT=A
//CJ69CNTL DD *
* Select J69 employees, reformat fields, and insert text
   INCLUDE COND=(45,3,CH,EQ,C'J69')
   OUTREC FIELDS=(21,10,X,1,15,C'is in department J69',34X)
```

```
//CJ82CNTL DD *                                                37
* Select J82 employees, reformat fields, and insert text       38
  INCLUDE COND=(45,3,CH,EQ,C'J82')                             39
  OUTREC FIELDS=(21,10,X,1,15,C'is in department J82',34X)     40
//IN1      DD DSN=FLY.INPUT1,DISP=SHR                          41
//OUTJ69D  DD DSN=&&OUTJ69D,DISP=(,PASS),SPACE=(TRK,(10,10)),  42
// UNIT=SYSDA                                                  43
//OUTJ82D  DD DSN=&&OUTJ82D,DISP=(,PASS),SPACE=(TRK,(10,10)),  44
// UNIT=SYSDA                                                  45
//CONCAT   DD DSN=*.OUTJ69D,VOL=REF=*.OUTJ69D,DISP=(OLD,PASS)  46
//         DD DSN=*.OUTJ82D,VOL=REF=*.OUTJ82D,DISP=(OLD,PASS)  47
//ABCDCNTL DD *                                                48
* Sort by last name, first name                                49
  SORT FIELDS=(12,15,CH,A,1,10,CH,A)                           50
//DEPTSD   DD DSN=FLY.OUTPUT1,DISP=SHR                         51
//DEPTSP   DD SYSOUT=A                                         52
//IN2      DD DSN=FLY.INPUT2,DISP=SHR                          53
//OUT4     DD DSN=FLY.OUTPUT2,DISP=OLD                         54
//CTL1CNTL DD *                                                55
* Sort and produce totals in one record for each unique key    56
  SORT FIELDS=(5,10,CH,A)                                      57
  SUM FIELDS=(22,6,PD,30,3,ZD)                                 58
//LIST1    DD SYSOUT=A                                         59
```

This example shows how ICETOOL can be used to perform multiple operations in a single step.

Line	Explanation
01	JOB statement. Introduces this job to the operating system.
02	EXEC statement. Calls ICETOOL specifying the recommended REGION of 1024K.
03	TOOLMSG DD statement. Directs ICETOOL messages and statements to system output class A.
04	DFSMSG DD statement. Directs DFSORT messages and control statements to SYSOUT class A.
05	TOOLIN DD statement. ICETOOL statements follow. The MODE for the ICETOOL run is initially set to STOP. If an error is detected for an operator, SCAN mode will be entered.
06	Comment statement. Printed but otherwise ignored.
07-10	OCCUR operator. Prints, in the LT5 data set, a report detailing each value for the specified field in the IN1 data set and the number of times that value occurs.
11	Comment statement.
12	COPY operator. Records from the IN1 data set are copied to the OUTJ69D data set using the DFSORT control statements in the CJ69CNTL data set. As a result, &&OUTJ69D contains a reformatted subset of the records from FLY.INPUT1 (those records containing ’J69’ in the positions 45-47).
13	COPY operator. Records from the IN1 data set are copied to the OUTJ82D data set using the DFSORT control statements in the CJ82CNTL data set. As a result, &&OUTJ82D contains a reformatted subset of the records from FLY.INPUT1 (those records containing ’J82’ in the positions 45-47).
14	Comment statement.
15	SORT operator. Records from the CONCAT data sets are sorted to the

DEPTSD and DEPTSP data sets using the DFSORT control statements in the ABCDCNTL data set. As a result, FLY.OUTPUT1 and DEPTSP (SYSOUT) contain the sorted combined records from &&OUTJ69D and &&OUTJ82D

16-17 Comment statements.

18 MODE operator. The MODE is reset to STOP (needed in case SCAN mode was entered due to an error for a previous operator). If an error is detected for a subsequent operator, SCAN mode will be entered. This divides the previous operators and subsequent operators into two unrelated groups.

19 Comment statement.

20 VERIFY operator. Identifies invalid values, if any, in the specified decimal fields of the IN2 data set. Used to stop subsequent operations if any invalid value is found in FLY.INPUT2.

21 Comment statement.

22 STATS operator. Prints the minimum, maximum, average, and total for the specified fields of the IN2 data set.

ON(VLEN) operates on the record length of the records in FLY.INPUT2. Thus, the values printed for ON(VLEN) represent the shortest record, the longest record, the average record length, and the total number of bytes for FLY.INPUT2.

23 Comment statement.

24 SORT operator. Records from the IN2 data set are sorted and summarized to the OUT4 data set using the DFSORT control statements in the CTL1CNTL data set. As a result, FLY.OUTPUT2 contains one record from FLY.INPUT2 for each unique sort field with totals for the sum fields.

25-27 Comment statements.

28-31 DISPLAY operator. Prints, in the LIST1 data set, a report detailing each sort and sum value for the OUT4 data set resulting from the previous operation, and the lowest and highest value for each sum field.

32-59 DD statements. Defines the data sets and DFSORT control statements used for the ICETOOL operations described above.

Appendix A. Using Work Space

Introduction 539
Hiperspace 539
Work Data Set Devices 540
DASD and Tape Devices 540
Number of Devices 540
Non-Synchronous Storage Subsystems 541
Allocation of Work Data Sets 541
Dynamic Allocation of Work Data Sets 542
Automatic Dynamic Allocation 542
Device Defaults 543
File Size and Dynamic Allocation 543
Dynamic Over-Allocation of Work Space 544
JCL Allocation of Work Data Sets 545
DASD Capacity Considerations 546
Exceeding DASD Work Space Capacity 547
Tape Capacity Considerations 547
Exceeding Tape Work Space Capacity 547

Introduction

When a sort application cannot be performed entirely invirtual storage, DFSORT must use work space. The amount of work space required depends on:

- The amount of data being sorted
- The amount of virtual storage available to DFSORT
- The amount of Hiperspace available to DFSORT
- The type of devices you use
- The DFSORT functions and features you use (for example, VLSHRT, locale processing, EFS, and ALTSEQ can increase the amount of work space required).

There are three ways to supply work space for a DFSORT application:

- Hiperspace
- Dynamic allocation of work data sets
- JCL allocation of work data sets.

For best performance, an optimal amount of Hiperspace, in combination with dynamically allocated DASD work data sets, is strongly recommended. See “Use Hipersorting” on page 503 for more information on using the HIPRMAX option. The DYNAUTO installation option, or the DYNALLOC run-time option, can be used to dynamically allocate work data sets.

Hiperspace

Hiperspace is the most efficient form of intermediate storage for DFSORT. Using the default ICEMAC option HIPRMAX=OPTIMAL ensures that DFSORT will use Hiperspace for Hipersorting whenever possible. Sites can tune their definition of HIPRMAX=OPTIMAL through use of the ICEMAC parameters EXPMAX, EXPOLD, and EXPRES. See*Installation and Customization* for more information.

DFSORT’s use of Hiperspace depends upon the availability of expanded storage, the needs of other concurrent Hipersorting applications throughout the time the application runs, and the settings of the DFSORT installation options EXPMAX, EXPOLD, and EXPRES. Consequently, it is possible for the same application to use

© Copyright IBM Corp. 1973, 2000

varying amounts of Hiperspace from run to run. If enough Hiperspace is available, DFSORT uses Hiperspace exclusively for intermediate storage. If the amount of Hiperspace is insufficient, DFSORT uses a combination of Hiperspace and work data sets, or even work data sets alone.

DFSORT only uses Hipersorting when there is sufficient expanded storage to back all the DFSORT Hiperspace data. Hipersorting is very dynamic: multiple concurrent Hipersorting applications always know each other's expanded storage needs and never try to back their Hiperspaces with the same portion of expanded storage. In addition, DFSORT checks the available expanded storage throughout the run, and switches from using Hiperspace to using DASD work data sets when either an expanded storage shortage is predicted or the total Hipersorting activity on the system reaches the limits set by the DFSORT installation options EXPMAX, EXPOLD, and EXPRES.

Hipersorting requires that work data sets be available, as well as Hiperspace. DFSORT forces the use of dynamic allocation for Hipersorting if work data sets are not requested explicitly. For further details, see the HIPRMAX option of the "OPTION Control Statement" on page 134.

Work Data Set Devices

The type of device selected for work data sets can have a significant effect on performance. Consider the following when selecting devices for work data sets.

DASD and Tape Devices

For optimal performance, use direct access devices to which little other activity is directed for work data sets. Avoid using 3390-9 DASD, optical DASD, or tape devices for work data sets, if at all possible.

Using tape devices for work data sets rather than DASD causes significant performance degradation for the following reasons:

- Tape work data sets prevent DFSORT from using its more efficient sorting techniques, Blockset and Peerage/Vale. DASD work data sets allow DFSORT to use these techniques.
- Tape work data sets must be accessed sequentially. DASD data sets can be accessed randomly.
- DASD control units can provide additional features, such as cache fast write, that are not available with tape devices.

Number of Devices

Although one work data set is sufficient, using two or more work data sets on separate devices usually reduces the elapsed time of the application significantly. In general, using more than three work data sets does not reduce elapsed time any further, and is only necessary if the work data sets are small or the file size is large.

For optimum allocation of resources such as virtual storage, avoid specifying a large number of work data sets unnecessarily.

No more than 255 work data sets can be specified. If you specify more than 32 work data sets, and the Blockset technique is not selected, a maximum of 32 work data sets is used.

Non-Synchronous Storage Subsystems

Allocation of work data sets on devices attached to non-synchronous storage subsystems can affect the performance of certain DFSORT applications. Whether or not a particular application is affected depends on many factors, the most critical of which is the ratio of input file size to available storage.

In general, to maximize performance, DFSORT needs the following:

- Accurate knowledge of the size of the file being sorted

  In most cases, DFSORT is able to calculate the file size accurately. However, for applications that sort many input tapes that are not managed by DFSMSrmm (especially compacted input tapes) or that use E15 exits that add or delete records, we recommend that you specify the file size using the FILSZ or SIZE parameter.
- Adequate storage relative to the size of the file being sorted

  Table 62 shows the minimum recommended storage to provide DFSORT based on various input file sizes.

Table 62. Minimum Storage Required for Various File Sizes

Input file size	Minimum storage
Less than 200MB	4MB
200MB to 500MB	8MB
500MB to 1GB	16MB
More than 1GB	16-32MB

Under some circumstances, DFSORT does not perform as well when using ESCON channels as it does when using parallel channels. The two types of applications most likely to cause a noticeable decrease in performance are:

1. Applications where DFSORT cannot accurately determine the size of the file to be sorted. These applications often involve DFSORT E15 user exits that insert records into the sorting process.
2. Sort applications with a low ratio of available storage to input file size.

Allocation of Work Data Sets

Dynamic allocation has the following advantages over JCL allocation:

- ICEMAC can be set to automatically activate dynamic allocation for all sort applications.

  To use JCL allocation, appropriate DD statements must be specified for each individual application.
- As the characteristics (file size, virtual storage, and so on) of an application change over time, DFSORT can automatically optimize the amount of dynamically allocated work space for the application. This eliminates unneeded allocation of DASD space.

  JCL allocation is fixed; DFSORT cannot adjust it. DASD space might be wasted.
- As the amount of Hiperspace available to the application varies from run to run, DFSORT can automatically adjust the amount of space it dynamically allocates to complement the amount of Hiperspace. This eliminates unneeded allocation of DASD space.

  JCL allocation is fixed; DFSORT cannot adjust it, even if all sorting can be done in Hiperspace. DASD space might be wasted.

Allocation of Work Data Sets

Dynamic allocation has one drawback: for certain applications, as described in “File Size and Dynamic Allocation” on page 543, you might need to give DFSORT a reasonable estimate of the input file size. Later, if the input file size for the application increases significantly, you must update the file size estimate accordingly.

However, JCL allocation has a similar drawback, except that it applies to all applications. Unless you overallocate the work data sets initially and waste space, you have to update the JCL allocation when the input file size increases significantly for any application to avoid out-of-space abends.

If you can allocate enough work data set space with JCL to guarantee your applications will never exceed the space allocated, you do not need dynamic allocation. However, since efficient use of DASD space is usually desirable, dynamic allocation is recommended over JCL allocation.

For both dynamic allocation and JCL allocation:

- The amount of work space actually used will often be less than the amount allocated. DFSORT tries to minimize dynamic over-allocation while making certain that the application will not fail due to lack of space. With JCL allocation, you could minimize the amount of allocated space manually, but this might require changes to JCL allocation as the characteristics of the application change over time.
- Limiting the virtual storage available to DFSORT can increase the amount of work space required. With a reasonable amount of storage, 4MB for example, DFSORT can sort using a reasonable amount of work space. If storage is limited, more work space might be required. If storage is drastically limited (for example, to 200KB), significantly more work space might be required.

Dynamic Allocation of Work Data Sets

ICEMAC options are available to request automatic dynamic allocation of work data sets and to supply defaults for the device type and number of devices.

For certain applications, it is very important to specify a reasonable estimate of the input file size when using dynamic allocation.

Automatic Dynamic Allocation

Your system programmer has set the DYNAUTO installation option to control whether dynamic allocation is used automatically, or only when requested by the DYNALLOC run-time option.

DYNAUTO also controls whether dynamic allocation or JCL allocation takes precedence when JCL work data sets are specified.

If your system programmer selected DYNAUTO=IGNWKDD, dynamic allocation takes precedence over JCL allocation (JCL work data sets are actually deallocated). If you want the opposite precedence for selected applications, use the run-time option USEWKDD.

If your system programmer selected DYNAUTO=YES, JCL allocation takes precedence over dynamic allocation. If you want the opposite precedence, you must remove the JCL allocation statements.

If your system programmer selected DYNAUTO=NO, dynamic allocation of work data sets is not used unless you specify the DYNALLOC run-time option. JCL allocation takes precedence over dynamic allocation.

Device Defaults

When the device type, or the number of devices for dynamic allocation, is not explicitly specified, DFSORT obtains the missing information from the DYNALOC installation option information supplied by your system programmer.

File Size and Dynamic Allocation

DFSORT bases the amount of work space it dynamically allocates on the number of bytes to be sorted—the input file size. Generally, DFSORT can automatically make an accurate determination of the file size by determining the number of input records. However, DFSORT cannot always determine the input file size accurately in the following cases:

- An E15 user exit routine supplies all input records (an input data set is not present). DFSORT cannot automatically determine the number of records to be inserted.
- An input data set is present, along with an E15 user exit routine. DFSORT can automatically determine the number of records in the input data set, but cannot automatically determine the number of records to be inserted or deleted.
- A spool (DD *) or pipe data set is used as input.
- The input consists of small data sets on tape that are not managed by DFSMSrmm. When the tape data sets are not managed by DFSMSrmm, DFSORT cannot know how much of the tapes are used, so it determines the file size assuming full volumes at the maximum regular density for the drives.
- The Improved Data Recording Capability (IDRC) feature is used for the input device and the tape data sets are not managed by DFSMSrmm.
- Input data sets are members of partitioned data sets. DFSORT cannot determine the size of a member in a partitioned data set. Therefore, when input data sets are partitioned, DFSORT uses the size of the entire data set as the input file size. This is usually an over-estimation, which leads to over-allocation of work space.

In the above circumstances, DFSORT may dynamically over-allocate or under-allocate the work space, possibly leading to wasted space or an out-of-space abend, respectively. If this happens, you should specify either the exact number of records to be sorted or an estimate of the number of records to be sorted.

There are several ways to specify the number of records to be sorted at run-time, as explained below:

- Run-time option SIZE=n enables you to specify the exact number of records in the input data sets. If specified, DFSORT always uses this number to determine the file size and issues an error message if the actual number of input records does not match the given number (this happens *only* if FSZEST=NO is in effect). You must change SIZE=n whenever the number of records in the input data sets changes.
- Run-time option FILSZ=n works the same way as SIZE=n, except that the exact number you specify must take into account the number of records in the input data sets, and the number of records inserted and deleted before sorting using an E15 exit routine, an INCLUDE or OMIT control statement, or the SKIPREC or STOPAFT options.

Use FILSZ=n rather than SIZE=n if your application inserts or deletes a significant number of records before sorting. You must change FILSZ=n whenever the total number of records to be sorted changes.

- Run-time option SIZE=Un enables you to specify the number of records in the input data sets. DFSORT always uses your specified value to determine the file size, but does not issue an error message if the actual number of input records does not match the given number. As long as the supplied value is reasonably close to the actual number of records read in, DFSORT can dynamically allocate the work space efficiently. You only need to update the value when the number of input records changes significantly; the better the estimate is, the more efficient the allocation.
- Run-time option FILSZ=Un works the same way as SIZE=Un, except that the record count you specify should take into account not only the number of records in the input data sets, but also, if significant, the number of records inserted or deleted before sorting.
- Run-time option SIZE=En enables you to specify an estimate of the number of records in the input data sets. DFSORT only uses your estimate to determine the file size in the following instances:
  - An E15 user exit routine supplies all input records (an input data set is not present).
  - An input data set is present, along with an E15 user exit routine.
  - The input consists of small data sets on tape that are not managed by DFSMSrmm.
  - The IBM 3480/3490 Magnetic Tape Subsystem with the Improved Data Recording Capability (IDRC) feature is used for the input device.

  No error message is issued if the supplied estimate is incorrect.
- Run-time option FILSZ=En works the same way as SIZE=En, except that the record count you specify should take into account not only the number of records in the input data sets, but also, if significant, the number of records inserted or deleted before sorting.

For variable-length records, DFSORT uses one-half of the maximum record length to determine the input file size, unless you specify the AVGRLEN option. If your actual average record length is significantly different from one-half of the maximum record length, specifying AVGRLEN=n can prevent DFSORT from over- or under-allocating dynamic work space.

See “OPTION Control Statement” on page 134 for more information about the AVGRLEN, SIZE and FILSZ parameters.

Dynamic Over-Allocation of Work Space

DFSORT can dynamically over-allocate the work space even when you specify the number of records under the following circumstances:

- When you delete a significant number of records using:
  - An INCLUDE or OMIT statement, or the SKIPREC or STOPAFT option. Use of these statements and options does *not* force DFSORT to use a SIZE=En or FILSZ=En specification. DFSORT ignores the En value unless it cannot compute the input file size.
  - One or more partitioned data set members as input. DFSORT uses the size of the entire partitioned data set rather than the size of the member in its calculations. DFSORT ignores any SIZE=En or FILSZ=En value unless it cannot determine the input file size itself.

You can avoid over-allocation in these cases by specifying SIZE=Un or FILSZ=Un.

- When the average record length of variable-length records is substantially shorter than one-half of the maximum record length. If DFSORT uses your exact or estimated number of records, it uses one-half of the maximum record length to determine the file size. You can avoid over-allocation in this case by specifying AVGRLEN=n.

Dynamic over-allocation of work space can occur when you do not specify the number of records (for example, with small input data sets on tape), or even when you do (for example, when a significant number of records is deleted). In these cases, you might prefer to use JCL allocation of work data sets to control the amount of space allocated. However, there are drawbacks to doing so, as previously explained. If DYNAUTO=IGNWKDD is used, remember to specify run-time option USEWKDD when you want to use JCL allocation of work data sets.

JCL Allocation of Work Data Sets

The amount of required work space is dependent on many factors such as virtual storage and type of devices used, but is especially sensitive to the file size of the input data set.

Because of the number of variables involved, an exact formula cannot be given for calculating the needed work space. However, the following guidelines usually hold true:

- For fixed length record (FLR) sort applications, 1.5 to 2 times the input file size is usually adequate.
- For variable-length record (VLR) sort applications, 1.5 to 2.5 times the input file size is usually adequate.

These guidelines assume that a reasonable amount of storage (at least 1M) is available to DFSORT. Limiting the available amount of storage can increase the amount of needed work space.

DFSORT can often run with less than the amount of work space indicated by the above guidelines.

To get the best performance using JCL allocation of work data sets:

- Use devices without much activity on them.
- For best performance, use emulated 3390-9 devices (such as RAMAC) or other high-speed IBM DASD devices for work data sets, and avoid using tape, virtual (VIO) or real 3390-9 devices for work data sets.
- Allocate space in cylinders.
- Specify contiguous space for each work data set, and make sure there is enough primary space so that secondary space is not needed.
- Allocate two or more work data sets.
- Assign one work data set per actuator.
- Use multiple channel paths to the devices.
- Use different spindles and separate channel paths for the work data sets and the input/output data sets.

The following table shows the work data set space needed with 4M of storage for applications with various characteristics when Hipersorting and dataspace sorting are not used (HIPRMAX=0 and DSPSIZE=0).

Allocation of Work Data Sets

Table 63. Work Space Requirements for Various Input Characteristics

Input Data set Characteristics				Cylinders (3390)	
Filesize (MB)	**FLR/VLR**	**Max LRECL**	**BLKSIZE**	**Input Data Set**	**Work Data Set**
4	FLR	80	27920	6	6
4	FLR	160	27840	6	6
20	FLR	80	27920	26	36
20	FLR	160	27840	26	36
20	FLR	1000	27000	26	36
40	FLR	80	27920	51	56
40	FLR	160	27840	51	56
40	FLR	1000	27000	52	56
150	FLR	160	27840	189	198
4	VLR	300	27998	6	9
40	VLR	300	27998	51	63
40	VLR	6000	27998	55	59
150	VLR	300	27998	188	200
150	VLR	6000	27998	203	200

DASD Capacity Considerations

You can specify a mixture of direct access devices for a given sort application. Any IBM DASD device supported by your operating system can be used for work data sets.

For best performance, use emulated 3390-9 devices (such as RAMAC) or other high-speed IBM DASD devices for work data sets, and avoid using real 3390-9 devices for work data sets.

System performance is improved if work data sets arespecified in cylinders, rather than tracks or blocks. Storage on temporary work data sets will be readjusted to cylinders if possible. The number of tracks per cylinder for direct access devices is shown in Table 64.

Table 64. Number of Tracks per Cylinder for Direct Access Devices

Device	Tracks per Cylinder	Maximum Bytes used per Track
3380	15	47476
3390	15	56664
9345	15	46456

If WRKSEC is in effect and the work data set is not allocated to VIO, DFSORT allocates secondary extents as required, even if not requested in the JCL.

Exceeding DASD Work Space Capacity

If during sorting, the allocation of secondary space on one of the work data sets fails, the system issues a B37 informational message. DFSORT can recover by allocating space on one of the other work data sets, if one is available.

DFSORT normally allocates secondary extents for work data sets, even if not requested in the JCL. This reduces the probability of exceeding work space capacity.

If the DASD work space is not sufficient to perform the sort, DFSORT issues a message and terminates.

Tape Capacity Considerations

Any IBM tape device supported by your operating system can be used for work space. However, using tape devices for work data sets rather than DASD causes significant performance degradation and should, therefore, be avoided.

Three different tape work data set techniques are available to DFSORT: Balanced, Polyphase, and Oscillating. For information on how to calculate their requirements, see Table 65.

Note: The value you obtain for “min” is literally a minimum value; if, for example, your input uses a more efficient blocking factor than DFSORT or is spanned, you need more work space. Space requirements are also summarized in Table 65. DFSORT selects the most appropriate tape technique using these criteria.

Table 65. Work Space Requirements of the Various Tape Techniques

Tape Technique	Maximum Input	Work Space Areas Required	Max. No. of Work Areas	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.

Note:

V = Number of input volumes. Number of input volumes of blocking equals work space blocking.

Exceeding Tape Work Space Capacity

At the beginning of a sort using tape work data sets, DFSORT estimatesthe maximum sort capacity (Nmax) and issues message ICE038I. See the explanation of this message for details.

The value for Nmax 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 space, sort capacity may be exceeded at a lower value because of the higher number of partly empty, end-of-string blocks.

DASD Capacity Considerations

For magnetic tape, a tape length of 2400 feet is assumed in calculating Nmax. 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 user 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 work space. If the estimate of the input data set size was high, enough work space 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.

If you do not include an E16 routine, DFSORT continues to process records for as long as possible. If the work space is sufficient to contain all the records in the input data set, DFSORT completes normally; when work space is not sufficient, DFSORT issues a message and terminates.

The program generates a separate message for each of the three possible error conditions. They are:

1. **ICE041A—N GT NMAX:** Generated before sorting begins when the exact file size is greater than Nmax.
2. **ICE046A—SORT CAPACITY EXCEEDED:** Generated when the sort has used all available work space.
3. **ICE048I—NMAX EXCEEDED:** Generated when the 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. Specification/Override of DFSORT Options

"Installation Defaults" on page 16 discusses DFSORT's installation (ICEMAC) options and environments, and shows you how to use ICETOOL's DEFAULTS operator to list the installation defaults selected at your site.

Listed below are the places in DFSORT where you can specify various options that will override the IBM-supplied 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.

Directly 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, TSO OR TDx).

Program 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, TSOINV or TDx).

Notes:

1. For the DEBUG and OPTION statements, override is at the option level. For example, with:

   ```
   //DFSPARM DD *
      OPTION EQUALS
   //SYSIN DD *
      OPTION NOEQUALS,SKIPREC=50
   ```

   EQUALS from DFSPARM overrides NOEQUALS from SYSIN, but SKIPREC=50 from SYSIN in not affected by the OPTION statement in DFSPARM, so both EQUALS and SKIPREC=50 will be used.

   For control statements other than DEBUG and OPTION, override is at the statement level. For example, with:

   ```
   //DFSPARM DD *
     MODS E15=(CHECK,4096,EXIT)
   //SYSIN DD *
     MODS E35=(MOVE,2048,EXITX)
   ```

© Copyright IBM Corp. 1973, 2000

the MODS statement in DFSPARM completely overrides the MODS statement in SYSIN, so the E15 exit will be used, but the E35 exit will not.

2. 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.
3. For OUTFIL statements, override is at the ddname level. See “OUTFIL Statements Notes” on page 236 for details.

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 direct or program 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 direct 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 program invocation only.
- Accepts all DFSORT program control statements.
- Ignores these OPTION statement parameters: EFS, LIST, NOLIST, LISTX, NOLISTX, LOCALE, MSGPRT, MSGDDN, SMF, SORTDD, SORTIN, SORTOUT, and USEWKDD.
- 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 direct invocation only.
- Accepts all DFSORT program control statements.
- Ignores these OPTION statement parameters: EFS, LIST, NOLIST, LISTX, NOLISTX, LOCALE, MSGPRT, MSGDDN, SMF, SORTDD, SORTIN, SORTOUT, and USEWKDD.
- Permits comment statements, blank statements, and remarks.
- Can contain user exit routines in object deck format for link-editing.

Parameter Lists

- Use with program 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 user 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.

Directly Invoked DFSORT

Table 66 on page 552 shows where each sort, merge, or copy option may be specified when DFSORT is directly invoked (that is, not invoked by programs).

DFSPARM: PARM options selectively override corresponding options in any other source. DEBUG and OPTION control statement options selectively override corresponding options in EXEC PARM and SYSIN. Control statements other than DEBUG and OPTION completely override corresponding control statements in SYSIN.

EXEC PARM options selectively override options in SYSIN.

SORT and MERGE are considered to be corresponding control statements.

INCLUDE and OMIT are considered to be corresponding control statements.

Table 66. Directly Invoked DFSORT Option Specification/Override. Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL, TSO OR TDx	Description of Option	Function
NO	NO	NO	ABCODE	ABEND code	S,M,C
DEBUG ABSTP	NO	DEBUG ABSTP	NO	Abnormal stop	S,M,C
ALTSEQ CODE	NO	ALTSEQ CODE	ALTSEQ	Alternate sequence	S,M
PARM ARESALL OPTION ARESALL	ARESALL	OPTION ARESALL	ARESALL	System storage above 16MB virtual	S,M,C
DEBUG NOASSIST	NO	DEBUG NOASSIST	NO	Bypass Sorting Instructions	S
PARM AVGRLEN OPTION AVGRLEN	AVGRLEN	OPTION AVGRLEN	NO	Average record length	S
PARM BSAM DEBUG BSAM	BSAM	DEBUG BSAM	NO	Force BSAM	S,M,C
DEBUG CFW|NOCFW	NO	DEBUG CFW|NOCFW	CFW	Cache fast write	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
PARM COBEXIT OPTION COBEXIT	COBEXIT	OPTION COBEXIT	COBEXIT	COBOL library	S,M,C
INCLUDE|OMIT COND|FORMAT	NO	INCLUDE|OMIT COND|FORMAT	NO	Include|Omit fields	S,M,C
OPTION COPY SORT|MERGE FIELDS	NO	OPTION COPY SORT|MERGE FIELDS	NO	Copy records	C
DEBUG CTRx	NO	DEBUG CTRx	NO	ABEND record count	S,M

Table 66. Directly Invoked DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL, TSO OR TDx	Description of Option	Function
NO	NO	NO	Time-of-day for activation	Simulate SORTDIAG DD Statement	S,M,C
NO	NO	NO	DIAGSIM	Simulate SORTDIAG DD Statement	S,M,C
NO	NO	NO	DSA	Dynamic storage adjustment limit	S
PARM DSPSIZE OPTION DSPSIZE	DSPSIZE	OPTION DSPSIZE	DSPSIZE	Dataspace sorting	S
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	DYNALOC[1]	Dynamic SORTWKs	S
PARM DYNALLOC OPTION DYNALLOC|USEWKDD SORT DYNALLOC	DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	DYNAUTO	Automatic dynamic allocation	S
NO	NO	NO	DYNSPC	Dynamic allocation default space	S
PARM EFS OPTION EFS	EFS	NO[2]	EFS	EFS program specified	S,M,C
NO	NO	NO	ENABLE	Enable Time-of-Day modules	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

Table 66. Directly Invoked DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL, TSO OR TDx	Description of Option	Function
PARM ABEND|NOABEND DEBUG ABEND|NOABEND	ABEND|NOABEND	DEBUG ABEND|NOABEND	ERET	Error action	S,M,C
DEBUG ESTAE|NOESTAE	NO	DEBUG ESTAE|NOESTAE	ESTAE	ESTAE routine	S,M,C
OPTION EXITCK	NO	OPTION EXITCK	EXITCK	E15/E35 return code checking	S,M,C
NO	NO	NO	EXPMAX	Available expanded storage limit for all DFSORT Hiperspaces	S
NO	NO	NO	EXPOLD	Old expanded storage limit for all DFSORT Hiperspaces	S
NO	NO	NO	EXPRES	Available expanded storage reserved for non-Hipersorting use	S
PARM E15=COB PARM E35=COB MODS Exx|HILEVEL=YES	E15=COB E35=COB	MODS Exx|HILEVEL=YES	NO	User Exit Exx (xx=11,15-19,31,35,37-39, and 61)	S,M,C[3]
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
SUM FIELDS/FORMAT	NO	SUM FIELDS/FORMAT	NO	Sum fields	S,M
MERGE FILES	NO	MERGE FILES	NO	Merge input files	M

Table 66. Directly Invoked DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL, TSO OR TDx	Description of Option	Function
PARM FILSZ OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	FILSZ	OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	FSZEST	File size	S,M
PARM HIPRMAX OPTION HIPRMAX	HIPRMAX	OPTION HIPRMAX	HIPRMAX	Hipersorting	S
NO	NO	NO	IDRCPCT	IDRC compaction	S
NO	NO	NO	IEXIT	ICEIEXIT	S,M,C
OPTION CKPT[4] SORT CKPT[4]	NO	OPTION CKPT[4] SORT CKPT[4]	IGNCKPT	Checkpoints	S
NO	NO	NO	IOMAXBF	Maximum SORTIN/SORTOUT data set buffer space	S,M,C
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
PARM LOCALE OPTION LOCALE	LOCALE	NO[2]	LOCALE	Locale processing	S,M,C
NO	NO	NO	MAXLIM	Maximum storage below 16MB 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

Table 66. Directly Invoked DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL, TSO OR TDx	Description of Option	Function
NO	NO	NO	MSGCON	Write messages on master console	S,M,C
PARM MSGPRT OPTION MSGPRT	MSGPRT	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 ODMAXBF OPTION ODMAXBF	ODMAXBF	OPTION ODMAXBF	ODMAXBF	Maximum OUTFIL data set buffer space	S,M,C
OUTFIL[9]	OUTFIL[9]	OUTFIL[9]	NO	OUTFIL processing	S,M,C
PARM OUTREL|NOOUTREL OPTION NOOUTREL	OUTREL|NOOUTREL	OPTION NOOUTREL	OUTREL	Release output data set space	S,M,C
OPTION NOOUTSEC	NO	OPTION NOOUTSEC	OUTSEC	Output data set secondary allocation	S,M,C
NO	NO	NO	OVERRGN	Storage over REGION	S,M,C
PARM OVFLO OPTION OVFLO	OVFLO	OPTION OVFLO	OVFLO	Summary fields overflow action	S,M
PARM PAD OPTION PAD	PAD	OPTION PAD	PAD	DFSORT LRECL padding action	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

Table 66. Directly Invoked DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL, TSO OR TDx	Description of Option	Function
PARM RESET|NORESET OPTION RESET|NORESET	RESET|NORESET	OPTION RESET|NORESET	RESET	NEW or MOD VSAM output	S,M,C
PARM SDB OPTION SDB	SDB	OPTION SDB	SDB	System-determined output data set block size	S,M,C
NO	NO	NO	SDBMSG	System-determined block size for message and list data sets	S,M,C
PARM SIZE OPTION MAINSIZE	SIZE	OPTION MAINSIZE	SIZE	Storage	S,M,C
PARM SKIPREC OPTION SKIPREC SORT SKIPREC	SKIPREC	OPTION SKIPREC SORT SKIPREC	NO	Skip records	S,C
OPTION SMF	NO	NO	SMF	SMF records	S,M,C
PARM SOLRF|NOSOLRF OPTION SOLRF|NOSOLRF	SOLRF|NOSOLRF	OPTION SOLRF|NOSOLRF	SOLRF	SORTOUT length	S,M,C
OPTION SORTDD	NO	NO[2]	NO	ddname prefix	S,M,C
OPTION SORTIN[7]	NO	NO[2]	NO	Alternate SORTIN ddname	S,C
NO	NO	NO	SORTLIB	Conventional modules library	S,M
OPTION SORTOUT[8]	NO	NO[2]	NO	Alternate SORTOUT ddname	S,M,C
PARM SPANINC OPTION SPANINC	SPANINC	OPTION SPANINC	SPANINC	Incomplete spanned records action	S,M,C

Table 66. Directly Invoked DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL, TSO OR TDx	Description of Option	Function
PARM STIMER|NOSTIMER OPTION NOSTIMER	STIMER|NOSTIMER	OPTION NOSTIMER	STIMER	Use of STIMER	S,M,C
PARM STOPAFT OPTION STOPAFT SORT STOPAFT	STOPAFT	OPTION STOPAFT SORT STOPAFT	NO	Input limit	S,C
NO	NO	NO	SVC	DFSORT SVC Information	S,M,C
PARM SZERO|NOSZERO OPTION SZERO|NOSZERO	SZERO|NOSZERO	OPTION SZERO|NOSZERO	SZERO	Signed or unsigned zero	S,M,C
NO	NO	NO	TEXIT	ICETEXIT	S,M,C
NO	NO	NO	TMAXLIM	Maximum storage above and below 16MB virtual[6]	S,M,C
PARM TRUNC OPTION TRUNC	TRUNC	OPTION TRUNC	TRUNC	DFSORT LRECL truncation action	S,M,C
RECORD TYPE	NO	RECORD TYPE	NO	Record format	S,M,C
PARM VERIFY|NOVERIFY OPTION VERIFY|NOVERIFY	VERIFY|NOVERIFY	OPTION VERIFY|NOVERIFY	VERIFY	Sequence check	S,M
NO	NO	NO	VIO	SORTWK virtual I/O	S
PARM VLLONG|NOVLLONG OPTION VLLONG|NOVLLONG	VLLONG|NOVLLONG	OPTION VLLONG|NOVLLONG	VLLONG	Truncate long output records	S,M,C
PARM VLSCMP|NOVLSCMP OPTION VLSCMP|NOVLSCMP	VLSCMP|NOVLSCMP	OPTION VLSCMP|NOVLSCMP	VLSCMP	Pad short compare fields	S,M,C
PARM VLSHRT|NOVLSHRT OPTION VLSHRT|NOVLSHRT	VLSHRT|NOVLSHRT	OPTION VLSHRT|NOVLSHRT	VLSHRT	Action for short control or compare fields	S,M,C
NO	NO	NO	VSAMBSP	VSAM buffer space	S

Table 66. Directly Invoked DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with EXEC PARM	Specified with SYSIN	Specified with ICEMAC JCL, TSO OR TDx	Description of Option	Function
PARM VSAMEMT|NVSAMEMT OPTION VSAMEMT|NVSAMEMT	VSAMEMT|NVSAMEMT	OPTION VSAMEMT|NVSAMEMT	VSAMEMT	Emty VSAM input	S,M,C
PARM VSAMIO|NOVSAMIO OPTION VSAMIO|NOVSAMIO	VSAMIO|NOVSAMIO	OPTION VSAMIO|NOVSAMIO	VSAMIO	Same VSAM input and output	S
PARM WRKREL|NOWRKREL OPTION WRKREL|NOWRKREL	WRKREL|NOWRKREL	OPTION WRKREL|NOWRKREL	WRKREL	Release SORTWK space	S
PARM WRKSEC|NOWRKSEC OPTION WRKSEC|NOWRKSEC	WRKSEC|NOWRKSEC	OPTION WRKSEC|NOWRKSEC	WRKSEC	SORTWK secondary allocation	S
PARM Y2PAST OPTION Y2PAST SORT|MERGE Y2PAST	Y2PAST	OPTION Y2PAST SORT|MERGE Y2PAST	Y2PAST	Set century window	S,M,C
PARM ZDPRINT|NZDPRINT OPTION ZDPRINT|NZDPRINT	ZDPRINT|NZDPRINT	OPTION ZDPRINT|NZDPRINT	ZDPRINT	ZD SUM results	S,M

Notes to Directly Invoked DFSORT Table

1. Does not request dynamic allocation; only supplies defaults.
2. Not used in SYSIN.
3. All functions do not apply to all exits. See Table 36 on page 283 and Table 37 on page 284 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.
9. Override is at the ddname level.

Program Invoked DFSORT with the Extended Parameter List

Table 67 on page 561 shows where each sort, merge, or copy option may be specified when DFSORT is program invoked and an extended parameter list is passed to it.

DFSPARM: PARM options selectively override corresponding options in any other source. DEBUG and OPTION control statement options selectively override corresponding options in SORTCNTL and the Parameter List. Control statements other than DEBUG and OPTION completely override corresponding control statements in SORTCNTL and the Parameter List.

SORTCNTL: DEBUG and OPTION control statement options selectively override corresponding options in the Parameter List. Control statements other than DEBUG and OPTION completely override corresponding control statements in the Parameter List.

SORT and MERGE are considered to be corresponding control statements.

INCLUDE and OMIT are considered to be corresponding control statements.

Table 67. Extended Parameter List DFSORT Option Specification/Override. Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
NO	NO	NO	ABCODE	ABEND code	S,M,C
DEBUG ABSTP	DEBUG ABSTP	DEBUG ABSTP	NO	Abnormal stop	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 16MB virtual	S,M,C
OPTION ARESINV	OPTION ARESINV	OPTION ARESINV	ARESINV	Storage above 16MB virtual for invoking program	S,M,C
DEBUG NOASSIST	DEBUG NOASSIST	DEBUG NOASSIST	NO	Bypass Sorting Instructions	S
PARM AVGRLEN OPTION AVGRLEN	OPTION AVGRLEN	OPTION AVGRLEN	NO	Average record length	S
PARM BSAM DEBUG BSAM	DEBUG BSAM	DEBUG BSAM	NO	Force BSAM	S,M,C
DEBUG CFW|NOCFW	DEBUG CFW|NOCFW	DEBUG CFW|NOCFW	CFW	Cache fast write	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
PARM COBEXIT OPTION COBEXIT	OPTION COBEXIT	OPTION COBEXIT	COBEXIT	COBOL library	S,M,C
INCLUDE|OMIT COND|FORMAT	INCLUDE|OMIT COND|FORMAT	INCLUDE|OMIT COND|FORMAT	NO	Include|Omit fields	S,M,C
OPTION COPY SORT|MERGE FIELDS	OPTION COPY SORT|MERGE FIELDS[2]	OPTION COPY SORT|MERGE FIELDS	NO	Copy records	C

Table 67. Extended Parameter List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
DEBUG CTRx	DEBUG CTRx	DEBUG CTRx	NO	ABEND record count	S,M
NO	NO	NO	day	Time-of-day for activation	S,M,C
NO	NO	NO	DIAGSIM	Simulate SORTDIAG DD statement	S,M,C
NO	NO	NO	DSA	Dynamic storage adjustment limit	S
PARM DSPSIZE OPTION DSPSIZE	OPTION DSPSIZE	OPTION DSPSIZE	DSPSIZE	Dataspace sorting	S
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC[2]	OPTION DYNALLOC SORT DYNALLOC	DYNALOC[1]	Dynamic SORTWKs	S
PARM DYNALLOC OPTION DYNALLOC|USEWKDD SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	OPTION DYNALLOC|USEWKDD SORT DYNALLOC	DYNAUTO	Automatic DYNALLOC	S
NO	NO	NO	DYNSPC	Dynamic allocation default space	S
PARM EFS OPTION EFS	NO[3]	OPTION EFS	EFS	EFS program specified	S,M,C
NO	NO	NO	ENABLE	Enable Time-of-Day modules	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 67. Extended Parameter List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
DEBUG EQUCOUNT	DEBUG EQUCOUNT	DEBUG EQUCOUNT	NO	Equal key count message	S
PARM ABEND|NOABEND DEBUG ABEND|NOABEND	DEBUG ABEND|NOABEND	DEBUG ABEND|NOABEND	ERET	Error action	S,M,C
DEBUG ESTAE|NOESTAE	DEBUG ESTAE|NOESTAE	DEBUG ESTAE|NOESTAE	ESTAE	ESTAE routine	S,M,C
OPTION EXITCK	OPTION EXITCK	OPTION EXITCK	EXITCK	E15/E35 return code checking	S,M,C
NO	NO	NO	EXPMAX	Available expanded storage limit for all DFSORT Hiperspaces	S
NO	NO	NO	EXPOLD	Old expanded storage limit for all DFSORT Hiperspaces	S
NO	NO	NO	EXPRES	Available expanded storage reserved for non-Hipersorting use	S
PARM E15=COB MODS E15[4]|HILEVEL=YES	MODS E15[4]|HILEVEL=YES	Offset 4 entry[4] MODS E15[4]|HILEVEL=YES	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]|HILEVEL=YES	MODS E35[4]|HILEVEL=YES	Offset 8 entry[4] MODS E35[4]|HILEVEL=YES	NO	Exit E35	S,M,C

Table 67. Extended Parameter List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
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	User Exit Exx (xx=11,16,17,19, 31,37,38, and 61)	S,M,C[5]
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
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
PARM FILSZ OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE[2]	OPTION FILSZ|SIZE SORT|MERGE FILSZ|SIZE	FSZEST	File size	S,M
NO	NO	NO	GENER	IEBGENER name	C
NO	NO	NO	GNPAD	ICEGENER LRECL padding action	C
NO	NO	NO	GNTRUNC	ICEGENER LRECL truncation action	C
PARM HIPRMAX OPTION HIPRMAX	OPTION HIPRMAX	OPTION HIPRMAX	HIPRMAX	Hipersorting	S
NO	NO	NO	IDRCPCT	IDRC compaction	S
NO	NO	NO	IEXIT	ICEIEXIT	S,M,C
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

Table 67. Extended Parameter List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
NO	NO	NO	IOMAXBF	Maximum SORTIN/SORTOUT data set buffer space	S,M,C
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
PARM LOCALE OPTION LOCALE	NO[3]	OPTION LOCALE	LOCALE	Locale processing	S,M,C
NO	NO	NO	MAXLIM	Maximum storage below 16MB 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

Table 67. Extended Parameter List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
PARM ODMAXBF OPTION ODMAXBF	OPTION ODMAXBF	OPTION ODMAXBF	ODMAXBF	Maximum OUTFIL data set buffer space	S,M,C
OUTFIL[11]	OUTFIL[11]	OUTFIL[11]	NO	OUTFIL processing	S,M,C
PARM OUTREL|NOOUTREL OPTION NOOUTREL	OPTION NOOUTREL	OPTION NOOUTREL	OUTREL	Release output data set space	S,M,C
OPTION NOOUTSEC	OPTION NOOUTSEC	OPTION NOOUTSEC	OUTSEC	Output data set secondary allocation	S,M,C
NO	NO	NO	OVERRGN	Storage over REGION	S,M,C
PARM OVFLO OPTION OVFLO	OPTION OVFLO	OPTION OVFLO	OVFLO	Summary fields overflow action	S,M
PARM PAD OPTION PAD	OPTION PAD	OPTION PAD	PAD	DFSORT LRECL padding action	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
PARM RESET|NORESET OPTION RESET|NORESET	OPTION RESET|NORESET	OPTION RESET|NORESET	RESET	NEW or MOD VSAM output	S,M,C
OPTION RESINV	OPTION RESINV	OPTION RESINV	RESINV	Program reserved storage[8]	S,M,C
PARM SDB OPTION SDB	OPTION SDB	OPTION SDB	SDB	System-determined output data set block size	S,M,C

Table 67. Extended Parameter List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
NO	NO	NO	SDBMSG	System-determined block size for message and list data sets	S,M,C
PARM SIZE OPTION MAINSIZE	OPTION MAINSIZE	OPTION MAINSIZE	SIZE	Storage	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 SMF	NO	OPTION SMF	SMF	SMF records	S,M,C
PARM SOLRF|NOSOLRF OPTION SOLRF|NOSOLRF	OPTION SOLRF|NOSOLRF	OPTION SOLRF|NOSOLRF	SOLRF	SORTOUT length	S,M,C
OPTION SORTDD	NO[3]	OPTION SORTDD	NO	ddname prefix	S,M,C
OPTION SORTIN[9]	NO[3]	OPTION SORTIN[9]	NO	Alternate SORTIN ddname	S,C
NO	NO	NO	SORTLIB	Conventional modules library	S,M
OPTION SORTOUT[10]	NO[3]	OPTION SORTOUT[10]	NO	Alternate SORTOUT ddname	S,M,C
PARM SPANINC OPTION SPANINC	OPTION SPANINC	OPTION SPANINC	SPANINC	Incomplete spanned records action	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

Table 67. Extended Parameter List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
NO	NO	NO	SVC	DFSORT SVC information	S,M,C
PARM SZERO|NOSZERO OPTION SZERO|NOSZERO	OPTION SZERO|NOSZERO	OPTION SZERO|NOSZERO	SZERO	Signed or unsigned zero	S,M,C
NO	NO	NO	TEXIT	ICETEXIT	S,M,C
NO	NO	NO	TMAXLIM	Maximum storage above and below 16MB virtual[8]	S,M,C
PARM TRUNC OPTION TRUNC	OPTION TRUNC	OPTION TRUNC	TRUNC	DFSORT LRECL truncation action	S,M,C
RECORD TYPE	RECORD TYPE	RECORD TYPE	NO	Record format	S,M,C
PARM VERIFY|NOVERIFY OPTION VERIFY|NOVERIFY	OPTION VERIFY|NOVERIFY	OPTION VERIFY|NOVERIFY	VERIFY	Sequence check	S,M
NO	NO	NO	VIO	SORTWK virtual I/O	S
PARM VLLONG|NOVLLONG OPTION VLLONG|NOVLLONG	OPTION VLLONG|NOVLLONG	OPTION VLLONG|NOVLLONG	VLLONG	Ttruncate long output records	S,M,C
PARM VLSCMP|NOVLSCMP OPTION VLSCMP|NOVLSCMP	OPTION VLSCMP|NOVLSCMP	OPTION VLSCMP|NOVLSCMP	VLSCMP	Pad short compare fields	S,M,C
PARM VLSHRT|NOVLSHRT OPTION VLSHRT|NOVLSHRT	OPTION VLSHRT|NOVLSHRT	OPTION VLSHRT|NOVLSHRT	VLSHRT	Action for short control field or compare field	S,M,C
NO	NO	NO	VSAMBSP	VSAM buffer space	S
PARM VSAMEMT|NVSAMEMT OPTION VSAMEMT|NVSAMEMT	OPTION VSAMEMT|NVSAMEMT	OPTION VSAMEMT|NVSAMEMT	VSAMEMT	Empty VSAM input	S,M,C

Table 67. Extended Parameter List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with Extended Parameter List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
PARM VSAMIO|NOVSAMIO OPTION VSAMIO|NOVSAMIO	OPTION VSAMIO|NOVSAMIO	OPTION VSAMIO|NOVSAMIO	VSAMIO	Same VSAM input and output	S
PARM WRKREL|NOWRKREL OPTION WRKREL|NOWRKREL	OPTION WRKREL|NOWRKREL	OPTION WRKREL|NOWRKREL	WRKREL	Release SORTWK space	S
PARM WRKSEC|NOWRKSEC OPTION WRKSEC|NOWRKSEC	OPTION WRKSEC|NOWRKSEC	OPTION WRKSEC|NOWRKSEC	WRKSEC	SORTWK secondary allocation	S
PARM Y2PAST OPTION Y2PAST SORT|MERGE Y2PAST	OPTION Y2PAST SORT|MERGE Y2PAST[2]	OPTION Y2PAST SORT|MERGE Y2PAST	Y2PAST	Set century window	S,M,C
PARM ZDPRINT|NZDPRINT OPTION ZDPRINT|NZDPRINT	OPTION ZDPRINT|NZDPRINT	OPTION ZDPRINT|NZDPRINT	ZDPRINT	ZD SUM results	S,M

Notes to Extended Parameter List Table

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 Table 36 on page 283 and Table 37 on page 284 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.
11. Override is at the ddname level.

Program Invoked DFSORT with the 24-Bit Parameter List

Table 68 on page 571 shows where each sort, merge, or copy option may be specified when DFSORT is program invoked and a 24-bit parameter list is passed to it.

DFSPARM: PARM options selectively override corresponding options in any other source. DEBUG and OPTION control statement options selectively override corresponding options in SORTCNTL and the Parameter List. Control statements other than DEBUG and OPTION completely override corresponding control statements in SORTCNTL and the Parameter List.

SORTCNTL: DEBUG control statement options selectively override corresponding options in the Parameter List. Control statements other than DEBUG completely override corresponding control statements in the Parameter List.

SORT and MERGE are considered to be corresponding control statements.

INCLUDE and OMIT are considered to be corresponding control statements.

Table 68. 24-Bit List DFSORT Option Specification/Override. Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
NO	NO	NO	ABCODE	ABEND code	S,M,C
DEBUG ABSTP	DEBUG ABSTP	DEBUG ABSTP	NO	Abnormal stop	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 16MB virtual	S,M,C
OPTION ARESINV	OPTION ARESINV	NO	ARESINV	Storage above 16MB virtual for invoking program	S,M,C
DEBUG NOASSIST	DEBUG NOASSIST	DEBUG NOASSIST	NO	Bypass Sorting Instructions	S
PARM AVGRLEN OPTION AVGRLEN	OPTION AVGRLEN	NO	NO	Average record length	S
PARM BSAM DEBUG BSAM	DEBUG BSAM	DEBUG BSAM	NO	Force BSAM	S,M,C
DEBUG CFW|NOCFW	DEBUG CFW|NOCFW	DEBUG CFW|NOCFW	CFW	Cache fast write	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
PARM COBEXIT OPTION COBEXIT	OPTION COBEXIT	NO	COBEXIT	COBOL library	S,M,C
INCLUDE|OMIT COND|FORMAT	INCLUDE|OMIT COND|FORMAT	INCLUDE|OMIT COND|FORMAT	NO	Include|Omit fields	S,M,C
OPTION COPY SORT|MERGE FIELDS	OPTION COPY SORT|MERGE FIELDS	SORT|MERGE FIELDS	NO	Copy records	C

Table 68. 24-Bit List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
DEBUG CTRx	DEBUG CTRx	DEBUG CTRx	NO	ABEND record count	S,M
NO	NO	NO	day	Time-of-day for activation	S,M,C
NO	NO	NO	DIAGSIM	Simulate SORTDIAG DD statement	S,M,C
NO	NO	NO	DSA	Dynamic storage adjustment limit	S
PARM DSPSIZE OPTION DSPSIZE	OPTION DSPSIZE	NO	DSPSIZE	Dataspace sorting	S
PARM DYNALLOC OPTION DYNALLOC SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	SORT DYNALLOC	DYNALOC[1]	Dynamic SORTWKs	S
PARM DYNALLOC OPTION DYNALLOC|USEWKDD SORT DYNALLOC	OPTION DYNALLOC SORT DYNALLOC	SORT DYNALLOC	DYNAUTO	Automatic DYNALLOC	S
NO	NO	NO	DYNSPC	Dynamic allocation default space	S
PARM EFS OPTION EFS	NO[2]	NO	EFS	EFS program specified	S,M,C
NO	NO	NO	ENABLE	Enable Time-of-Day modules	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
DEBUG EQUCOUNT	DEBUG EQUCOUNT	DEBUG EQUCOUNT	NO	Equal key count message	S

Table 68. 24-Bit List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
PARM ABEND|NOABEND DEBUG ABEND|NOABEND	DEBUG ABEND|NOABEND	DEBUG ABEND|NOABEND	ERET	Error action	S,M,C
DEBUG ESTAE|NOESTAE	DEBUG ESTAE|NOESTAE	DEBUG ESTAE|NOESTAE	ESTAE	ESTAE routine	S,M,C
OPTION EXITCK	OPTION EXITCK	NO	EXITCK	E15/E35 return code checking	S,M,C
NO	NO	NO	EXPMAX	Available expanded storage limit for all DFSORT Hiperspaces	S
NO	NO	NO	EXPOLD	Old expanded storage limit for all DFSORT Hiperspaces	S
NO	NO	NO	EXPRES	Available expanded storage reserved for non-Hipersorting use	S
PARM E15=COB MODS E15[3]|HILEVEL=YES	MODS E15[3]|HILEVEL=YES	Offset 18 entry[3] MODS E15[3]|HILEVEL=YES	NO	User exit E15	S,C
NO	NO	Offset 18 entry	NO	User exit E32	M
PARM E35=COB MODS E35[3]|HILEVEL=YES	MODS E35[3]|HILEVEL=YES	Offset 22 entry[3] MODS E35[3]|HILEVEL=YES	NO	User exit E35	S,M,C
MODS Exx	MODS Exx	MODS Exx	NO	User Exit Exx (xx=11,16-19, 31,37-39, and 61)	S,M,C[4]
INREC FIELDS	INREC FIELDS	INREC FIELDS	NO	INREC fields	S,M,C
OUTREC FIELDS	OUTREC FIELDS	OUTREC FIELDS	NO	OUTREC fields	S,M,C

Table 68. 24-Bit List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
SORT|MERGE FIELDS|FORMAT	SORT|MERGE FIELDS|FORMAT	SORT|MERGE FIELDS|FORMAT	NO	Control fields	S,M,C
SUM FIELDS|FORMAT	SUM FIELDS|FORMAT	SUM FIELDS|FORMAT	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	FSZEST	File size	S,M
NO	NO	NO	GENER	IEBGENER name	C
NO	NO	NO	GNPAD	ICEGENER LRECL padding action	C
NO	NO	NO	GNTRUNC	ICEGENER LRECL truncation action	C
PARM HIPRMAX OPTION HIPRMAX	OPTION HIPRMAX	NO	HIPRMAX	Hipersorting	S
NO	NO	NO	IDRCPCT	IDRC compaction	S
NO	NO	NO	IEXIT	ICEIEXIT	S,M,C
OPTION CKPT[5] SORT|MERGE CKPT[5]	OPTION CKPT[5] SORT|MERGE CKPT[5]	SORT|MERGE CKPT[5]	IGNCKPT	Checkpoints	S
NO	NO	NO	IOMAXBF	Maximum SORTIN/SORTOUT data set buffer space	S,M,C
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

Table 68. 24-Bit List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
PARM LISTX|NOLISTX OPTION LISTX|NOLISTX	NO[2]	NO	LISTX	Print control statements returned by an EFS program[6]	S,M,C
PARM LOCALE OPTION LOCALE	NO[2]	NO	LOCALE	Locale processing	S,M,C
NO	NO	NO	MAXLIM	Maximum storage below 16MB 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 ODMAXBF OPTION ODMAXBF	OPTION ODMAXBF	NO	ODMAXBF	Maximum OUTFIL data set buffer space	S,M,C
OUTFIL[10]	OUTFIL[10]	OUTFIL[10]	NO	OUTFIL processing	S,M,C
PARM OUTREL|NOOUTREL OPTION NOOUTREL	OPTION NOOUTREL	NO	OUTREL	Release output data set space	S,M,C
OPTION NOOUTSEC	OPTION NOOUTSEC	NO	OUTSEC	Output data set secondary allocation	S,M,C

Table 68. 24-Bit List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
NO	NO	NO	OVERRGN	Storage over REGION	S,M,C
PARM OVFLO OPTION OVFLO	OPTION OVFLO	NO	OVFLO	Summary fields overflow action	S,M
PARM PAD OPTION PAD	OPTION PAD	NO	PAD	DFSORT LRECL padding action	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
PARM RESET|NORESET OPTION RESET|NORESET	OPTION RESET|NORESET	NO	RESET	NEW or MOD VSAM output	S,M,C
OPTION RESINV	OPTION RESINV	X'01' entry	RESINV	Program reserved storage[7]	S,M,C
PARM SDB OPTION SDB	OPTION SDB	NO	SDB	System-determined output data set block size	S,M,C
NO	NO	NO	SDBMSG	System-determined block size for message and list data sets	S,M,C
PARM SIZE OPTION MAINSIZE	OPTION MAINSIZE	X'00' entry	SIZE	Storage	S,M,C
PARM SKIPREC OPTION SKIPREC SORT|MERGE SKIPREC	OPTION SKIPREC SORT|MERGE SKIPREC	SORT|MERGE SKIPREC	NO	Skip records	S,C
OPTION SMF	NO	NO	SMF	SMF records	S,M,C

Table 68. 24-Bit List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
PARM SOLRF|NOSOLRF OPTION SOLRF|NOSOLRF	OPTION SOLRF|NOSOLRF	NO	SOLRF	SORTOUT length	S,M,C
OPTION SORTDD	NO[2]	Prefix entry	NO	ddname prefix	S,M,C
OPTION SORTIN[8]	NO[2]	NO	NO	Alternate SORTIN ddname	S,C
NO	NO	NO	SORTLIB	Conventional modules library	S,M
OPTION SORTOUT[9]	NO[2]	NO	NO	Alternate SORTOUT ddname	S,M,C
PARM SPANINC OPTION SPANINC	OPTION SPANINC	NO	SPANINC	Incomplete spanned records action	S,M,C
PARM STIMER|NOSTIMER OPTION NOSTIMER	OPTION NOSTIMER	NO	STIMER	Use of STIMER	S,M,C
PARM STOPAFT OPTION SORT OPTION STOPAFT SORT|MERGE STOPAFT	OPTION STOPAFT SORT|MERGE STOPAFT	SORT|MERGE STOPAFT	NO	Input limit	S,C
NO	NO	NO	SVC	DFSORT SVC information	S,M,C
PARM SZERO|NOSZERO OPTION SZERO|NOSZERO	OPTION SZERO|NOSZERO	NO	SZERO	Signed or unsigned zero	S,M,C
NO	NO	NO	TEXIT	ICETEXIT	S,M,C
NO	NO	NO	TMAXLIM	Maximum storage above and below 16MB virtual[7]	S,M,C

Table 68. 24-Bit List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
PARM TRUNC OPTION TRUNC	OPTION TRUNC	NO	TRUNC	DFSORT LRECL truncation action	S,M,C
RECORD TYPE	RECORD TYPE	RECORD TYPE	NO	Record format	S,M,C
PARM VERIFY|NOVERIFY OPTION VERIFY|NOVERIFY	OPTION VERIFY|NOVERIFY	NO	VERIFY	Sequence check	S,M
NO	NO	NO	VIO	SORTWK virtual I/O	S
PARM VLLONG|NOVLLONG OPTION VLLONG|NOVLLONG	OPTION VLLONG|NOVLLONG	NO	VLLONG	Trucate long output records	S,M,C
PARM VLSCMP|NOVLSCMP OPTION VLSCMP|NOVLSCMP	OPTION VLSCMP|NOVLSCMP	NO	VLSCMP	Pad short compare fields	S,M,C
PARM VLSHRT|NOVLSHRT OPTION VLSHRT|NOVLSHRT	OPTION VLSHRT|NOVLSHRT	NO	VLSHRT	Action for short control or compare field	S,M,C
NO	NO	NO	VSAMBSP	VSAM buffer space	S
PARM VSAMEMT|NVSAMEMT OPTION VSAMEMT|NVSAMEMT	OPTION VSAMEMT|NVSAMEMT	NO	VSAMEMT	Empty VSAM input	S,M,C
PARM VSAMIO|NOVSAMIO OPTION VSAMIO|NOVSAMIO	OPTION VSAMIO|NOVSAMIO	NO	VSAMIO	Same VSAM input and output	S
PARM WRKREL|NOWRKREL OPTION WRKREL|NOWRKREL	OPTION WRKREL|NOWRKREL	NO	WRKREL	Release SORTWK space	S
PARM WRKSEC|NOWRKSEC OPTION WRKSEC|NOWRKSEC	OPTION WRKSEC|NOWRKSEC	NO	WRKSEC	SORTWK secondary allocation	S
PARM Y2PAST OPTION Y2PAST SORT|MERGE Y2PAST	OPTION Y2PAST SORT|MERGE Y2PAST	SORT|MERGE Y2PAST	Y2PAST	Set century window	S,M,C

Table 68. 24-Bit List DFSORT Option Specification/Override (continued). Options are arranged alphabetically on the ICEMAC column. If "NO" is specified in the ICEMAC column, move to the next column to the left and so on.

The order of override is from left to right and from top to bottom within a row.

Specified with DFSPARM	Specified with SORTCNTL	Specified with 24-Bit List	Specified with ICEMAC INV, TSOINV or TDx	Description of Option	Function
PARM ZDPRINT|NZDPRINT OPTION ZDPRINT|NZDPRINT	OPTION ZDPRINT|NZDPRINT	NO	ZDPRINT	ZD SUM results	S,M

Notes to 24-Bit List Table

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 user exit is specified in a MODS statement.
4. All functions do not apply to all user exits. See Table 36 on page 283 and Table 37 on page 284 for applicable user 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.
10. Override is at the ddname level.

Appendix C. Data Format Descriptions

The format descriptions refer to the assembled data formats as used with IBM System/390. 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 **Notes:** 1. If CHALT is in effect, a format CH field collates according to the ALTSEQ (alternate collating sequence) table in effect. AQ format can be used for the same purpose. 2. If locale processing is in effect, a format CH field collates according to the collating rules of the active locale.
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 **Notes:** 1. The following are treated as positive sign indicators: F, E, C, A, 8, 6, 4, 2, 0. 2. The following are treated as negative sign indicators: D, B, 9, 7, 5, 3, 1. 3. For SUM processing, 0 through 9 for the sign or A through F for a digit results in a data exception (0C7 ABEND). For example, a ZD value such as 3.5 (X'F34BF5') results in an 0C7 because B is treated as an invalid digit. ICETOOL's DISPLAY or VERIFY operator can be used to identify ZD values with invalid digits. ICETOOL's VERIFY operator can be used to identify ZD values with invalid signs. 4. The first four bits of the last digit is the sign indicator. The first four bits of each other digit is ignored. Thus the EBCDIC strings '0025' and ' 25' are both treated as 25 because a leading blank (X'40') is equivalent to a 0 digit (X'F0').

© Copyright IBM Corp. 1973, 2000

Data Format Examples

Format	Description
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 2 4 7D Hexadecimal 00100100 01111101 Binary The number +247 becomes 247C in hexadecimal. **Notes:** 1. The following are treated as positive sign indicators: F, E, C, A, 8, 6, 4, 2, 0. 2. The following are treated as negative sign indicators: D, B, 9, 7, 5, 3, 1. 3. For SUM processing, 0 through 9 for the sign or A through F for a digit results in a data exception (0C7 ABEND). For example, a PD value such as X'0123BF' results in an 0C7 because B is treated as an invalid digit. ICETOOL's DISPLAY or VERIFY operator can be used to identify PD values with invalid digits. ICETOOL's VERIFY operator can be used to identify PD values with invalid signs.
PD0	(packed decimal, with sign and first digit ignored) The PD0 format can be represented as follows: `xddd...ds` x is hexadecimal 0-F and is ignored. d is hexadecimal 0-9 and represents a decimal digit. s is hexadecimal 0-F and is ignored. PD0 can be used for parts of PD fields. For example, in the PD field P'mmddyy' (hexadecimal 0mmddyyC), PD0 can be used separately for 0mmd (mm), mddy (dd) and dyyC (yy).
FI	(fixed point, signed). The complete number is represented by its binary equivalent with the sign indicator placed in the most significant bit position. 0 for + or 1 for -. Negative numbers are in 2's complement form. Example: +247 becomes in halfword form 00F7 Hexadecimal 0000000011110111 Binary The number -247 becomes FF09 Hexadecimal
BI	(binary unsigned). Any bit pattern.
FL	(floating point, signed). The specified number is in the two-part format of characteristic 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)

Format	Description
CSF or FS	(signed numeric with optional leading floating sign). The floating sign format can be represented as follows: <s>d . . .d s is an optional sign immediately to the left of the digits d . . .d. If s is a −, the number is treated as negative, otherwise it is treated as positive. Thus, − must be used for a minus sign, but any other character (for example, + or blank) can be used for a plus sign. The first non-decimal digit (that is, not 0-9) going from right to left is treated as the sign and anything to the left of the sign is ignored. Examples: Value: Treated as: 34 +34 +34 +34 00034 +34 -003 -3 -1234 -1234 1234 +1234 +01234 +1234 0 +0 The types of data handled by the CSF/FS format encompass those produced by several different FORTRAN, PL/I and COBOL formats, such as those shown below (using a width of 4 for purposes of illustration): * FORTRAN: I4 ; G4.0 ; SP,I4 ; SP,I4.3 ; S,I4.3 * PL/I: F(4) ; P'S999' ; P'SSS9' ; P'---9' * COBOL: PIC +++9 ; PIC +999 ; PIC ++++ ; PIC ---9 ; PIC ---- ; PIC ZZZZ Because CSF/FS format fields are processed less efficiently than the other formats, CSF/FS should not be used when another format is also appropriate (for example, CSL).
CSL or LS	(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 or TS	(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] or OL[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

Data Format Examples

Format	Description
CTO or OT	(signed numeric, trailing overpunch sign). This format has the same representation as for the CLO format, except that the sign indicator is overpunched on the last decimal digit of the number. Example: +247 with + overpunched on 7 becomes F2 F4 C7 hexadecimal
ASL	(signed numeric, ASCII, leading separate sign). Similar to the CSL format but with decimal data assembled into ASCII code. Example: +247 punched into card becomes + 2 4 7 Punched numeric data 2B 32 34 37 Hexadecimal 0101011 00110010 00110100 00110111 Binary ASCII code Similarly -247 becomes 2D 32 34 37 hexadecimal
AST	(signed numeric, ASCII, trailing separate sign). This gives the same bit representation as the ASL format, except that the sign is punched after the number. Example: 247+ becomes 32 34 37 2B hexadecimal
Y2T	(character or zoned decimal yyx, yyxx, yyxxx and yyxxxx full date format with special indicators). The date field can be represented as follows: 3,Y2T: C'yyx' or Z'yyx' 4,Y2T: C'yyxx' or Z'yyxx' 5,Y2T: C'yyxxx' or Z'yyxxx' 6,Y2T: C'yyxxxx' or Z'yyxxxx' y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-9 and represents a non-year digit. x...x must be in correct collating order. The special indicators are X'00...00' (BI zeros), X'40...40' (blanks), C'0...0' (CH zeros), Z'0...0' (ZD zeros), C'9...9' (CH nines), Z'9...9' (ZD nines) and X'FF...FF' (BI ones).
Y2U	(packed decimal yyx and yyxxx full date format with special indicators). The date field can be represented as follows: 2,Y2U: P'yyx' (X'yyxs') 3,Y2U: P'yyxxx' (X'yyxxxs') y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-9 and represents a non-year digit. s is hexadecimal 0-F and is ignored. xxx must be in correct collating order. The special indicators are P'0...0' (PD zeros) and P'9...9' (PD nines).
Y2V	(packed decimal yyxx and yyxxxx full date format with special indicators). The date field can be represented as follows: 3,Y2V: P'yyxx' (X'0yyxxs') 4,Y2V: P'yyxxxx' (X'0yyxxxxs') y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-9 and represents a non-year digit. s is hexadecimal 0-F and is ignored. xx or xxxx must be in correct collating order. The special indicators are P'0...0' (PD zeros) and P'9...9' (PD nines).

Format	Description
Y2W	(character or zoned decimal xyy, xxyy, xxxyy and xxxxyy full date format with special indicators). The date field can be represented as follows: 3,Y2W: C'xyy' or Z'xyy' 4,Y2W: C'xxyy' or Z'xxyy' 5,Y2W: C'xxxyy' or Z'xxxyy' 6,Y2W: C'xxxxyy' or Z'xxxxyy' y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-9 and represents a non-year digit. x...x must be in correct collating order. x...xyy will be treated as yyx...x when collating the date field. The special indicators are X'00...00' (BI zeros), X'40...40' (blanks), C'0...0' (CH zeros), Z'0...0' (ZD zeros), C'9...9' (CH nines), Z'9...9' (ZD nines) and X'FF...FF' (BI ones).
Y2X	(packed decimal xyy and xxxyy full date format with special indicators). The date field can be represented as follows: 2,Y2X: P'xyy' (X'xyys') 3,Y2X: P'xxxyy' (X'xxxyys') y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-9 and represents a non-year digit. s is hexadecimal 0-F and is ignored. xxx must be in correct collating order. x...xyy will be treated as yyx...x when collating the date field. The special indicators are P'0...0' (PD zeros) and P'9...9' (PD nines).
Y2Y	(packed decimal xxyy and xxxxyy full date format with special indicators). The date field can be represented as follows: 3,Y2Y: P'xxyy' (X'0xxyys') 4,Y2Y: P'xxxxyy' (X'0xxxxyys') y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-9 and represents a non-year digit. s is hexadecimal 0-F and is ignored. xx or xxxx must be in correct collating order. x...xyy will be treated as yyx...x when collating the date field. The special indicators are P'0...0' (PD zeros) and P'9...9' (PD nines).
Y2C or Y2Z	(two-digit, two-byte character or zoned-decimal year data). The two-digit year data can be represented as follows: xyxy y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-F and is ignored. Thus, 96 might be represented as hexadecimal F9F6 (character 96) or as hexadecimal F9C6 or 0906 (zoned decimal 96).
Y2P	(two-digit, two-byte packed-decimal year data). The two-digit year data can be represented as follows: xyyx y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-F and is ignored. Thus, 96 might be represented as hexadecimal 096F or 896C (packed decimal 96).
Y2D	(two-digit, one-byte decimal year data). The two-digit year data can represented as follows: yy y is hexadecimal 0-9 and represents a year digit. Thus, 96 would be represented as hexadecimal 96 (decimal 96).

Data Format Examples

Format	Description
Y2S	(two-digit, two-byte character or zoned-decimal year data with special indicators). The two-digit year data can represented as follows: xyxy y is hexadecimal 0-9 and represents a year digit. x is hexadecimal 0-F and is ignored. Thus, 96 might be represented as hexadecimal F9F6 (character 96) or as hexadecimal F9C6 or 0906 (zoned decimal 96). The special indicators can be represented as follows: qxzx qx is hexadecimal 00, 40 or FF. zx is hexadecimal 00-FF (although typically 00, 40 and FF). Thus, special indicators might be hexadecimal 0000, 0005, 4040, FFFF, FF85 and so on.
Y2B	(two-digit, one-byte binary year data). The binary year data can be represented as follows: hh hh is the hexadecimal equivalent of a decimal yy value as follows: Binary Values — Decimal Values — yy X'00'-X'63' — 00-99 — 00-99 X'64'-X'C7' — 100-199 — 00-99 X'C8'-X'FF — 200-255 — 00-55 Thus, 96 might be represented as hexadecimal 60 (decimal 96) or C4 (decimal 196).

[1] The overpunch sign bit is always 'C' for positive and 'D' for negative.

A detailed description of CH, ZD, PD, FI, BI, and FL data formats are found in *Assembler Reference*.

The following tables show the statements, operands, and operators allowed with each of the various data formats.

Table 69. Allowed with Frequently Used Data Types

Statement, Operand, or Operator	CH	BI	FI	PD	ZD	FS or CSF
DFSORT statements						
INCLUDE	X	X	X	X	X	X
INREC		X	X	X	X	X
MERGE	X	X	X	X	X	X
OMIT	X	X	X	X	X	X
OUTREC		X	X	X	X	X
SORT	X	X	X	X	X	X
SUM		X	X	X	X	
OUTFIL statement operands						
INCLUDE	X	X	X	X	X	X
OMIT	X	X	X	X	X	X
OUTREC		X	X	X	X	X
TRAILERx		X	X	X	X	X
ICETOOL operators						
DISPLAY (ON, BREAK)	X	X	X	X	X	X
OCCUR (ON)	X	X	X	X	X	X
RANGE (ON)		X	X	X	X	X
SELECT (ON)	X	X	X	X	X	X
STATS (ON)		X	X	X	X	X
UNIQUE (ON)		X	X	X	X	X

Table 69. Allowed with Frequently Used Data Types (continued)

Statement, Operand, or Operator	CH	BI	FI	PD	ZD	FS or CSF
VERIFY (ON)				X	X	

Table 70. Allowed with Other Data Types

Statement or Operand	AQ	AC	FL	LS or CSL	TS or CST	OL or CLO	OT or CTO	ASL	AST	D1	D2	PD0	Y2x
DFSORT statements													
INCLUDE	X	X		X	X	X	X	X	X		X		X
INREC												X	X
MERGE	X	X	X	X	X	X	X	X	X	X		X	X
OMIT	X	X		X	X	X	X	X	X		X		X
OUTREC												X	X
SORT	X	X	X	X	X	X	X	X	X	X		X	X
SUM			X										
OUTFIL statement operands													
INCLUDE	X	X		X	X	X	X	X	X				X
OMIT	X	X		X	X	X	X	X	X				X
OUTREC												X	X

Appendix D. EBCDIC and ISCII/ASCII Collating Sequences

EBCDIC

Table 71 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.

ALTSEQ, CHALT, and LOCALE can be used to select alternate collating sequences for character data.

Packed decimal, zoned decimal, fixed-point, and normalized floating-point data are collated algebraically, that is, each quantity is interpreted as having a sign.

Table 71. EBCDIC Collating Sequence

Collating Sequence	Bit Configuration	Symbol	Meaning
0	00000000		
.			
.			
64	01100100	SP	Space
.			
.			
74	01001010	θ	Cent sign
75	01001011	.	Period, decimal point
76	01001100	<	Less than sign
77	01001101	(	Left parenthesis
78	01001110	+	Plus sign
79	01001111	I	Vertical bar, Logical OR
80	01010000	&	Ampersand
.			
.			
90	01011010	!	Exclamation point
91	01011011	$	Dollar sign
92	01011100	*	Asterisk
93	01011101	)	Right parenthesis
94	01011110	;	Semicolon
95	01011111		Logical not
96	01100000	—	Minus, hyphen
97	01100001	/	Slash
107	01101011	,	Comma
108	01101100	%	Percent sign
109	01101101	_	Underscore
110	01101110	>	Greater than sign
111	01101111	?	Question mark
.			
.			
122	01111010	:	Colon
123	01111011	#	Number sign
124	01111100	@	Commercial At
125	01111101	’	Apostrophe, prime
126	01111110	=	Equal sign
127	01111111	″	Quotation marks
.			

© Copyright IBM Corp. 1973, 2000

Table 71. EBCDIC Collating Sequence (continued)

Collating Sequence	Bit Configuration	Symbol	Meaning
.			
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	0	
151	10010111	p	
152	10011000	q	
153	10011001	r	
.			
.			
162	10100010	s	
163	10100011	t	
164	10100100	u	
165	10100101	v	
166	10100110	w	
167	10100111	x	
168	10101000	y	
169	10101001	z	
193	11000001	A	
194	11000010	B	
195	11000011	C	
196	11000100	D	
197	11000101	E	
198	11000110	F	
199	11000111	G	
200	11001000	H	
201	11001001	I	
.			
.			
209	11010001	J	
210	11010010	K	
211	11010011	L	
212	11010100	M	
213	11010101	N	
214	11010110	O	
215	11010111	P	
216	11011000	Q	
217	11011001	R	
.			
.			

Table 71. EBCDIC Collating Sequence (continued)

Collating Sequence	Bit Configuration	Symbol	Meaning
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		

ISCII/ASCII

Table 72 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.

Table 72. ISCII/ASCII Collating Sequence

Collating Sequence	Bit 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

Table 72. ISCII/ASCII Collating Sequence (continued)

Collating Sequence	Bit Configuration	Symbol	Meaning
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	001101100	6	
55	00110111	7	
56	00111000	8	
57	00111001	9	
58	00111010	:	Colon
59	00111011	;	Semicolon
60	00111100	<	Less than
61	00111101	=	Equals
62	00111110	>	Greater than
63	00111111	?	Question mark
64	01000000	@	Commercial At
65	01000001	A	
66	01000010	B	
67	01000011	C	
68	01000100	D	
69	01000101	E	
70	01000110	F	
71	01000111	G	
72	01001000	H	
73	01001001	I	
74	01001010	J	
75	01001011	K	
76	01001100	L	
77	01001101	M	
78	01001110	N	
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	ˆ	Circumflex, Logical NOT
95	01011111	_	Underscore
96	01100000	′	Grave Accent

Table 72. ISCII/ASCII Collating Sequence (continued)

Collating Sequence	Bit Configuration	Symbol	Meaning
97	01100001	a	
98	01100010	b	
99	01100011	c	
100	01100100	d	
101	01100101	e	
102	01100110	f	
103	01100111	g	
104	01101000	h	
105	01101001	i	
106	01101010	j	
107	01101011	k	
108	01101100	l	
109	01101101	m	
110	01101110	n	
111	01101111	o	
112	01110000	p	
113	01110001	q	
114	01110010	r	
115	01110011	s	
116	01110100	t	
117	01110101	u	
118	01110110	v	
119	01110111	w	
120	01111000	x	
121	01111001	y	
122	01111010	z	
123	01111011	{	Opening Brace
124	01111100	|	Vertical Line
125	01111101	}	Closing Brace
126	01111110	~	Tilde

Appendix E. DFSORT Abend Processing

This appendix explains how DFSORT processes an abend. It is intended to help you get the dump you need 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 application. 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.

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, or Major Calls 4 and 5 are made to the EFS program before the application terminates processing. 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 allows information about an application to be recorded so that same application can be restarted after abnormal termination or after some portion of the application has been completed. Restart can take place immediately or be deferred until the application is resubmitted.

DFSORT takes checkpoints when requested during a sort that uses the Peerage or Vale techniques.

To have DFSORT record checkpoints you must code a SORTCKPT DD statement and ensure the Peerage or Vale technique is selected. See “SORTCKPT DD Statement” on page 73 and “OPTION Control Statement” on page 134 for more information on the SORTCKPT and CKPT options, respectively.

In general, no checkpoints are taken if the following conditions exist:

- No work data set is specified.
- The application is a copy or merge.
- Blockset is selected.

© Copyright IBM Corp. 1973, 2000

Notes:

1. No ANSI Standard Label tape files can be open during Checkpoint/Restart processing.
2. Do not specify CHKPT=EOV on any DFSORT DD statement.

For more information on the Checkpoint/Restart facility, see *Checkpoint/Restart*.

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 *Messages, Codes and Diagnosis* 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 *Messages, Codes and Diagnosis* for complete information about user abends issued by DFSORT.

Note: See “SmartBatch Pipe Considerations” on page 15 for information about special user abend processing in conjunction with SmartBatch pipe data sets.

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 runs any of the functions specified above if they have not already been run 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 *Messages, Codes and Diagnosis*
- Snap dumps

The DFSORT recovery routine provides a snap dump of 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 SYSMDUMP) DD statement is included in the application.

- 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 5. Invoking DFSORT from a Program” on page 331 for more information.
- Continuation of an application after successful SORTOUT output

  If an unexpected abend occurs after the sort, merge, or copy application writes the SORTOUT data set successfully, DFSORT issues message ICE186A and completes its normal cleanup and termination functions. The SORTOUT data set written by DFSORT is closed. The run is successful except for the function causing the abend. Message ICE186A says that the SORTOUT data set is usable even though the run has abended. You can then decide to use the SORTOUT data set or rerun the application.
- 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 runs 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 run. As a result, the dump produced might 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 77. 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 Processing

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.

Appendix F. Locales Supplied with C/370

The following table lists the locales supported by the C/370 product. All of these locale files are provided with OS/390 Release 1 (or higher) or MVS/ESA 5.2 (or higher) with Language Environment for MVS & VM 1.5.0 (or higher). Consult your system programmer to determine whether they have been installed at your site.

For details about locales and their customization, see *Using Locales.* The table of supported locales has been reproduced here for your convenience.

Table 73. Supported locale names

Locale Name	Language	Country
C		
DA_DK	Danish	Denmark
DE_CH	German	Switzerland
DE_DE	German	Germany
EL_GR	Greek	Greece
EN_GB	English	United Kingdom
EN_JP	English	Japan
EN_US	English	United States
ES_ES	Spanish	Spain
FI_FI	Finnish	Finland
FR_BE	French	Belgium
FR_CA	French	Canada
FR_CH	French	Switzerland
FR_FR	French	France
IS_IS	Icelandic	Iceland
IT_IT	Italian	Italy
JA_JP	Japanese	Japan
NL_BE	Dutch	Belgium
NL_NL	Dutch	Netherlands
NO_NO	Norwegian	Norway
PT_PT	Portuguese	Portugal
SV_SE	Swedish	Sweden
TR_TR	Turkish	Turkey

© Copyright IBM Corp. 1973, 2000

Appendix G. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION"AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Information Enabling Requests
Dept. DZWA

© Copyright IBM Corp. 1973, 2000

5600 Cottle Road
San Jose, CA 95193 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Programming Interface Information

This book documents intended Programming Interfaces that allow the customer to write programs to obtain services of DFSORT.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AD/Cycle	ESCON	OS/390
C/370	IBM	RACF
COBOL/370	Hipersorting	RAMAC
DFSMS/MVS	Hiperspace	SmartBatch
DFSMSrmm	Language Environment	System/390
DFSORT	MVS/ESA	VM/ESA
ES/9000		VM/XA
ESA/370		3090

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of other companies.

Summary of Changes

New Programming Support for Release 13 (PTFs after April, 1996)

Additional Year 2000 Features

A new Y2S format can order and transform two-digit character or zoned decimal year data according to the century window, while handling binary zeros, blanks and binary ones in the year field as special indicators.

A new Y2B format can order and transform two-digit binary year data according to the century window.

FREE=CLOSE support for DFSPARM makes it possible to override the SORT statements generated by multiple COBOL SORT verbs in the same COBOL program.

OS/390 Registration

With OS/390 R2 and above, a check is performed to ensure that the DFSORT product is licensed for use, either as a feature of OS/390 or as a separate program product.

New Programming Support for Release 13 (PTFs - April, 1996)

Year 2000 Features

New Y2C, Y2Z, Y2P and Y2D formats, in conjunction with a new Y2PAST installation and run-time option, allow you to handle two-digit year data in the following ways:

- Set the appropriate century window for your applications (for example, 1915-2014 or 1950-2049).
- Order two-digit character, zoned decimal, packed decimal or decimal year data according to the century window using Blockset SORT or MERGE (for example, order 96 representing 1996 before 00 representing 2000 in ascending sequence, or order 00 before 96 in descending sequence).
- Transform two-digit character, zoned decimal, packed decimal or decimal year data to four-digit character year data according to the century window using OUTFIL OUTREC (for example, transform 96 to 1996 and 00 to 2000).

A new PD0 format allows you to order and transform parts of packed decimal fields (for example, month and day in date fields) using SORT, MERGE and OUTFIL.

Performance Improvements for FLR and VLR Blockset Sorts

Performance improvements for FLR and VLR Blockset sorts include the following:

- Dataspace sorting can now be used for variable-length record sort applications.
- DFSORT data processing methods have been improved.
- Dynamic storage adjustment is a new feature that allows DFSORT to automatically use more storage than the TMAXLIM value for a Blockset sort application if DFSORT determines that doing so should improve performance. New installation option DSA=n has been added to enable you to specify the dynamic storage adjustment limit.

© Copyright IBM Corp. 1973, 2000

- The upper limit for the amount of main storage that can be specified and used by DFSORT has been raised from 32M to 2000M. Specifying more main storage can provide the following benefits:
  - It allows DFSORT to sort very large data sets more efficiently.
  - It allows more sort applications to be done entirely in main storage, eliminating the need for intermediate work space and greatly reducing the EXCP counts for those applications.
  - It increases the maximum amount of data DFSORT can process in a single sort application.
- New installation option IOMAXBF=n has been added to enable you to specify the upper limit for the amount of storage to be used for SORTIN and SORTOUT data set buffers, which in turn limits the amount of data that can be transferred in a single I/O operation.
- The upper limit for the number of JCL and dynamically allocated work data sets that can be specified and used by DFSORT's Blockset technique has been raised from 32 to 100. The use of more work data sets increases the maximum amount of data DFSORT can process in a single sort application.
- Changes to the DFSORT SVC provide caching selection enhancements that improve storage control caching performance, especially for SORTIN and SORTOUT devices.
- DFSORT can now use NOEQUALS for VLR Blockset applications if EQUALS=NO is specified at installation or NOEQUALS is specified at run-time. The use of NOEQUALS can improve performance and is recommended for applications for which the order of records that collate identically need not be preserved from input to output. To minimize migration concerns, the IBM-supplied default for the ICEMAC EQUALS option is the new value VLBLKSET, which is equivalent to EQUALS=YES for VLR Blockset applications and to EQUALS=NO for all other applications.

Floating Point for SUM

FL format can now be used with the SUM control statement for short (4-byte), long (8-byte) and extended (16-byte) floating point data.

Security Improvements

Changes to the DFSORT SVC provide security improvements that bring DFSORT up to B1 security standards.

EXCPVR Processing Removed

To enhance DFSORT's protection of system integrity, EXCPVR processing will no longer be used. EXCPVR parameter values will continue to be accepted, but will have no effect on DFSORT processing. In general, the performance improvements provided by EXCPVR processing have diminished with newer technologies and will be more than offset by the performance improvements listed above. Please ignore any references to EXCPVR in this book; all such references will be deleted when the book is updated.

New Device Support for Release 13 (PTFs)

The IBM 3590 Magnetic Tape Subsystem is supported for input, output and work data sets.

New Programming Support for Release 13

DFSORT's Performance Booster for The SAS** System

DFSORT Release 13 provides significant CPU time improvements for SAS applications. To take advantage of this new feature, contact SAS Institute Inc. for details of the support they provide to enable this enhancement.

Dynamic Hipersorting

Dynamic Hipersorting is a new, automatic feature that eliminates the unintended system paging activity and expanded storage and paging data set space shortages that sometimes resulted from a large amount of Hipersorting activity, especially from multiple concurrent Hipersorting applications.

Dynamic Hipersorting allows for more optimal DFSORT and system performance and provides installation options that allow you to customize HIPRMAX=OPTIMAL to your own criteria. With the advent of this feature, we recommend that you use HIPRMAX=OPTIMAL as your site default.

Performance

Performance enhancements for DFSORT applications that use the Blockset technique include the following:

- Dataspace sorting, introduced in R12 for fixed-length record sort applications, now available for variable-length record sort applications (MVS/ESA only)
- Improved data processing methods for fixed-length record sort applications
- OUTFIL processing for producing multiple output data sets using a single pass over one or more input data sets.

OUTFIL Processing

OUTFIL is a new DFSORT control statement that allows you to create one or more output data sets for a sort, copy, or merge application from a single pass over one or more input data sets. You can use multiple OUTFIL statements, with each statement specifying the OUTFIL processing to be performed for one or more output data sets. OUTFIL processing begins after all other processing ends (that is, after processing for exits, options, and other control statements). OUTFIL statements support a wide variety of output data set tasks, including:

- Creation of multiple output data sets containing unedited or edited records from a single pass over one or more input data sets.
- Creation of multiple output data sets containing different ranges or subsets of records from a single pass over one or more input data sets. In addition, records that are not selected for any subset can be saved in another output data set.
- Conversion of variable-length record data sets to fixed-length record data sets.
- Sophisticated editing capabilities such as hexadecimal display and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, symbols (for example, the thousands separator and decimal point), leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.

- Selection of a character or hexadecimal string for output from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).
- Highly detailed three-level (report, page, and section) reports containing a variety of report elements you can specify (for example, current date, current time, page number, character strings, and blank lines) or derive from the input records (for example, character fields, edited numeric input fields, record counts, and edited totals, maximums, minimums, and averages for numeric input fields).

National Language Support

Cultural Sort and Merge

DFSORT will allow the selection of an active locale at installation or run time and will produce sorted or merged records for output according to the collating rules defined in the active locale. This provides sorting and merging for single- or multi-byte character data based on defined collating rules which retain the cultural and local characteristics of a language.

Cultural Include and Omit

DFSORT will allow the selection of an active locale at installation or run time and will include or omit records for output according to the collating rules defined in the active locale. This provides inclusion or omission for single- or multi-byte character data based on defined collating rules which retain the cultural and local characteristics of a language.

OUTFIL Reports

OUTFIL allows date, time, and numeric values in reports to be formatted in many of the notations used throughout the world.

ICETOOL Reports

ICETOOL's DISPLAY operator allows date, time, and numeric values in reports to be formatted in many of the notations used throughout the world.

ICETOOL Enhancements

ICETOOL is now even more versatile as a result of enhancements to the existing operators. The improvements to ICETOOL include:

- Allowing more data to be displayed with the DISPLAY and OCCUR operators. DISPLAY now allows up to 20 fields (increased from 10) and a line length of up to 2048 characters (increased from 121). OCCUR now allows a line length of up to 2048 characters (increased from 121).
- More extensive formatting capabilities for numeric fields with the DISPLAY operator. Formatting items can be used to change the appearance of individual numeric fields in reports with respect to separators, decimal point, decimal places, signs, division, leading strings, floating strings and trailing strings. Thirty-three pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. Leading and trailing strings can also be used with character fields.
- Display of the four-digit or two-digit year with the DISPLAY and OCCUR operators.
- Division of reports into sections with the DISPLAY operator, based on the values in a character or numeric break field. Statistics (total, maximum, minimum and/or average) can be displayed for each section as well as for the entire report.
- Automatic use of OUTFIL processing for a list of TO ddnames with the COPY and SORT operators, resulting in creation of multiple TO (output) data sets from a single pass over the FROM (input) data set.

- Allowing OUTFIL statements to be specified in the USING data set in addition to or instead of the TO operand with the COPY and SORT operators.
- Allowing the active locale to be specified for the COPY, COUNT and SORT operators, in order to override the installation default for the active locale. Thus, multiple active locales can be used in the same ICETOOL job step for these operators.
- Allowing the last record for each unique field value to be kept with the SELECT operator.

INCLUDE/OMIT Substring Search

INCLUDE and OMIT function enhancements provide powerful substring search capability to allow inclusion or omission of records when:

- A specified character or hexadecimal constant is found anywhere within a specified input field (that is, a constant is a substring within a field) or
- A specified input value is found anywhere within a specified character or hexadecimal constant (that is, a field is a substring within a constant).

SMF Type-16 Record Enhancements

New fields, such as information pertaining to each DFSORT run about SORTIN, SORTINnn, SORTOUT and OUTFIL data sets, control statements, record counts, specified values for E15, E35, HIPRMAX, DSPSIZE, FILSZ, LOCALE and AVGRLEN, have been added to DFSORT's SMF type-16 record.

SMF=FULL, SMF=SHORT, and SMF=NO can now be specified in an OPTION statement in DFSPARM or the extended parameter list, to produce or suppress the SMF type-16 record for an individual application.

Note: The offsets of fields ICESPGN, ICEUSER, and ICEGROUP have changed in the Release 13 SMF record. If you have programs that reference those fields, recompile them using the Release 13 version of the ICESMF macro, before attempting to run them against Release 13 SMF records.

Other Enhancements

Several ICEMAC installation options have been added or changed:

- The IBM-supplied default for EXCPVR has been changed from ALL to NONE.
- The IBM-supplied default for DYNAUTO has been changed from NO to YES.
- SDBMSG enables you to specify whether DFSORT should use the system-determined optimum block size for DFSORT message data sets and ICETOOL message and list data sets.
- LOCALE enables you to select an active locale.
- ODMAXBF enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.
- EXPMAX enables you to specify the maximum total amount of available storage to be used for all Hipersorting applications.
- EXPOLD enables you to specify the maximum total amount of old expanded storage to be used at any one time by all Hipersorting applications.
- EXPRES enables you to specify the minimum amount of available expanded storage to be reserved by DFSORT for use by non-Hipersorting applications.

Several run-time options have been added or changed:

- LOCALE enables you to select an active locale.

- SMF enables you to specify whether DFSORT is to produce SMF type-16 records.
- ODMAXBF enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.
- NZDPRINT enables you to indicate that positive ZD summation results are not to be converted to printable numbers (overrides ZDPRINT).
- HILEVEL=YES on the MODS statement enables you to indicate that the E15 and E35 routines are to be treated as COBOL exits.
- DEBUG options BUFFERS=ANY and BUFFERS=BELOW will now be recognized but not used.

DFSORT will now ignore any DD statements not needed for the application (for example, a SORTIN DD statement will be ignored for a merge application).

For unsuccessful completion due to an unsupported operating system, DFSORT, ICEGENER, and ICETOOL will now pass back a return code of 24 to the operating system or invoking program.

The installation initialization exit, ICEIEXIT, enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.

The installation termination exit, ICETEXIT, contains additional fields such as a flag to indicate that OUTFIL processing was used.

For INREC and OUTREC:

- The upper limit for columns and the end of fields has been raised from 32000 to 32752.
- 1: before the RDW field of variable-length records will be accepted and ignored.

For INCLUDE and OMIT, COND=ALL, COND=(ALL), COND=NONE, and COND=(NONE) enable you to include or omit all records.

The L2 value from the RECORD statement will be used if the L1 value is not specified when an E15 or E32 user exit passes all of the input records.

When input is a VSAM data set and output is a non-VSAM data set with RECFM not specified, DFSORT will now set the output RECFM as blocked rather than unblocked, when doing so will allow the use of the system-determined optimum block size for output.

New Programming Support for Release 12 (PTFs)

ICEGENER, copy, and Blockset sort and merge can now be used when a tape output data set is specified with DISP=MOD or DISP=OLD, without specifying the RECFM, LRECL, or BLKSIZE in the DD statement.

Sequential striping is supported for input and output data sets.

Compression is supported for input and output data sets.

BatchPipes/MVS input and output pipes are supported.

New Device Support for Release 12 (PTFs)

Four-digit device numbers are supported.

The IBM 3390-9 DASD is supported for input, output, and work data sets, although it is not recommended for work data sets for performance reasons.

The IBM RAMAC Array DASD and RAMAC Array Subsystem are supported for input, output, and work data sets.

The IBM 3990 Model 6 control unit is supported.

The IBM cached 9343 control unit models are supported.

Index

Numerics

24-bit parameter list
  examples 343, 346
  format 334, 339

A

ABCODE
  ABEND Code 33
  installation option 17
abend
  categories 596
  checkpoint/restart 595
  critical 597
  CTRx processing 597
  processing 595, 598
  processing for unexpected abends 596, 597
  recovery 596, 598
    ESTAE 595
ABEND
  DEBUG control statement option 87
  EXEC PARM option 33
ABSTP
  DEBUG control statement option 88
  processing 596
AC (ISCII/ASCII character) format
  description 582
  INCLUDE statement 96
  SORT statement 267
action codes 468
adding record values 2
adding records 285
  E15 user exit 291, 312
  E35 user exit 318
addressing
  EFS program 456
  EFS program user exit routine 482
  user exits 286
ALIAS statement 288
aliases
  DFSORT 27
  OPTION statement options 177
  PARM options 58
alignment field 113, 252
allocating storage
  intermediate storage 501
  main storage 498
  temporary work space 501
allocating temporary work space efficiently 501, 503
altering records 285
  See also reformatting records 112
ALTSEQ
  installation option 17
ALTSEQ control statement 86
  examples 86
  function 81
  TABLE Option 85
ALTSEQ control statement 86 *(continued)*
  using 85, 86
ALTSEQ Statement Examples 86
AMODE 286, 289
AQ (alternate character) format
  INCLUDE statement 96
  SORT statement 267
ARESALL
  EXEC PARM option 33
  installation option 17
  OPTION control statement option 136
  releasing main storage 500
  using RESERVEX instead of ARESALL 34
ARESINV
  installation option 17
  OPTION control statement option 137
  releasing main storage 500
ASL (ISCII/ASCII leading sign) format
  description 584
  INCLUDE statement 96
  SORT statement 267
Assembler user exit routines
  input phase user exits 290, 299
  output phase user exits 299, 306
AST (ISCII/ASCII trailing sign) format
  description 584
  INCLUDE statement 96
  SORT statement 267
ATTACH
  description 331
  writing macro instructions 343
AVGRLEN
  EXEC PARM option 34
  OPTION control statement option 138

B

BI (binary) format
  description 582
  DISPLAY operator 372
  INCLUDE statement 96
  INREC statement 114
  OCCUR operator 401
  OUTFIL statements 198
  RANGE operator 407
  SELECT operator 411
  SORT statement 267
  STATS operator 419
  SUM statement 274
  UNIQUE operator 421
bit comparison tests 104
bit operators 104
BLDINDEX 507
block
  minimum length 13
blocking records 491
Blockset
  DFSORT 23

© Copyright IBM Corp. 1973, 2000

BSAM
DEBUG control statement option 88
E18 user exit 294
E19 user exit 297
EXEC PARM option 34

C

cache fast write
specifying use of with OPTION control statement 88
using to improve performance 494
cataloged procedures
defined 28
SORT 28
SORT cataloged procedure 28
SORTD 29
SORTD cataloged procedure 29
specifying 28
century window 176, 247, 268, 559
CFW
installation option 17
using on OPTION control statement 88
using to improve performance 494
CH (character) format
description 581
DISPLAY operator 372
INCLUDE statement 96
OCCUR operator 401
SELECT operator 411
SORT statement 267
UNIQUE operator 421
CHALT
installation option 17
OPTION control statement option 138
changing records 2, 285
E15 user exit 290, 291, 312
E35 301, 302
E35 user exit 318
See also reformatting records 112
changing the collating sequence 85
character constants 98
CHECK
installation option 17
OPTION control statement option 138
checkpoint/restart (CHKPT)
restrictions 596
using 595
CINV
EXEC PARM option 35
installation option 17
OPTION control statement option 139
CKPT
efficiency 497
OPTION control statement option 139
SORT control statement option 271
CLO/OL (leading overpunch sign) format
description 583
INCLUDE statement 96
SORT statement 267
closing data sets
E17 user exit 294
closing data sets *(continued)*
E37 user exit 304
housekeeping 466
with an EFS program 462, 466
with user exits 286
COBEXIT
efficiency 494
EXEC PARM option 35
installation option 17
OPTION control statement option 140
COBOL
input phase user exits 312
output phase user exit 318
overview 309
requirements for copy processing 310
storage requirements 311
user exit routine requirements 309
user exit routines 309, 312, 318
COBOL E15 user exit
altering records 324
changing records for Sort 312
passing records for Sort 312
COBOL E35 user exit
changing records 318
inserting records 325
CODE
ALTSEQ control statement option 85
coding control statements 81
coding restrictions 84, 85
collating sequence 85
altering with user exit 284
alternate 6
defined 5
EBCDIC 5
ISCII/ASCII 5
modifying 6
combining data sets
See merging records 124
comment statement 84
Compare Field Formats and Lengths Table 96
comparison operator 96, 109
comparisons
OMIT control statement 131
COND
INCLUDE control statement option 94
OMIT control statement option 132
considerations
data set 11
key-sequenced data set (KSDS) 14
QSAM data set 14
record descriptor word (RDW) 15
VSAM data set 14
constants
bit string 107, 113
character string 98
date string 110
decimal number 98
hexadecimal string 99
continuation column 82
continuation lines 83, 84
continuing control statements 83

control field
  defined 4, 5
  deleting
    with INREC control statement 112
    with OUTREC control statement 251
  describing on MERGE control statement 125
  describing on SORT control statement 265
  efficient design 491
  equal 5
  format 267
  length 267
  modifying with E61 user exit 298
  modifying with user exit 284
  overview 4, 6
  reordering
    with INREC control statement 112
    with OUTREC control statement 251
Control Field Formats and Lengths Table 267
control statement
  coding 81
  coding restrictions 84
  comment statement 84
  continuation column 82
  EFS coding rules 469, 471
  EFS interface request list 469
  EFS string 469
  examining, altering, or ignoring 464
  format 82
  functions 80, 81
  label field 82
  operand field 82
  operation field 82
  overview 79
  preparing image 333
  printing with an EFS program 466
  remark field 82
  request list 469
  string returned by the EFS program 471
  string sent to the EFS program 469
  summary 80, 81
  using with EXEC statement 29
control statements
  using other IBM programs 85
control word 265
conventions, notational xv
CONVERT parameter
  OUTFIL control statements option 184, 212
COPY
  OPTION control statement option 140
copy examples 532, 534
COPY operator (ICETOOL) 360
copy restrictions 347, 348
copying
  data set requirements 11
  defined 1
  overview 11
copying records
  SORT control statement option 270
  with MERGE control statement 125
COUNT operator (ICETOOL) 364
critical errors 597
CSF/FS (floating sign) format
  description 583
  DISPLAY operator 372
  INCLUDE statement 96
  OCCUR operator 401
  OUTFIL statements 198
  RANGE operator 407
  SELECT operator 411
  SORT statement 267
  STATS operator 419
  UNIQUE operator 421
CSL/LS (leading sign) format
  description 583
  SORT statement 267
CST/TS (trailing sign) format
  description 583
  INCLUDE statement 96
  SORT statement 267
CTO/OT (trailing overpunch sign) format
  description 584
  INCLUDE statement 96
  SORT statement 267
CTRx
  abend processing 597
  DEBUG control statement option 89
cultural environment
  See LOCALE 6
cylinders 492, 546

D

D1 format
  FIELDS operand 473
  SORT statement 267
D2 format
  COND operand 473
  INCLUDE statement 96
DASD
  capacity considerations 546, 547
  efficiency 492, 501
  exceeding capacity 547
data formats 376
  descriptions 581, 589
  DFSORT statements 586, 587
  ICETOOL operators 586
  OUTFIL statement operands 586, 587
data management rules
  system data management rules 12
data set 11
  closing 286
  closing with user exit routines 294, 304
  defining 11
  handling input with user exit routines 305
  handling output with user exit routines 305
  input 10
    shared tape unit 61
  key-sequenced, considerations 14
  message data set 22
  notes and limitations 12, 15
  opening with user exit routines 284, 290, 300
  output 10
    shared tape unit 61

data set 11 *(continued)*
page=end.considerations 15
QSAM considerations 14
requirements 11
SmartBatch pipe 15
system data management rules 12
valid types 11
VSAM considerations 14
data space
definition 141
specifying with EXEC PARM 36
specifying with OPTION control statement 141
data types 12
dataspace sorting
advantages 503
considerations 503
definition 503
date constant 110
date formats
INCLUDE and OMIT 131
OUTFIL 195
SORT and MERGE 267
DBCS ordering 456
DD statements
overview 59
program DD statements 64
summary 26
system DD statements 62
using 59, 76
ddnames
duplicate 61
DEBUG control statement
example 87, 91
function 81
special handling 471
using 87, 92
DEBUG Statement Examples 91
debugging jobs 87
decimal number constants 98
defaults
installation 16
listing with ICETOOL 17
defaults, installation 16
DEFAULTS operator (ICETOOL) 365
definitions
cataloged procedures 28
collating sequence 5
control field 4
copying 1
DD statements 26
direct invocation 4
EXEC statement 27
installation options 17, 21
JOB statement 27
key 5
merging 1
program invocation 4
sorting 1
deleting control fields
with INREC 112
with OUTREC control statement 252
deleting records 285
E15 user exit 291, 312
E35 user exit 318
with INCLUDE control statement 93, 130
with OMIT control statement 130
designing applications to maximize performance 490, 497
designing new applications 491
determining action when intermediate storage is insufficient 286
devices, improving elapsed time with 494
DFSORT 16
calls to your EFS program 458
compatible operating systems 4
dynamic invocation 331
exit routines 280
improving efficiency 490
invoking 3
job control statements 25, 76
logic examples for input/user exit/output 283
messages 22
operating as a guest under VM 4
override of options 549
overview 1
processing order 6
processing OUTFIL operands 186
program control statements 79, 277
program phases 281, 283, 457
terminating with user exit 286
DFSORT home page 3
DFSORT phases
definition 457
initialization 460, 486
input 462
termination 462, 488
DFSPARM data set 550
DFSPARM DD statement
defined 26
function 64
using 74, 76
DFSPARM statement
PARM options 29
alias PARM options 58
diagnosis
EFS program 486
diagnostic messages 22
DIAGSIM
installation option 18
direct access storage devices
See DASD 492
direct access work storage devices 501
direct invocation
definition 4
DFSORT processing 490
using JCL 551
DISPLAY operator (ICETOOL) 369
Double Byte Character Set (DBCS) ordering
See DBCS ordering 12
Double Byte Character Set Ordering Support Program
See DBCS ordering 456

DSA (Dynamic Storage Adjustment)
enhancing performance 494
installation option 18
limit 553
DSPSIZE
enhancing performance 494
EXEC PARM option 35
installation option 18
OPTION control statement option 141
duplicate ddnames 61
duplicate records
OCCUR operator (ICETOOL) 399
SELECT operator (ICETOOL) 409
SUM control statement 274
DYNALLOC
EXEC PARM option 36
OPTION control statement option 141
SORT control statement option 271
DYNALLOC=OFF
EXEC PARM option 37
OPTION control statement option 143
DYNALOC
installation option 18
dynamic link-editing
See link-editing 289
Dynamic Storage Adjustment (DSA)
enhancing performance 494
installation option 18
limit 553
dynamically-invoked DFSORT
with the 24-bit parameter list 570, 581
with the extended parameter list 560, 570
DYNAUTO
installation option 18
DYNSPC
installation option 18

E

E11 user exit
initializing routines 290
opening data sets 290
E15/E35 return codes and EXITCK 327, 331
E15 user exit
changing records for sort and copy applications 290
EXEC PARM option 38
passing records for sort and copy applications 290
return codes 292
E15 User Exit
altering record length 306
interface with COBOL 312
LINKAGE SECTION code example for fixed-length records 315
LINKAGE SECTION code example for variable-length records 316
LINKAGE SECTION fields for fixed-length records 315
LINKAGE SECTION fields for variable-length records 315
PROCEDURE DIVISION requirements 318
return codes 316
E16 user exit
handling intermediate storage miscalculation 293
return codes 293
sorting current records when NMAX is exceeded 307
E17 user exit
closing data sets 294
E18 user exit
handling input data sets 294
using with QSAM/BSAM 294
using with VSAM 295
E19 user exit
handling output to work data sets 297
using with QSAM/BSAM 297
E31 user exit
initializing routines 300
opening data sets 300
E32 user exit
handling input to a merge only 300
restriction with MERGE control statement 127
return codes 301
E35 user exit
altering record length 308
Changing Records 301
EXEC PARM option 38
interface with COBOL 319
LINKAGE SECTION fields for fixed-length records 321
LINKAGE SECTION fields for variable-length records 321
Procedure Division Requirements 324
return codes 303
E37 user exit
closing data sets 304
E38 user exit
handling input data sets 305
using with VSAM 305
E39 user exit
handling output data sets 305
using with QSAM/BSAM 305
using with VSAM 305
E61 user exit
altering control fields 308
information DFSORT passes to your routine 298
modifying control fields 298
uses 298
edit masks
ICETOOL DISPLAY operator 374, 375
OUTFIL 199, 206
editing records 2
efficiency
using main storage 490
EFS 472
efficiency 497
EXEC PARM option 37
exit routines 462
initialization phase 460
input phase 462
installation option 18
OPTION control statement option 143
phases 458

EFS 472 *(continued)*
processing 458
termination phase 462
using 456, 488
what you can do with EFS 463, 466
EFS interface
control statement length 475
control statement request list 469
control statement string 469, 471
D1 format 473
D2 format 473
defined 466
DFSORT action codes 468
extract buffer offsets list 475
function 457
information flags 476
message list 478
program context area 475
record lengths list 476
EFS program
activating 457
addressing and residence mode 456
closing data sets 466
context area 475
examining, altering, ignoring control statements 464
example 486
exit routine 466, 479, 480
function 479
functions 456, 463
interface parameter list 466, 478
opening and initializing data sets 464
restrictions program in effect 84
restrictions when program in effect 85
return codes you must supply 483
supplying messages 466
terminating DFSORT 466
user exit routine
addressing and residence mode 482
EFS Program
example 488
EFS01
function description 479
parameter list 480
user exit routine 479
EFS02
address=0 488
EFS02
function description 479
parameter list 482
user exit routine 480
EFSDPAFT 486
DEBUG control statement option 89
EFSDPBFR 486
DEBUG control statement option 89
elapsed time
improving with devices 494
END control statement
examples 92
function 81
using 92
ENDREC parameter
OUTFIL control statements option 184, 188
enhancing performance with installation options 494
EODAD 295
EQUALS 5
efficiency 497
EXEC PARM option 38
installation option 18
MERGE control statement option 125
OPTION control statement option 144
SORT control statement option 271
EQUCOUNT
DEBUG control statement option 90
efficiency 497
ERET
installation option 18
EROPT 295
error messages 22
error recovery routine
user exit 285
errors
critical 597
debugging jobs 87
diagnosing EFS 486
error recovery routines 285
ESTAE
DEBUG control statement option 90
installation option 18
recovery routine 595
exceeding tape work space capacity 547, 548
EXEC statement
cataloged procedure
SORT 28, 63
SORTD 29, 64
cataloged procedures 28
defined 27
operands 30, 57
PARM options 29, 550
alias PARM options 58
syntax 30
using 27, 59
using with control statements 29
execution phase 457
exit
MODS control statement option 128
See also user exit 280
exit routine
EFS 478
EXITCK
ICEMAC installation option 290, 327
installation option 18
OPTION control statement option 145
user exit return codes 327
EXLST 295, 298
EXPMAX installation option 18, 496, 539
EXPOLD installation option 18, 496, 539
EXPRES installation option 18, 496, 539
Extended Function Support
See EFS 456
extended parameter list
example 346

extended parameter list *(continued)*
format 340, 343
extract buffer offsets list 475

F

FASTSRT
efficiency 494
FI (fixed-point) format
description 582
DISPLAY operator 372
INCLUDE statement 96
OCCUR operator 401
OUTFIL statements 198
RANGE operator 407
SELECT operator 411
SORT statement 267
STATS operator 419
SUM statement 274
UNIQUE operator 421
Field and Constant Symbols
overview 433
field formats
compare 96
control 267
ICETOOL operators
DISPLAY 372
OCCUR 401
RANGE 407
SELECT 411
STATS 419
UNIQUE 421
VERIFY 423
summary 274
FIELDS
D1 format (EFS) 473
INREC control statement option 113
MERGE control statement option 125
OUTREC control statement option 252
SORT control statement option 265
SUM control statement option 274
FIELDS=(COPY)
SORT control statement option 270
FIELDS=COPY
MERGE control statement option 125
SORT control statement option 270
FILES parameter
MERGE control statement option 125
OUTFIL control statements option 183, 187
FILSZ
EXEC PARM option 38
variation summary 40
MERGE control statement option 126
OPTION control statement option 145
SORT control statement option 271
filtering records 93, 130
fixed century window 176
FL (floating-point) format
description 582
SORT statement 267
SUM statement 274
floating-point data 111, 270, 284
floating-point fields 276
floating sign format
using ICETOOL 373
FNAMES parameter
OUTFIL control statements option 183, 186
format
alternate character format 96
binary format 96
character format 96
fixed-point format 96
floating-point format 96
floating sign format 96
ISCII/ASCII character format 96
ISCII/ASCII leading sign format 96
ISCII/ASCII trailing sign format 96
leading overpunch sign format 96
leading sign format 96
packed decimal format 96
trailing overpunch sign format 96
trailing sign format 96
user defined format (D1) 96
user defined format (D2) 96
zoned decimal format 96
FORMAT=f
INCLUDE control statement option 93, 95
MERGE control statement option 125
OMIT control statement option 130, 132
SORT control statement option 270
SUM control statement option 275
format of 24-bit parameter list 334, 339
format of extended parameter list 340, 343
formats for SORT, MERGE, INCLUDE, and OMIT control statements 472
formatting
OUTFIL 198
four-digit year
transforming dates 176
FSZEST installation option 18
FTP site 3
functions of routines at user exits 283, 286

G

GENER installation option 18
general coding rules 81, 85
general considerations 12, 13
GNPAD installation option 18, 505
GNTRUNC installation option 18, 505

H

handling input data sets
E18 user exit 294
E38 user exit 305
handling input to a merge
E32 user exit 300
handling intermediate storage miscalculation
E16 user exit 293
handling output data sets
E39 user exit 305
handling output to work data sets
E19 user exit 297

handling special I/O 285
HEADER1 parameter
  OUTFIL control statements option 184, 213, 216
HEADER2 parameter
  OUTFIL control statements option 184, 222
HEADER3 parameter
  OUTFIL control statements option 230
hexadecimal constants 99
hexadecimal display
  DISPLAY operator 376
  OCCUR operator 402
HFS 15
HILEVEL=YES
  MODS control statement option 128
Hipersorting
  advantages to using 503
  defined 503
Hiperspace
  defined 503
  limiting factors 149
HIPRMAX
  efficiency 503
  EXEC PARM option 41
  installation option 19
  OPTION control statement option 149
home page (web) 3
how EFS works 457, 463
how user exit routines affect DFSORT
  performance 287

I

I/O errors 285
ICEGENER
  efficiency 504
  example 534, 535
  return codes 506
ICEGENER facility 504, 506
ICEMAC installation options 17, 21
ICETOOL 350
  calling from a program 424
  coding rules 359
  complete sample job 535
  description 350
  example of simple job 353
  examples 354, 355, 362, 365, 385, 398, 404, 408, 412, 416, 420, 421, 423
  ICETOOL/DFSORT relationship 351
  invoking 354
  JCL 351
    DFSMSG DD statement 351
    JOBLIB DD statement 351
    restrictions 359
    statements 357
    STEPLIB DD statement 351
    summary 351
    SYMNAMES DD statemen 351
    SYMNOUT DD statemen 351
    TOOLIN DD statement 351, 358
    TOOLMSG DD statement 351, 357
  operators 352
    COPY 352, 356, 360
ICETOOL 350 *(continued)*
  operators 352 *(continued)*
    COUNT 352, 364
    DEFAULTS 365
    DISPLAY 352, 355, 369
    MODE 352, 354, 356, 397
    OCCUR 352, 355, 398
    RANGE 352, 355, 406
    SELECT 352, 356, 409
    SORT 352, 356, 414
    STATS 353, 355, 418
    summary 352
    UNIQUE 353, 356, 420
    VERIFY 353, 355, 422
  Parameter List Interface 354, 358, 424
  restrictions 430
  return codes 431
  statements 359
  TOOLIN Interface 354, 358, 424
  using symbols 353
IDRCPCT installation option 19
IEBGENER 504
IEFUSI 500, 501
IEXIT installation option 19
IGNCKPT installation option 19
improving efficiency 490, 506
INCLUDE control statement
  efficiency 496
  examples 100, 109
  function 80
  logical operator 111
  relational condition 95
    comparison operator 96, 109
  substring comparison operator 103
INCLUDE/OMIT Statement Notes 111, 112
INCLUDE parameter
  OUTFIL control statements option 184, 188
including records 1, 93, 130
  user-defined data types 456
information DFSORT passes to your routine
  E15 user exit 291
  E32 user exit 300
  E35 user exit 302
  E61 user exit 298
information flags 476
Initialization Phase 460
initializing data sets 284, 460
initializing routines
  E11 user exit 290
  E31 user exit 300
initiating DFSORT
  See invoking DFSORT 331
INPFIL control statement 85
input data set
  requirements 11
  valid types 11
input field 116
input file
  size and efficiency 492
Input Phase 462

INREC control statement
column alignment 113
examples 120, 123
function 80
input field 116
notes 119, 120
separation field
binary zero separation 114
blank separation 114
character string separation 115
hexadecimal string separation 115
using 112, 122, 123
inserting comment statements 84
inserting records 285
installation defaults 16
displaying with DEFAULTS operator (ICETOOL) 365
listing with ICETOOL 17
summary of options 17
installation options 17, 21
See ICEMAC 136
installation options, using to enhance performance 494
insufficient intermediate storage 547
intermediate storage 548
Internet 3
introducing DFSORT 1, 23
invoking DFSORT
24-bit parameter list 333, 339
dynamically 331
extended parameter list 339, 343
from a program 331, 348
methods 3
using JCL 25
IOMAXBF installation option 19

J

Japanese characters 12, 456
JCL 25
cataloged procedure 63, 64
cataloged procedures, specifying 28
DD statement summary 26
EFS coding rules 470
EXEC statement 27
improving DFSORT efficiency 490
JOB statement 27
overview 25
procedures, cataloged 28
required 25
JCL DD statements 332
JCL DD Statements 341
JCL-invoked DFSORT 551, 560
job control language
see also JCL 25
JOB statement
defined 27
using 27
JOBLIB DD statement
defined 26
using 62

K

keeping records 1
key, defined 5
key-sequenced data set (KSDS) 14

L

label field 82
length
altered control statement 475
LRECL for variable-length record 15
maximum record 12
original control statement 475
record descriptor word (RDW) 15
record lengths list 476
limitations
data set 12
length
maximum record 12
minimum block 13
minimum record 13
record
maximum length 12
storage constraints 13
LINES parameter
OUTFIL control statements option 184, 213
LINK 331
writing macro instructions 343
link-editing
performance 497
user exit routines 289
linkage conventions 288
linkage editor 63
linkage examples 289
LIST
EXEC PARM option 42
installation option 19
OPTION control statement option 150
with an EFS program 466
LISTX
EXEC PARM option 42
installation option 19
OPTION control statement option 151
with an EFS program 466
loading user exit routines 288
locale
affecting INCLUDE and OMIT processing 100
affecting MERGE processing 124
C/370 support 599, 601
defined 6
restrictions
CHALT 138
EFS 37, 143
LOCALE
efficiency 496, 497
EXEC PARM option 43
installation option 19
OPTION control statement option 151
using 6
logical operator 111
lookup and change 184, 209, 246

M

macro instructions
  See system macro instructions 331
main features of sources of DFSORT options 550, 551
main storage
  allocating
    consequences of increasing 499
  allocating efficiently 498
  factors affecting requirements 498
  minimum 498
  releasing 500
  tuning 498
  using efficiently 498, 501
MAINSIZE 49
  allocating storage 498
  OPTION control statement option 153
  releasing main storage 500
Major Call 1 486
Major Call 2 487
Major Call 3 488
Major Call 4 488
Major Call 5 488
major control field 5
managing system data, rules
  system data management rules 12
master console messages 22
maximizing performance 490
MAXLIM
  allocating storage 498
  installation option 19
  releasing main storage 500
MERGE control statement
  examples 126, 127
  function 80
  using 124, 127
merge examples 530, 532
merge restriction 347
merging
  data set requirements 12
  defined 1
  overview 12
  records 124
  specifying the estimated number of records to merge 39
  specifying the exact number of records to merge 39
  specifying the number of records to merge 40
  user-defined data types 456, 462
message data set 22
message list 478
messages
  master console messages 22
  message data set 22
  return codes 22
migration 21
minimum block length 13
minimum record length 13
MINLIM
  allocating storage 498
  installation option 19
minor control field 5
modifying 85
modifying control fields
  E61 user exit 298
  with user exit 284
modifying the collating sequence 85
MODS control statement
  examples 130
  function 81
  using 127, 130
MSGCON installation option 19
MSGDDN
  EXEC PARM option 43
  installation option 19
  OPTION control statement option 154
MSGPRT
  alternate forms 44
  EXEC PARM option 44
  installation option 19
  OPTION control statement option 154
multiple output data sets
  creating with OUTFIL 3, 182, 239

N

NOABEND
  DEBUG control statement option 87
  EXEC PARM option 33
NOASSIST
  DEBUG control statement option 91
NOBLKSET
  efficiency 497
  OPTION control statement option 155
NOCFW
  using on OPTION control statement 88
NOCHALT
  OPTION control statement option 138
NOCHECK
  OPTION control statement option 138
NOCINV
  efficiency 497
  EXEC PARM option 35
  OPTION control statement option 139
NODETAIL parameter
  OUTFIL control statements option 184, 236
NOEQUALS
  EXEC PARM option 38
  MERGE control statement option 125
  OPTION control statement option 144
NOESTAE
  DEBUG control statement option 90
NOLIST
  EXEC PARM option 42
  OPTION control statement option 150
  with an EFS program 466
NOLISTX
  EXEC PARM option 42
  OPTION control statement option 151
  with an EFS program 466
NOMSGDD installation option 19
NOOUTREL
  EXEC PARM option 45
  OPTION control statement option 155

NOOUTSEC
OPTION control statement option 155
NORESET
EXEC PARM option 47
NOSOLRF
EXEC PARM option 50
OPTION control statement option 164
NOSTIMER
EXEC PARM option 51
OPTION control statement option 156
NOSZERO
EXEC PARM option 52
OPTION control statement option 168
notational conventions xv
NOVERIFY
EXEC PARM option 53
OPTION control statement option 170
NOVLLONG
EXEC PARM option 53
OPTION control statement option 170
NOVLSCMP
EXEC PARM option 54
OPTION control statement option 171
NOVLSHRT
EXEC PARM option 54
OPTION control statement option 173
NOVSAMIO
EXEC PARM option 55
NOWRKREL
EXEC PARM option 56
OPTION control statement option 175
NOWRKSEC
EXEC PARM option 56
OPTION control statement option 176
numerice editing and formatting
DISPLAY operator 373
NVSAMEMT
EXEC PARM option 55
NZDPRINT
EXEC PARM option 57
OPTION control statement option 176

O

occurrences
OCCUR operator (ICETOOL) 399
SELECT operator (ICETOOL) 409
ODMAXBF
EXEC PARM option 45
installation option 19
OPTION control statement option 156
OUTFIL control statements option 236
OMIT control statement
efficiency 496
example 133
function 80
using 133
OMIT parameter
OUTFIL control statements option 184, 189
OMIT Statement Example 133
omitting records 1, 130
omitting records 1, 130 *(continued)*
user-defined data types 456
opening and initializing data sets 284, 464
opening data sets
E11 user exit 290
E31 user exit 300
EFS 460
user exit routines 284
operand field 82
operating systems, compatible 4
operation field 82
OPTION control statement
examples 178, 182
function 80
special handling 471
using 134, 182
OPTION Statement Examples 178, 182
options, installation 16
OUTFIL
DD statement 71
digits needed for numeric fields 201
edit field formats and lengths 198
edit mask output field lengths 202
edit mask patterns 199
edit mask signs 200
efficiency 496
lookup and change 184, 209, 246
producing reports 184, 191
storage limits 153, 237, 499
table lookup and change 209, 246
OUTFIL control statements
examples 239, 248, 249, 250
function 80
using 182, 248, 249, 250
outfil DD statement
defined 26
function 64
OUTFIL statements examples 239, 248, 249, 250
OUTFIL statements notes 236, 239
output data set
requirements 11
valid types 11
OUTREC control statement
column alignment 252
differences from OUTREC parameter 251
examples 257, 259, 260
function 81
input field 253
separation field
binary zero separation 253
blank separation 252
character string separation 253
hexadecimal string separation 253
using 251, 259, 260
OUTREC parameter
lookup 184, 209, 246
OUTFIL control statements option 184, 190, 212
OUTREC statement examples 257, 259, 260
OUTREC statement notes 256, 257
OUTREL
EXEC PARM option 45

OUTREL *(continued)*
installation option 19
OUTSEC installation option 19
overflow 120, 276
OVERRGN 501
installation option 19
releasing main storage 500
override tables 551
overriding
defaults 549
installation defaults 136
Overriding control statements 332
overview, DFSORT 1
OVFLO
EXEC PARM option 46
installation option 19
OPTION control statement option 157

P

PAD
EXEC PARM option 46
installation option 20
OPTION control statement option 158
padding
GNPAD 505
INCLUDE/OMIT 100
records 13, 100
truncating 100
PAGEHEAD parameter
OUTFIL control statements 231
parameter list
control statements 333, 339
description 550, 551
format 334, 340
PARM options
alias PARM options 58
PARMDDN installation option 20
passing control to user exits 127
passing records
E15 user exit 290, 312
PD (packed decimal) format
description 582
DISPLAY operator 372
INCLUDE statement 96
OCCUR operator 401
OUTFIL statements 198
RANGE operator 407
SELECT operator 411
SORT statement 267
STATS operator 419
SUM statement 274
UNIQUE operator 421
VERIFY operator 423
PD0 (part of packed decimal) format
OUTFIL control statement 198
SORT statement 267
performance
application design 491
dataspace sorting 503
efficient blocking 491
performance *(continued)*
Hipersorting 503
HIPRMAX 503
ICEGENER 504
improving elapsed time with devices 494
JCL 490
main storage 498
maximizing 490
merging techniques 492
ODMAXBF effects 237
options that degrade 497
options that enhance 494
sorting techniques 491
specifying data sets 492
temporary work space 501
using BLDINDEX support 507
using DFSORT's Performance Booster for The SAS System 506
using SmartBatch pipes 493
using VIO in expanded storage 493
Pipes
See SmartBatch pipe 15
procedures, catalogued
defined 28
specifying 28
processing and invoking programs 598
processing of error abends with A-type messages 597
processing order, record 6
processing user-defined data types with EFS program user exit routines 466
program control statements
extended parameter list 339, 343
using with EXEC statement 29
program DD statements 64
Program DD statements 76
program invocation, defined 4
program phase
defined 281
initialization 460
input 462
termination 462

Q

QSAM
data set 11
data set considerations 14
E18 user exit 294
E19 user exit 297

R

RANGE operator (ICETOOL) 406
rearranging records
See sorting records 264
record
blocking 491
changing with user exit routines 301, 302
copying 125
data types 12
deleting 93, 130

record *(continued)*
with OMIT control statement 130
descriptor word (RDW) 15
EFS constraints 13
estimated number to be sorted 39
exact number to be sorted 39
formatting 112
inserting, deleting, and altering 285
maximum length 12
merging 124
minimum length 13
modifying with user exit 285
number to be sorted 40
padding 100, 505
passing with user exit routines 290
processing for OUTFIL 184
processing order 7, 111, 119, 120, 256
EFS 483
reformatting 251
sorting 264
storage constraints 13
summing 2, 274
E35 user exit 304
with user exits 285
truncating 100, 505
user-defined data types 456
variable-length
efficiency 492
RECORD control statement
coding notes 263
examples 263
function 81
using 260
record processing order 483, 486
record type
specifying 260
records
duplicate 274, 399, 409
unique
OCCUR operator (ICETOOL) 399
SELECT operator (ICETOOL) 409
UNIQUE operator (ICETOOL) 420
recovering from unexpected abends 596
reformatting records 2, 285
with INREC 112
with OUTREC 251
REGION
allocating storage 498
determining storage 498
releasing main storage 500
size 498
relational condition
comparison operator 96, 109
constants
character string format 98
date string format 110
decimal number format 98
hexadecimal string format 99
defined 95
description 95
format 95, 100, 109
releasing main storage 500
remark field 82
RENT 288
reordering control fields
See reformatting records 112, 251
report
ASA carriage control character 185, 191, 213, 217, 222, 224, 228, 230, 232, 237
header, OUTFIL 213
ICETOOL DISPLAY 370, 397
ICETOOL OCCUR 399, 406
OUTFIL elements 3, 183
producing for OUTFIL 184, 191
trailer, OUTFIL 217
requesting a SNAP dump 486
requirements
input data set 11
JCL 25
main storage
factors affecting 498
output data set 11
RESALL
EXEC PARM option 47
installation option 20
OPTION control statement option 158
RESERVEX 34
RESET
installation option 20
OPTION control statement option 159
residence mode
EFS program 456
EFS program user exit routine 482
user exits 286
RESINV 500
installation option 20
OPTION control statement option 160
restarting after an abend 595
restrictions for dynamic invocation 347, 348
Return Code
DFSORT 22
return codes
EFS 483
Return Codes
ICEGENER 506
ICETOOL 431
REXX examples 510
RMODE 289
rules, for managing system data
system data management rules 12
rules for parsing 471
run-time phase 457
running DFSORT with JCL 59, 76

S

sample job streams 509
sample jobs listing installation defaults 17
sample routines written in Assembler 306, 309
sample routines written in COBOL 324, 327
SAS
DFSORT's Performance Booster for The SAS System 506

SAVE parameter
  OUTFIL control statements option 184, 190
SDB
  EXEC PARM option 48
  installation option 20
  OPTION control statement option 160
SDB (system-determined block size) installation option 72
SDBMSG installation option 20
SECTIONS parameter
  OUTFIL control statements option 184, 228
SELECT operator (ICETOOL) 409
separation field 114, 252
shared tape units 61, 62
SIZE
  allocating storage 498
  EXEC PARM option 49
  installation option 20
  MERGE control statement option 126
  OPTION control statement option 145, 163
  releasing main storage 500
  SORT control statement option 271
SKIP parameter
  OUTFIL control statements 229
SKIPREC
  efficiency 496
  EXEC PARM option 50
  MERGE control statement option 126
  OPTION control statement option 163
  SORT control statement option 271
sliding century window 176
SmartBatch pipe
  and ICETOOL 430
  considerations 15
SmartBatch Pipe
  Sort example 526
SmartBatch pipes
  OUTFIL example 247
SMF
  OPTION control statement option 164
SMF installation option 20
SNAP dump 486
SOLRF
  EXEC PARM option 50
  installation option 20
  OPTION control statement option 164
SORT cataloged procedure 28, 29, 63
SORT control statement
  effects of EQUALS 265
  examples 272, 274
  field formats 267
  function 80
  using 264, 274
sort examples 511, 529
SORT operator (ICETOOL) 414
SORT statement examples 272, 274
SORT statement image example 333, 334
SORT statement note 272
SORTCKPT DD statement
  function 64
  using 73
SORTCNTL data set 550
SORTCNTL DD statement
  defined 26
  function 64
  using 73, 74
SORTD cataloged procedure 64
SORTDD
  OPTION control statement option 165
SORTDIAG DD statement
  defined 26
  function 65
  using 76
SORTDKdd DD statement
  function 65
  using 76
SORTIN
  OPTION control statement option 166
SORTIN DD statement
  defined 26
  function 64
  using 66, 67
sorting
  data set requirements 11
  defined 1
  identifying information to sort 4
  overview 11
  records 264
  specifying the estimated number of records to sort 39
  specifying the exact number of records to sort 39
  specifying the number of records to sort 40
  user-defined data types 456, 462
  using data space 494
sorting records 264
SORTINnn DD statement
  defined 26
  duplicate 61
  function 64
  using 67, 69
SORTLIB
  ICEMAC installation option 65
SORTLIB DD statement
  defined 26
  function 64
  using 65, 66
SORTLIB installation option 20
SORTMODS DD statement
  defined 27
  function 65
SORTOUT
  OPTION control statement option 166
  OUTFIL ddname 183
SORTOUT DD statement
  defined 26
  function 64
  using 71, 73
SORTSNAP DD statement
  defined 26
  function 65
  using 76

SORTWKdd DD statement
  defined 26
  duplicate 61
  function 64
  using 69
SPANINC
  EXEC PARM option 51
  installation option 20
  option control statement 167
special handling of OPTION and DEBUG control statements 471
specification/override of DFSORT options 549, 581
specifying efficient sort/merge techniques 491
specifying input/output data set characteristics accurately 492
SPLIT parameter
  OUTFIL control statements option 185, 190
STARTREC parameter
  OUTFIL control statements option 184, 188
STATS operator (ICETOOL) 418
STEPLIB DD statement
  defined 26
  using 62
STIMER
  EXEC PARM option 51
  installation option 20
STOPAFT
  efficiency 496
  EXEC PARM option 52
  MERGE control statement option 126
  OPTION control statement option 167
  SORT control statement option 271
storage
  efficient 492, 547
  exceeding capacity 547
  intermediate 501
  limits, OUTFIL 237
  main
    factors affecting requirements 498
    releasing 500
    tuning 498
  specifying for user exit routine 128, 130
  temporary 501
  tracks versus cylinders 492, 546
  user exit routine 286, 311
storage administrator examples 510
storage usage
  records at E35 user exit 304
substring comparison operator 103
substring comparison tests 103
  relational condition format 102
SUM control statement 277
  description 274
  efficiency 496
  examples 277
  function 81
  summary field 274
  using 277
SUM statement examples 277
SUM statement notes 275, 277
summarizing records 274
summary field
  formats 274
  table of formats and lengths 274
Summary Field Formats and Lengths Table 274
summing
  records 274, 285
  records at E35 user exit 304
summing records 2
supplying messages for printing to the message data set 466
SVC installation option 20
SYMNAMES DD statement
  defined 26
  function 64
SYMNOUT DD statement
  defined 26
  function 64
SYNAD 295, 298
syntax diagrams
  notational conventions xv
  option control statement 134
SYSABEND DD statement
  defined 26
  using 63
SYSIN data set 550
SYSIN DD statement
  defined 26
  using 62
SYSLIN DD statement
  defined 27
  using 63
SYSLMOD DD statement
  defined 27
  using 63
SYSMDUMP DD statement
  defined 26
  using 63
SYSOUT DD statement
  defined 26
  using 63
SYSPRINT DD statement
  defined 27
  using 63
system DD statements 62, 64
system-determined block size (SDB) 72
system macro instructions
  defined 331
  using 331, 343
  writing 343, 347
SYSUDUMP DD statement
  defined 26
  using 63
SYSUT1 DD statement
  defined 27
  using 63
SZERO
  EXEC PARM option 52
  installation option 20
  OPTION control statement option 168

T

tape
  capacity considerations 547, 548
  efficiency 497, 502, 547
  insufficient intermediate storage 547
  work space capacity 547
  work storage devices 502
terminating DFSORT
  E35 user exit 318
  with an EFS program 466
  with user exits 286
TEXIT installation option 20
TMAXLIM
  allocating storage 498
  installation option 20
  releasing main storage 500
tracks 492, 546
TRAILER1 parameter
  OUTFIL control statements option 184, 216, 222
TRAILER2 parameter
  OUTFIL control statements option 184, 223, 227
TRAILER3 parameter
  OUTFIL control statements option 232
TRUNC
  EXEC PARM option 52
  installation option 20
  OPTION control statement option 169
truncating
  GNTRUNC 505
  INCLUDE/OMIT 100
  records 13
truncating records 100
tuning main storage 498
two-digit year
  conversion 176, 247
  sorting 274
  transforming dates 3, 183
TYPE
  RECORD control statement option 260

U

unexpected abends 596
UNIQUE operator (ICETOOL) 420
unique records
  OCCUR operator (ICETOOL) 399
  SELECT operator (ICETOOL) 409
  UNIQUE operator (ICETOOL) 420
user exit
  activating 280
  addressing and residence mode 286
  assembler routines
    input phase 290
    output phase 299
  COBOL routines
    input phase 312
    output phase 318
    overview 309
  conventions for routines 287
  DFSORT performance 287
user exit *(continued)*
  E11 290
  E15 290, 312
  E16 293
  E17 294
  E18 294
  E19 297
  E31 300
  E32 300
  E35 301, 302, 318
  E37 304
  E38 305
  E39 305
  E61 298
  efficiency 497
  functions 283
  language requirements 280
  link-editing 289
  linkage conventions 288
  loading routines 288
  overview 280
  passing control with MODS control statement 127
  summary of rules 287, 290
  using RECORD control statement 260
  using routines 280, 306
  using your own routines 306, 331
user exit linkage conventions 288
USEWKDD
  OPTION control statement option 170
using control statements from other IBM programs 85
using DD statements 59, 76
using DFSORT program control statements 79, 277
using options that enhance performance 494
using Symbols
  Comment and Blank Statement 436
  example 434
  for Fields and Constants 433
  in ICETOOL Operators 450
    DISPLAY 451
    ICETOOL Example 451
    OCCUR 451
    RANGE 451
    SELECT 451
    STATS, UNIQUE and VERIFY 451
  INCLUDE and OMIT 447
  INREC and OUTREC 448
  Keyword Statements 443
  Notes 453
  OUTFIL 449
  overview 433
  SORT and MERGE 447
  SUM 447
  Symbol Statements 437
  SYMNAMES DD Statement 436
  SYMNAMES Statements 436
  SYMNOUT DD Statement 436

V

variable-length record
  longest record length 15

variable-length record *(continued)*
record descriptor word 15
VERIFY
efficiency 497
EXEC PARM option 53
installation option 21
OPTION control statement option 170
VERIFY operator (ICETOOL) 422
VIO
ICEMAC installation option 76
installation option 21
VLFILL parameter
OUTFIL control statements option 212
VLLONG
EXEC PARM option 53
installation option 21
OPTION control statement option 170
VLSCMP
EXEC PARM option 54
installation option 21
OPTION control statement option 171
VLSHRT
EXEC PARM option 54
installation option 21
OPTION control statement option 173
VSAM
data set 11
data set considerations 14
E18 user exit 295
E38 user exit 305
E39 user exit 305
key-sequenced data set (KSDS) 14
maximum record size
with INREC control statement 120, 257
user exit functions 286
using RECORD control statement 260
VSAMBSP installation option 21
VSAMEMT
EXEC PARM option 55
installation option 21
OPTION control statement option 174
VSAMIO
EXEC PARM option 55
installation option 21
OPTION control statement option 174

W

Web 3
web site 3
work space
requirements for DFSORT 539
using 539, 548
WRKREL
EXEC PARM option 56
installation option 21
OPTION control statement option 175
WRKSEC
EXEC PARM option 56
installation option 21
OPTION control statement option 176

X

XCTL
using 331
writing macro instructions 343

Y

Y2 formats
description 584
in INCLUDE and OMIT 109, 131
in OUTFIL OUTREC 195
in SORT and MERGE 267
Y2PAST
EXEC PARM option 57
installation option 21
MERGE control statement option 126
OPTION control statement option 176
SORT control statement option 272
Year 2000
century window 176
comparing dates 110
ordering dates 268
transforming dates 195

Z

ZD (zoned decimal) format
description 581
DISPLAY operator 372
INCLUDE statement 96
OCCUR operator 401
OUTFIL statements 198
RANGE operator 407
SELECT operator 411
SORT statement 267
STATS operator 419
SUM statement 274
UNIQUE operator 421
VERIFY operator 423
ZDPRINT
EXEC PARM option 57
installation option 21
OPTION control statement option 176

Readers' Comments — We'd Like to Hear from You

DFSORT
Application Programming Guide
Release 14

Publication No. SC33-4035-20

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC33-4035-20

Cut or Fold
Along Line

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, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
RCF Processing Department
G26/M86 050
5600 Cottle Road
SAN JOSE, CA 95193-0001

Fold and Tape | **Please do not staple** | Fold and Tape

SC33-4035-20

Cut or Fold
Along Line

Program Number: 5740-SM1

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-4035-20