COVER Book Cover

DFSORT/VSE

Application Programming Guide

Version 3 Release 4

Document Number SC26-7040-03

Program Number
5746-SM3

NOTICES Notices

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" in topic FRONT_1.

EDITION Edition Notice

Fourth Edition (May 1998)

This edition replaces and makes obsolete the previous edition, SC26-7040-02. The 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 Version 3 Release 4 of DFSORT/VSE, Program Number 5746-SM3, 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.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. Comments may be addressed to:

- International Business Machines Corporation
- RCF Processing Department
- G26/026
- 5600 Cottle Road
- San Jose, CA, 95193-0001
- U.S.A.

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.

CONTENTS Table of Contents

FIGURES Figures

FRONT_1 Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter 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
- 500 Columbus Avenue
- Thornwood, NY 10594
- USA

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.

Subtopics:

FRONT_1.1 Programming Interface Information

```
This book documents intended Programming Interfaces that allow the
customer to write programs to obtain services of DFSORT/VSE.
```

FRONT_1.2 Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

```
DFSORT                          System/390
ECKD                            VSE/ESA
IBM                             VM/ESA
Language Environment            3090
System/370
```

The following terms are trademarks of other companies:

```
X/Open              X/Open Company Limited
```

PREFACE Preface

This book is intended to help you to sort, merge, and copy files using DFSORT/VSE. This book is not designed to teach you how to use DFSORT/VSE, but is for programmers who already have a basic understanding of DFSORT/VSE, 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/VSE* first. *Getting Started with DFSORT/VSE* is a self-study guide that tells you what you need to know to begin using DFSORT/VSE quickly, with step-by-step examples and illustrations.

Subtopics:

PREFACE.1 About This Book

The various sections of this book present related information grouped according to tasks you want to do. The first three chapters of this book explain what you need to know to invoke and use DFSORT/VSE's primary record-processing functions. The remaining chapters explain more specialized features. The appendixes provide specific information about various topics.

Chapter 1, "Introducing DFSORT/VSE" in topic 1.0, presents an overview of DFSORT/VSE, explaining what you can do with DFSORT/VSE and how you invoke DFSORT/VSE processing. It describes how DFSORT/VSE works, discusses file formats and limitations, and explains the defaults that might have been modified during installation at your site.

Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0, explains how to use job control language (JCL) to run your DFSORT/VSE applications. It explains how to code JOB and EXEC job control statements, and how to define files.

Chapter 3, "Using DFSORT/VSE Program Control Statements" in topic 3.0, presents the DFSORT/VSE control statements you use to sort, merge, and copy data. It explains how to filter your data so you work only with the records you need, and how to edit data by reformatting and summing records. It explains how to write control statements that direct DFSORT/VSE to use your own routines during processing.

Chapter 4, "Using Your Own User Exit Routines" in topic 4.0, describes how to use DFSORT/VSE's program exits to call your own routines during program processing. You can write routines to delete, insert, alter, and summarize records, write or check nonstandard labels, take checkpoints and modify VSAM processing.

Chapter 5, "Invoking DFSORT/VSE from a Program" in topic 5.0, describes how to use system macros to initiate DFSORT/VSE processing from your own assembler program.

Chapter 6, "Using ICETOOL" in topic 6.0, describes how to use the multi-purpose DFSORT/VSE utility ICETOOL. It explains the JCL and operators you can use to perform a variety of tasks with ICETOOL.

Chapter 7, "Improving Efficiency" in topic 7.0, recommends ways you can maximize DFSORT/VSE processing efficiency. This chapter covers a wide spectrum of improvements, from designing individual applications for efficient processing at your site to using DFSORT/VSE features such as dataspace sorting and getvis sorting.

Chapter 8, "Examples of DFSORT/VSE Applications" in topic 8.0, contains annotated example applications for sorting, merging, and copying records.

| Appendix A, "Estimating Storage Requirements" in topic A.0, explains
| virtual storage considerations and how to estimate the amount of intermediate storage you might require when sorting data.

Appendix B, "Data Format Examples" in topic B.0, gives examples of the assembled data formats used with IBM System/370 and IBM System/390.

Appendix C, "EBCDIC and ISCII/ASCII Collating Sequences" in topic C.0, lists the collating sequences from low to high order for EBCDIC and ISCII/ASCII characters.

Appendix D, "DFSORT/VSE Abend Processing" in topic D.0, describes the DFSORT/VSE STXIT routine for processing abends, and the Checkpoint/Restart facility.

Appendix E, "Locales Supplied with C Run-Time Library" in topic E.0, lists the locales supplied with the LE/VSE 1.4 C Run-time library or VSE/ESA 2.2 C Run-time library.

PREFACE.2 Required Publications

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

Short Title	Publication	Order Number
JCL Reference	*VSE/ESA System Control Statements (for VSE/ESA Version 1 Release 3)*	SC33-6513
	VSE/ESA System Control Statements (for VSE/ESA Version 2)	SC33-6613
	VSE/ESA System Utilities (for VSE/ESA Version 1 Release 3)	SC33-6517
	VSE/ESA System Utilities (for VSE/ESA Version 2)	SC33-6617
	VSE/ESA Quick Reference (for VSE/ESA Version 1 Release 3)	GX33-9023
	VSE/ESA Quick Reference (for VSE/ESA Version 2)	GX33-9026

PREFACE.3 DFSORT/VSE Publications

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

Task	Publication	Order Number
Evaluating DFSORT/VSE	*DFSORT/VSE General Information*	GC26-7039
Diagnosing failures and interpreting messages	*DFSORT/VSE Messages, Codes and Diagnosis Guide*	SC26-7132
Planning for, installing, customizing, and tuning DFSORT/VSE	*DFSORT/VSE Installation and Tuning Guide*	SC26-7041
Quick reference	*DFSORT/VSE Reference Summary*	SX26-6008
Learning to use DFSORT/VSE	*Getting Started with DFSORT/VSE*	SC26-7101

You can order a complete set of DFSORT/VSE publications with the order number SBOF-6130, except for *DFSORT/VSE Licensed Program Specifications* (GC26-7038), which must be ordered separately.

Subtopics:

- PREFACE.3.1 DFSORT/VSE Library Softcopy Information

PREFACE.3.1 DFSORT/VSE Library Softcopy Information

A softcopy version of the DFSORT/VSE library is available on the CD-ROM shown in the table that follows. The *IBM Online Library VSE Collection* contains all of the DFSORT/VSE books for Releases 2, 3, and 4, with the exception of the *DFSORT/VSE Reference Summary*, and books from other VSE libraries.

Order Number	Title
SK2T-0060	*IBM Online Library VSE Collection*

PREFACE.4 DFSORT/VSE on the World Wide Web

For news, tips, and examples, visit the DFSORT/VSE home page at URL:

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

PREFACE.5 Related Publications

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

Short Title	Publication	Order Number
General Information	*VSE/ESA General Information (for VSE/ESA Version 1 Release 3)*	GC33-6501
	VSE/ESA General Information - What's NEW (for VSE/ESA Version 2)	GC33-6627
	VSE/ESA General Information - Planning Aspects (for VSE/ESA Version 2)	GC33-6628
Planning	*VSE/ESA Planning (for VSE/ESA Version 1 Release 3)*	SC33-6503
	VSE/ESA Planning (for VSE/ESA Version 2)	SC33-6603
Installation	*VSE/ESA Installation and Service (for VSE/ESA Version 1 Release 3)*	SC33-6504
	VSE/ESA Installation (for VSE/ESA Version 2)	SC33-6604
Administration	*VSE/ESA Administration (for VSE/ESA Version 1 Release 3)*	SC33-6505
	VSE/ESA Administration (for VSE/ESA Version 2)	SC33-6605
Operation	*VSE/ESA Operation (for VSE/ESA Version 1 Release 3)*	SC33-6506
	VSE/ESA Operation (for VSE/ESA Version 2)	SC33-6606

Messages and Codes	*VSE/ESA Messages and Codes (for VSE/ESA Version 1 Release 3)*	SC33-6507
	VSE/ESA Messages and Codes (for VSE/ESA Version 2)	SC33-6607
Extended Addressability	*VSE/ESA Extended Addressability (for VSE/ESA Version 1 Release 3)*	SC33-6524
	VSE/ESA Extended Addressability (for VSE/ESA Version 2)	SC33-6621
VSE/VSAM Library	*VSE/VSAM Commands (for VSE/ESA Version 2)*	SC33-6631
	VSE/VSAM User's Guide and Application Programming (for VSE/ESA Version 2)	SC33-6632
	VSE/VSAM Messages and Codes (for VSE/ESA Version 1 Release 3)	SC24-5146

PREFACE.6 Referenced Publications

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

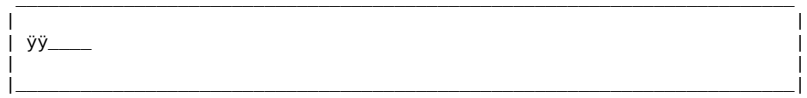
Short Title	**Publication**	**Order Number**
VSE/ESA System Control Statements	*VSE/ESA System Control Statements (for VSE/ESA Version 1 Release 3)*	SC33-6513
	VSE/ESA System Control Statements (for VSE/ESA Version 2)	SC33-6613
VSE/ESA Guide to System Functions	*VSE/ESA Guide to System Functions (for VSE/ESA Version 1 Release 3)*	SC33-6511
	VSE/ESA Guide to System Functions (for VSE/ESA Version 2.	SC33-6611
VSE/ESA System Macros Reference	*VSE/ESA System Macros Reference (for VSE/ESA Version 1 Release 3)*	SC33-6516
	VSE/ESA System Macros Reference (for VSE/ESA Version 2)	SC33-6616
VSE/ESA System Macros User's Guide	*VSE/ESA System Macros User's Guide (for VSE/ESA Version 1 Release 3)*	SC33-6515
	VSE/ESA System Macros User's Guide (for VSE/ESA Version 2)	SC33-6615
VSE/ESA Solving Problems and Diagnosis Tools	*VSE/ESA Guide for Solving Problems (for VSE/ESA Version 1 Release 3)*	SC33-6510
	VSE/ESA Guide for Solving Problems (for VSE/ESA Version 2)	SC33-6610
	VSE/ESA Diagnosis Tools (for VSE/ESA Version 1 Release 3)	SC33-6514
	VSE/ESA Diagnosis Tools (for VSE/ESA Version 2)	SC33-6614
VSE/POWER Library	*VSE/POWER Administration and Operation (for VSE/ESA Version 1 Release 3)*	SC33-6571
	VSE/POWER Administration and Operation (for VSE/ESA Version 2)	SC33-6633
Installation and Tuning	*DFSORT/VSE Installation and Tuning Guide*	SC26-7041
Messages, Codes and Diagnosis	*DFSORT/VSE Messages, Codes and Diagnosis Guide*	SC26-7132

Getting Started	*Getting Started with DFSORT/VSE*	SC26-7101
VSE/VSAM Library	*VSE/VSAM Commands and Macros*	SC33-6532
	VSE/VSAM Space Management SAM Feature User's Guide	SC24-5192
	VSE/VSAM User's Guide	SC33-6535
Assembler Reference	*High Level Assembler Language Reference*	SC26-4940
	Assembler H Version 2 Application Programming: Language Reference	GC26-4037
	OS/VS--DOS/VS--VM/370 Assembler Language Manual	GC33-4010
LE/VSE Library	*IBM Language Environment for VSE/ESA Programming Guide Release 4*	SC33-6684
	IBM Language Environment for V6E/ESA Programming Reference Release 4	SC33-6685
	IBM Language Environment for VSE/ESA Installation and Customization Guide Release 4	SC33-6682
	IBM Language Environment for VSE/ESA Debugging Guide and Run-Time Messages Release 4	SC33-6681
	IBM Language Environment for VSE/ESA C Run-Time Programming Guide Release 4	SC33-6688
	IBM Language Environment for VSE/ESA C Run-Time Library Reference Release 4	SC33-6689

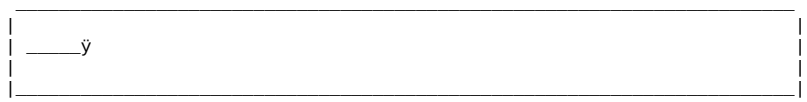
PREFACE.7 Notational Conventions

The syntax diagrams in this book are designed to make coding DFSORT/VSE 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:

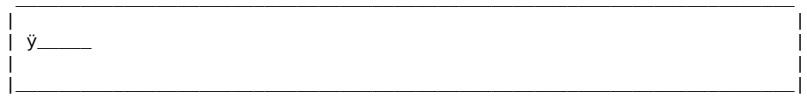
- Read the syntax diagrams from left to right and from top to bottom.
- Begin coding your statement at the spot marked with the double arrowhead.



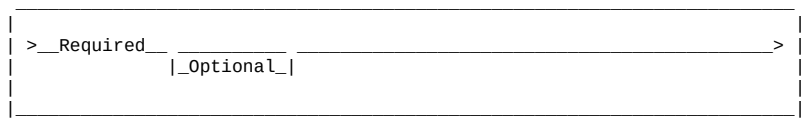
- A single arrowhead at the end of a line indicates that the diagram continues on the next line or at an indicated spot.



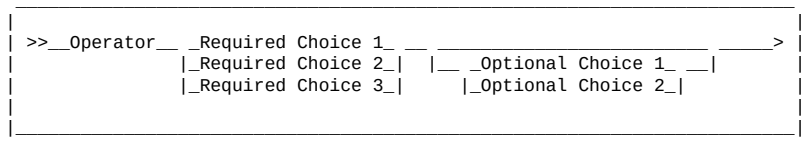
A continuation line begins with a single arrowhead.



- Strings in upper-case letters are operators and parameters, and must be coded exactly as shown. (The conventions require that at least one blank separates the initial operator from the succeeding parameters; no blanks are allowed between parameters.)
- Punctuation (parentheses, commas, and so on), must be coded exactly as shown.
- Strings in all lowercase letters represent information that you supply.
- Required parameters appear on the same horizontal line (the main path) as the operator, while optional parameters appear in a branch below the main path.

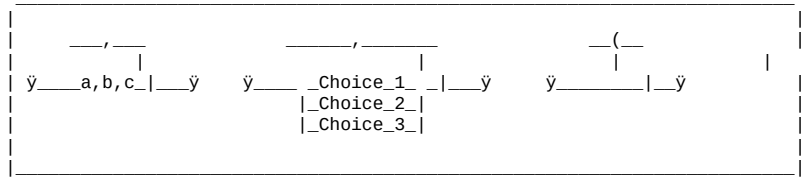


° Where you can make one choice between two or more parameters, the alternatives are stacked vertically.



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

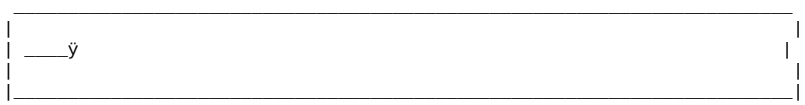
° The repeat symbol shows where you can return to an earlier position in the syntax diagram to specify a parameter more than once (see the leftmost example below), to specify more than one choice at a time from the same stack (see the center example below), or to nest parentheses (see the rightmost example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



Do not interpret a repeat symbol to mean that you can specify incompatible parameters.

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.



FRONT_2 Summary of Changes

Subtopics:

- FRONT_2.1 Fourth Edition, May 1998

FRONT_2.1 Fourth Edition, May 1998

Subtopics:

- FRONT_2.1.1 New Programming Support for Release 4

FRONT_2.1.1 New Programming Support for Release 4

DFSORT/VSE Version 3 Release 4 continues the strategy of providing performance improvements and productivity features. These improvements and features are described in more detail in the subsections that follow.

Subtopics:

- FRONT_2.1.1.1 Performance
- FRONT_2.1.1.2 Productivity
- FRONT_2.1.1.3 Additional Enhancements

FRONT_2.1.1.1 Performance

Performance enhancements for DFSORT/VSE Version 3 Release 4 include the following:

- Improved data processing methods for:
  - Dataspace and getvis sorting applications using work space
  - Merge and copy applications

- Improved input/output processing techniques for:
  - SAM output files
  - Non-VSAM input and output files
  - VSAM (and SAM ESDS accessed as VSAM) input and output files
  - Work files
- Improved ECKD disk device support for input, output, and work files by using the ECKD command set.
- New VSAMBSP installation option which allows users to control the number of buffers DFSORT/VSE can use for VSAM (or SAM ESDS accessed as VSAM) input and output file processing.
- Improved work file processing:
  - All work files are now closed at the end of an application.
  - Additional work file extents can now be used, if available, when end of extent is encountered regardless of whether STXIT is in effect.
  - All extents of an SD work file can now be used instead of only the first extent.

FRONT_2.1.1.2 Productivity

Additional Year 2000 Formats: New formats give users more flexibility in sorting, merging, or transforming two-digit year dates:

- Y2S interprets two-digit character year data according to the century window and allows special handling of indicators X'00' (binary zeros), X'40' (EBCDIC blanks), X'20' (ASCII blanks) and X'FF' (binary ones) in the year field.
- Y2B interprets two-digit binary year data according to the century window.

OUTREC Enhancements: The OUTREC control statement supports the following new features:

- 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).

INCLUDE/OMIT Enhancements: The following INCLUDE/OMIT enhancements are supported:

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

- ALL and NONE allow users to include or omit all records.

ZDPRINT Option: With the new ZDPRINT installation and run-time options, users can choose to have summed (totalled) positive zoned decimal fields converted to printable numbers.

Online Message Explanations Support: New Online Message Explanations (OME) allow users to request an explanation of a DFSORT/VSE message. The message explanation is displayed on the console.

FRONT_2.1.1.3 Additional Enhancements

The following additional enhancements are supported:

- The IBM-supplied default has been changed from STXIT=YES to STXIT=MIN. The STXIT=MIN installation option and the MINSTXIT run-time option allow users to specify that DFSORT/VSE should use its STXIT routine for abend recovery processing, **not** restoring its STXIT every time control is returned from a user exit routine. Unlike STXIT=YES (or STXIT), STXIT=MIN (or MINSTXIT) does not degrade performance when COBOL or PL/I programs invoke DFSORT/VSE and use E15/E35 user exit routines to process records.

- The DIAGINF installation option provides a new DFSORT/VSE capability that allows users to request diagnostic information (diagnostic messages and a dump), regardless of the options in effect at run time.

- The new NRECOUT installation and run-time option allows users to specify the action DFSORT/VSE should perform when it does not write any records to the output file. This gives users control over the action (continue or terminate), type of message (informational or error), and return code (0,4 or 16) when no records are written to the output file.

1.0 Chapter 1. Introducing DFSORT/VSE

Subtopics:

1.1 DFSORT/VSE Overview

This chapter introduces IBM DFSORT/VSE Licensed Program 5746-SM3. DFSORT/VSE 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 files into one.
- When you *copy* records, you make an exact duplicate of each record in your file.

You can sort records from up to nine input files to one output file.

Merging records first requires that the input files 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 records from up to nine input files to one output file.

You can copy records from up to nine input files to one output file.

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

You can control which records to keep in the final output file of a DFSORT/VSE run by using INCLUDE and OMIT control statements in your application. These control statements work like filters, testing each record against criteria that you supply and retaining only the ones you want for the output file. For example, you might choose to 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 control statements. Use INREC and OUTREC control statements to delete fields from your records, to rearrange the order of the fields within records, and to insert separators (that is, blanks or binary zeros) before, between, or after fields. For example, you might want to create a file containing financial data but without any of the names that were there originally.

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

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

You can direct DFSORT/VSE 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 (DFSORT/VSE already provides extensive editing capabilities through the INCLUDE, OMIT, INREC, OUTREC, and SUM control statements). For example, you can write your own routines to open and close files.

1.2 Invoking DFSORT/VSE

You can invoke DFSORT/VSE processing in the following ways:

- With an EXEC job control statement in the input stream using SORT as the program name. See Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0.
- With a program written in assembler language using a system macro. See Chapter 5, "Invoking DFSORT/VSE from a Program" in topic 5.0.
- With programs written in either COBOL, PL/I or RPG II with the Auto-Report Feature. See the programmer's guide describing the compiler version

available at your location.

- With the ICETOOL utility. See Chapter 6, "Using ICETOOL" in topic 6.0.

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

1.3 How DFSORT/VSE Works

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

Subtopics:

1.3.1 Operating Systems

DFSORT/VSE runs on any IBM processor supported by all releases of:

- VSE/ESA Version 1
- VSE/ESA Version 2

DFSORT/VSE runs under VSE/ESA when VSE/ESA is running in a virtual machine as a VM/ESA guest.

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

1.3.2 Control Fields and Collating Sequences

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

DFSORT/VSE 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/VSE to identify and sort, or merge the records. For all practical purposes, you can think of keys as equivalent to the control fields DFSORT/VSE 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/VSE 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/VSE to retain the original input order for records with identical values for all control fields by specifying EQUALS option.

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, their total lengths must not exceed 3072 bytes (see Figure 1), and the total number of control fields is limited to 64.

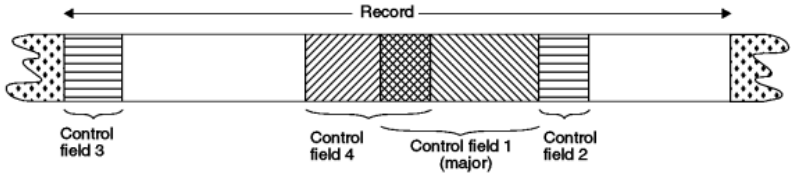


Figure 1. Control Fields.

If any of the above restrictions is violated, DFSORT/VSE issues an error message and terminates.

DFSORT/VSE 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/VSE 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/VSE 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. Packed decimal, zoned decimal, fixed-point, normalized floating-point, and the signed numeric data formats are collated numerically; that is, each value is interpreted as having a positive or negative sign.

You can modify the standard EBCDIC sequence to collate differently if, for example, you want to allow alphabetic collation of national characters. An alternate collating sequence can be defined during installation with the ILUINST ALTSEQ option, or you can define it yourself at run-time with the ALTSEQ control statement.

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

1.3.3 Cultural Environment Considerations

DFSORT/VSE's collating behavior can be modified according to your cultural environment. Your cultural environment is defined to DFSORT/VSE 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 multicharacter 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/VSE's SORT, MERGE, INCLUDE, and OMIT processing as follows:

- Sort and Merge

  DFSORT/VSE 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/VSE 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/VSE 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.

1.3.4 DFSORT/VSE Processing

You must prepare job control language (JCL) statements and DFSORT/VSE program control statements to invoke DFSORT/VSE processing. Job control statements (see Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0) are processed by your operating system. They describe your files to the operating system and initiate DFSORT/VSE processing. DFSORT/VSE program control statements (see Chapter 3, "Using DFSORT/VSE Program Control Statements" in topic 3.0) are processed by DFSORT/VSE. They describe the functions you want to perform and invoke the processing you request.

| DFSORT/VSE uses virtual storage (partition program area, GETVIS area, or
| data space) to sort records. When all input records can be processed in
| virtual storage, an *incore sort* is performed.

A sort application can require intermediate storage (work files) as working space during a DFSORT/VSE run. Work space must be on disk devices. You must use job control statements to indicate the intermediate storage devices to use, and the amount of work space required for any sort application. Methods for determining the amount of space required are explained in Appendix A, "Estimating Storage Requirements" in topic A.0. Merge and copy applications do not require intermediate storage.

Figure 2 illustrates the processing order for record handling, user exit routines, control statements, and options. Use this diagram with the text following it to understand the order DFSORT/VSE uses to run your application.

```
          Sorting                          Merging or Copying
```

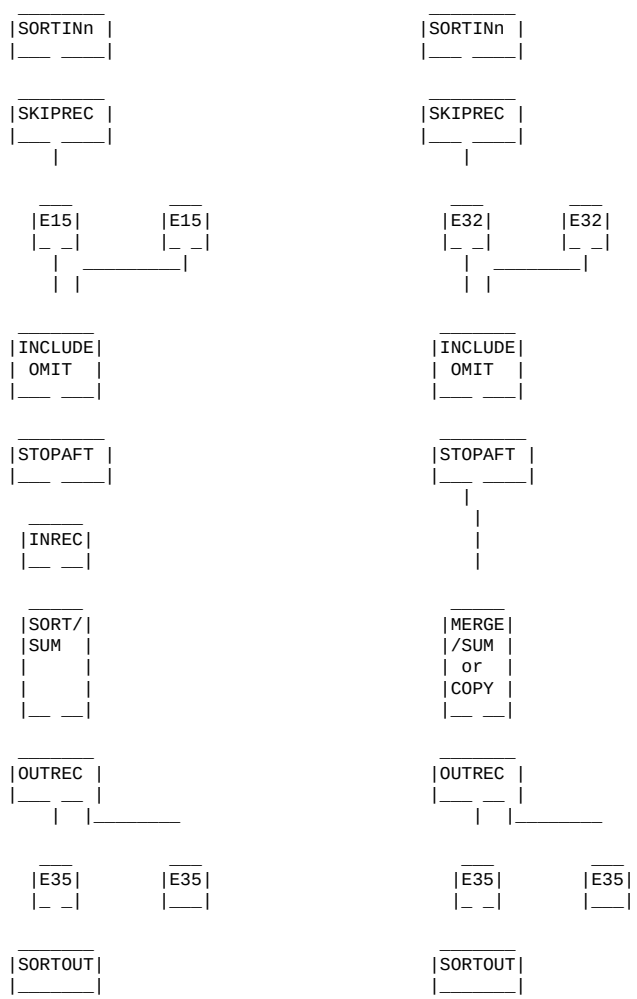


Figure 2. Record Processing Order

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

1. DFSORT/VSE first checks whether or not you supplied an EXIT operand in the INPFIL control statement for a sort, merge, or copy application. If not, DFSORT/VSE reads the input records from SORTINn files.
   - If an EXIT operand is present for a sort application, DFSORT/VSE does not read the input records from SORTINn files and you must use an E15 user exit routine to insert all the records.
   - If an EXIT operand is present for a merge or copy application, DFSORT/VSE does not read the input records from SORTINn files and you must use an E32 user exit routine to insert all the records.
2. If the input records for a sort or copy application are read from SORTINn files, DFSORT/VSE performs processing specified with the SKIPREC option. DFSORT/VSE skips records without processing until the SKIPREC count is satisfied. Eliminating records before sort or copy processing gives better performance.
3. If the input records for a sort application are read from SORTINn files, DFSORT/VSE checks whether or not you specified an E15 user exit. If so, DFSORT/VSE transfers control to the user exit routine. The E15 user exit routine can insert, delete, or reformat records.

4. If the input records for a merge or copy application are read from SORTINn files, DFSORT/VSE checks whether or not you specified an E32 user exit. If so, DFSORT/VSE transfers control to the user exit routine. The E32 user exit routine can only replace the records.

5. DFSORT/VSE performs processing specified with the INCLUDE or OMIT function. If you used an E15 user exit routine to modify the record format, the INCLUDE or OMIT control field definitions you specify must apply to the current format rather than to the original one. If you use the INCLUDE or OMIT function to delete unnecessary records before sort, merge, or copy processing, your applications run more efficiently.

6. For a sort or copy application, DFSORT/VSE performs processing specified with the STOPAFT option. DFSORT/VSE stops accepting the records for processing after the maximum number of records you specified has been reached. DFSORT/VSE accepts records for processing if they are:

    - Read from SORTINn files, inserted by the E15 (for a sort application) or E32 (for a merge or copy application) user exit routine
    - Not deleted by a SKIPREC option
    - Not deleted by the E15 user exit routine (for a sort application)
    - Not deleted by an INCLUDE or OMIT function

7. DFSORT/VSE performs processing specified with an INREC function. If you changed the 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. Use the INREC function to shorten records before further processing to gain better performance.

8. DFSORT/VSE performs processing specified in the SORT or MERGE control statement.

    - For a sort application, all input records are processed before any output record is processed.
    - For a merge or copy application, an output record is processed after an input record is processed.
    - For a sort or merge application, if a SUM control statement is present, DFSORT/VSE processes it during the sort or merge processing. DFSORT/VSE summarizes the records and deletes duplicates as soon as possible to gain better performance. If you made any changes to the record format prior to this step, the SORT or MERGE and SUM control field definitions you specified must apply to the current format rather than to the original one.

9. DFSORT/VSE performs processing specified with the OUTREC function. If you changed the 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.

10. For a sort application, DFSORT/VSE transfers control to the E35 user exit routine after processing of all input records is completed. For a merge or copy application, DFSORT/VSE transfers control to the E35 user exit routine after processing of each input record is completed. If you changed the record format prior to this step, the E35 user exit routine receives the records in the current format rather than in the original one. You can use the E35 user exit routine to insert, delete, reformat, or sum records.

11. If an EXIT operand in the OUTFIL control statement is present, the E35 user exit routine must dispose of all the records because DFSORT/VSE treats these records as deleted.

12. DFSORT/VSE writes records to the SORTOUT file, if present.

1.4 File Considerations

You must define any files you provide for DFSORT/VSE according to the conventions your operating system requires. You can use the checkpoint/restart and label checking facilities of the operating system during DFSORT/VSE processing.

Subtopics:

1.4.1 Input and Output Files

The input and output files can be VSAM or SAM. The output file need not be the same as the input file--that is, you can have a SAM input file and a VSAM output file or vice versa, but, unless the SAM files are SAM ESDS, the input files must be either all SAM or all VSAM.

The files can reside on tape, count-key-data (CKD), extended-count-key-data (ECKD), or on fixed block architecture (FBA) disks.

Output can be written to printers and punch devices.

Whenever CKD disks are referred to in this book, it also includes ECKD disks.

The characteristics of the input and output files that DFSORT/VSE can handle are given in Figure 3.

Figure 3. Input and Output File Characteristics

Characteristics	Sort input	Merge (M) Input Copy (C) Input	Output
Files			
Type of extent	Type 1 and 8	Type 1 and 8	Type 1 and 8
Organization(1)	SAM or VSAM; cannot be mixed	SAM or VSAM; cannot be mixed (M)	SAM or VSAM
Number(2)	1-9 files	1-9 files (M) 1-9 files (C)	1 file (can be multivolume)
Size	No restriction	No restriction	No restriction
Blocking	Blocked(3) or unblocked	Blocked(3) or unblocked	Blocked or unblocked(4)
Block size	Can differ; biggest must be specified	Can differ; biggest must be specified (M)	
Contents	Unsorted or sorted records; or can be empty	Previously sorted records; or can be empty In addition, can be unsorted (C)	Sorted records (possibly modified or summarized), or addresses of records (with or without control fields)
Labels	Any or none; can be mixed	Any or none; can be mixed (M)	Any or none
Pooling(5)	Can be pooled with output	Cannot be pooled	Can be pooled with sort input
Records			

Format	Fixed or variable; cannot be mixed	Fixed or variable; cannot be mixed (M)	Fixed or variable
Code	EBCDIC or ISCII/ASCII; cannot be mixed	EBCDIC or ISCII/ASCII; cannot be mixed (M)	Same as input
Control Fields			
Number	1-64	1-64 (M)	
Format	Formats can be mixed	Formats can be mixed (M)	
Order	Ascending, descending, or both(6)	Ascending, descending, or both (M)(6)	Output to a VSAM KSDS file must be in primary key sequence
Total lengths, maximum	3072 bytes	3072 bytes (M)	
Location	Must be all within first 4092 bytes of record	Must be all within first 4092 bytes of record (M)	

Notes:

(1) SAM ESDS files may be mixed with SAM or VSAM files but not both at once.

(2) If on tape, each unit may have as many alternates as permitted by VSE/ESA.

(3) With fixed-length records, short blocks are accepted if their length is a multiple of record length.

(4) For output to a printer or punch device, the output records must be unblocked.

(5) VSAM input and output can be the same file only if the file was defined with the REUSE attribute and the REUSE operand is specified in the OUTFIL control statement.

(6) When records with equal control fields are sorted or merged, their output order is unpredictable if the EQUALS option is not in effect.

1.4.2 SAM ESDS Files

SAM ESDS files always need the VSAM operand in their DLBL job control statements. If the VSAM or ESDS operand is specified in the INPFIL or OUTFIL control statement, as appropriate, SAM files are accessed as VSAM. Otherwise, they are accessed as SAM.

SAM ESDS files in control interval format can be accessed as either SAM or VSAM. As input, however, if they are mixed with ordinary SAM files, they must be accessed as SAM files; if mixed with VSAM files, they must be accessed as VSAM files. If SAM ESDS files are not in control interval format, they cannot be accessed as VSAM by DFSORT/VSE directly, but they can be read or written using E15 and E35 user exit routines.

1.4.3 Record and Data Format

Records can be either fixed (FLR) or variable (VLR) length. Variable-length spanned records are permitted; they exclude the use of the ADDROUT or ADDROUT=D option.

The record data can be numeric or alphanumeric coded EBCDIC (or ISCII/ASCII for tape input and output files).

1.4.4 Record Length and Block Size

See Figure 4 for record length; see Figure 5 for block size.

The maximum *block size* for input and output files is track capacity for EBCDIC data and 9999 bytes for ISCII/ASCII data.

There are two restrictions on the use of large blocks:

- | The larger the blocks, the more virtual storage your sort, merge, or copy application will need.
- For CKD devices, you cannot define an output block size larger than track capacity for the output device you have specified.

Figure 4. Record Length

Device	Minimum Length (in bytes)		Maximum Length (in bytes)	
	FLR	VLR	FLR	VLR
Tape input	12	12	32767	32767(1)
Tape output	18	18	32767	32767(1)
CKD input/output	1	5	Track Capacity	Track Capacity
FBA input/output	1	5	32761	32767(1)
SAM ESDS input/output	1	5	32761	32767(1)
VSAM input/output	1	1(2)	32767(1)	32767(1)
Print output	1	5	120	124
Punch output	1	5	80	84

Notes:

(1) Implies spanned records.

(2) You must add four bytes when specifying the length, because DFSORT/VSE adds a record descriptor word (RDW).

Input *records* cannot be larger than input block sizes; additional restrictions are:

- If you are using any CKD work devices, the input record must not be larger then the track capacity of the work device. If work devices are mixed, the shortest track capacity is the limiting factor.
- FBA work devices limit the input record length to 32767 bytes.
- The record length must not exceed the maximum length specified in the RECORD control statement.
- With variable-length records, the 4-byte record descriptor word is regarded as part of the record. Also, maximum record length is four bytes less than block size because of the block descriptor word.
- For output to a printer or punch device, the output records must be unblocked.

Figure 5. Block Size

Device	Minimum Block Size (in bytes)	Maximum Block Size (in bytes)
9345	1	46456
3390	1	56664
3380	1	47476
3375	1	35616
FBA	1	32761
Tape input	12	32767
Tape output	18	32767
Print output	1	124
Punch output	1	84

Note: If there are checkpoint records mixed with the tape input file, the minimum block size is 16 bytes.

1.5 Input/Output Devices

VSAM or SAM ESDS input/output files can reside on any disk devices supported by your release of VSAM, whether CKD or FBA. SAM files can reside on any of the following, if supported by your operating system:

- IBM 3375, 3380, 3390, or 9345 CKD direct access storage devices
- IBM RAMAC Array DASD
- IBM 3400 tape devices
- IBM 3590 Tape Subsystems
- All types of IBM FBA direct access storage devices, including virtual (FBAV) disks

Input files can be on a mixture of any of the permitted device types.

Output can be written to printers and punch devices.

Subtopics:

1.5.1 Input/Output Pooling

You can design a sort application in which two of the input/output files share the same disk extents or tape units. This allows you to run your sort application with fewer devices and is known as I/O pooling. This means that the output file will be written over the input file, so it is important to check that you have specified the files correctly. The rules for I/O pooling are:

- Sort input and output files can be pooled.
- Any file name can be used.
- Files can be multivolume and multiextent.
- Merge and copy files must not be pooled.
- Sort work files must not be pooled with any other sort file. If they are, an error message is produced and DFSORT/VSE terminates.

1.5.2 Input Device Sharing

In a sort application, several tape input files can use the same device. The files are read serially. Each tape can be demounted after it has been read so that a tape containing another file can be mounted. This is not I/O pooling, because only the input function is involved. However, like I/O pooling, it does increase the capacity of a given number of devices.

1.6 Intermediate Storage Devices

If intermediate storage is needed for your sort application, it must be on a disk device. The types of devices available for intermediate storage are listed in Appendix A, "Estimating Storage Requirements" in topic A.0.

DFSORT/VSE allows the work files to be allocated on two different device types, where device type is considered to be the same if track capacity and the number of tracks per cylinder are the same.

Notes:

1. Tape units cannot be used for intermediate storage.
2. Methods for determining the amount of space required are explained in Appendix A, "Estimating Storage Requirements" in topic A.0.

1.7 Label Processing

If your files have standard labels or are unlabeled, then DFSORT/VSE will take care of all opening, closing, and label handling for you. If you have nonstandard labels (or extra headers or trailers in addition to standard labels), you must use your own user exit routines to carry out label processing, as described in Chapter 4, "Using Your Own User Exit Routines" in topic 4.0.

Standard Labels:
Standard direct access and tape labels are processed when DFSORT/VSE opens and closes the files.

Unlabeled Tapes:
Unlabeled input and output files are processed by DFSORT/VSE: no user programming is required. Unlabeled output files are normally preceded and followed by a tape mark, but the leading tape mark can be eliminated by specifying the NOTPMK operand in the OUTFIL control statement.

Nonstandard Direct Access Labels:
DFSORT/VSE will not attempt to use the first track of the first extent in any CKD input or output file specified by the user as having nonstandard labels. This track may be handled in any way the user chooses by routines at E11 and E31 user exits. With an FBA file, the amount of space left for label processing depends on the control interval size, as described in Chapter 4, "Using Your Own User Exit Routines" in topic 4.0.

Nonstandard Tape Labels:
If labels are not standard, or if the standard tape labels include additional header and trailer labels or user header and trailer labels, the user must assume the responsibility for processing these extra labels.

All such nonstandard tape label processing must be done at the appropriate label-processing user exits (E11, E31, E17, and E37) where the files are also opened and closed.

For detailed information about label processing, you should refer to the following publications:

- *VSE/ESA System Control Statements*
- *VSE/ESA Guide to System Functions*

1.8 Passwords for VSAM Files

DFSORT/VSE allows the use of password-protected VSAM files for input and output. When a password is needed for a VSAM file, the appropriate E18 user exit routine is entered for sort input, E38 user exit routine for merge or copy input, and E39 user exit for sort, merge or copy output. With these user exit routines, the user has the opportunity to supply the required password(s). If a password is not supplied by a user exit routine, VSAM prompts the operator to supply the password. E18, E38, and E39 user exit routines are only available for SAM ESDS files if they are accessed as VSAM.

1.9 Installation Defaults

When your system programmers installed DFSORT/VSE, they selected (with ILUINST macro) a set of installation parameters to be used by default for DFSORT/VSE applications. The selected defaults can affect the way your applications run, and in many cases can be overridden by specifying the appropriate run-time parameters. This book assumes that DFSORT/VSE was installed at your site with the defaults set when the product was delivered.

You can use the DEFAULTS operator of the multi-purpose DFSORT/VSE utility ICETOOL to list the installation defaults actually in use at your site and the IBM-supplied defaults they override, where appropriate. See Chapter 6, "Using ICETOOL" in topic 6.0 and "DEFAULTS Operator" in topic 6.6 for more information on using ICETOOL and the DEFAULTS operator.

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

Parameter Function

ALTSEQ
Alters the normal EBCDIC collating sequence.

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

DIAG
Specifies whether diagnostic messages are produced.

| DIAGINF
Specifies whether DFSORT/VSE should always produce diagnostic
| information for all jobs or for a specific job. Diagnostic
| information consists of diagnostic messages and a dump if an
| abend occurs.

DSPSIZE
Specifies the maximum amount of data space to use for dataspace sorting.

| DUMP
Specifies whether a dump of virtual storage is produced when DFSORT/VSE terminates abnormally.

EQUALS
Specifies whether the order of records that collate identically is preserved from input to output.

ERASE
Specifies whether the work files are erased.

FMS
Specifies whether DFSORT/VSE will attempt to interact with the File Management System installed at your site.

GVSIZE
Specifies the maximum amount of GETVIS area to use for getvis sorting.

GVSRANY
Specifies the amount of 31-bit GETVIS area to be reserved for the operating system and user application when the GVSIZE=MAX option is in effect.

GVSRLOW
Specifies the amount of 24-bit GETVIS area to be reserved for the operating system and user application when the GVSIZE=MAX option is in effect.

LOCALE
Specifies whether locale processing is to be used and, if so, designates the active locale.

| NRECOUT
Specifies the actions to be taken by DFSORT/VSE when it does
| not write any records to the output file.

PRINT
Specifies which messages are to be produced by DFSORT/VSE.

ROUTE
Specifies where messages are to be routed by DFSORT/VSE.

SORTIN
Specifies the logical unit numbers of the input files.

SORTOUT
Specifies the output device.

| **STORAGE**
Specifies the maximum amount of virtual storage of the
| partition program area to be used, if available, by DFSORT/VSE.

STXIT
Specifies whether DFSORT/VSE is to use its STXIT routine.

VERIFY
Specifies whether direct access output blocks are verified as they are written.

| **VSAMBSP**
Specifies the number of buffers DFSORT/VSE can use for VSAM
| (or SAM ESDS accessed as VSAM) input and output file
| processing.

| **WRKSEC**
Specifies whether the secondary allocation for SAM ESDS or SD
| work files is used.

Y2PAST
Specifies the sliding or fixed century window.

| **ZDPRINT**
Specifies whether positive zoned decimal (ZD) fields resulting
| from summation must be converted to printable numbers.

1.10 DFSORT/VSE Messages and Return Codes

You can determine, during installation or run time, whether DFSORT/VSE writes messages to the SYSLST file, to the system console, or to both.

Messages written by DFSORT/VSE can be critical error messages, informational messages, or diagnostic messages, as determined during installation or run time.

| See *Messages, Codes and Diagnosis* for complete information about
| DFSORT/VSE messages. For successful completion, DFSORT/VSE passes back a
| return code of 0 or 4 to the operating system or the invoking program.

| For unsuccessful completion DFSORT/VSE passes back a return code of 16 or
| 20 to the operating system or the invoking program.

| The meanings of the return codes that DFSORT/VSE passes back are:

| **0**
Successful completion. DFSORT/VSE completed successfully.

| **4**
Successful completion. DFSORT/VSE completed successfully;
| NRECOUT=RC4 was in effect and DFSORT/VSE did not write any records
| to the output file.

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

| **20**
Unsuccessful completion. DFSORT/VSE failed to load the message
| phase into partition GETVIS area.

| The return code is available to both directly invoked and program invoked
| DFSORT/VSE applications.

| For directly invoked applications, the return code is passed in general
| register 15 using the VSE/ESA EOJ macro. For more details see *VSE/ESA*
| *Guide to System Functions*.

| For program invoked applications, the return code is available in a user
| specified area addressed by the DFSORT/VSE parameter list.

2.0 Chapter 2. Invoking DFSORT/VSE with Job Control Language

Subtopics:

2.1 Using the JCL

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

- Identify you as an authorized user.
- Allocate the necessary resources to run your application.
- Run your application.
- Return information to you about the results.
- Terminate your application.

The job control statements (JCS) and job control commands (JCC) and their functions are described briefly below. For a complete discussion of job control statements and commands and their formats, refer to *VSE/ESA System Control Statements*.

JCS/JCC Description

JOB
The JOB job control statement is the first statement of your application. All parameters in the operand field are optional, although your site might have made information such as account number and the name of the programmer mandatory.

ASSGN
ASSGN job control statements are required only if the devices to be used in an application have not previously been assigned to the appropriate symbolic names (SYS numbers) used in the DFSORT/VSE application. ASSGN statements are not required for VSAM or SAM ESDS files.

TLBL
A TLBL job control statement is required for every tape file with standard labels.

DLBL
A DLBL job control statement is required for every disk file. Requirements are as follows:

- The BLKSIZE parameter must not be specified.
- If any file is on the same disk as another file used by the program, the two files must have different file IDs.
- For a VSAM file, the file ID must specify the cluster name.

EXTENT
One EXTENT job control statement is required for each direct access area to define the limits which will be used by the program. Extents can be Type 1 or Type 8 for input/output files and must be Type 1 for work files. The defined extents must include the SYS number of the device containing the extent. EXTENT job control statements are not required for VSAM or SAM ESDS files, unless they are to be implicitly defined (SAM ESDS files only).

EXEC
The EXEC job control statement must be followed by DFSORT/VSE program control statements. The EXEC job control statement must contain the name of the DFSORT/VSE call program (SORT) and should contain a SIZE parameter if storage size is not specified elsewhere (SIZE job control statement, STORAGE option in the OPTION program control statement) or defaulted, and if a GETVIS area is required, when:

- DFSORT/VSE is to use getvis sorting.
- DFSORT/VSE is to use VSAM files.
- EXEC REAL is specified.

ALLOC
An ALLOC job control command defines a virtual partition size.

SIZE

A SIZE job control command defines the size of a partition program area.

SYSDEF
A SYSDEF job control statement defines the size of a data space.

RSTRT
A RSTRT job control statement is required to continue execution of an interrupted job from a checkpoint record.

The format of the job control statements is:

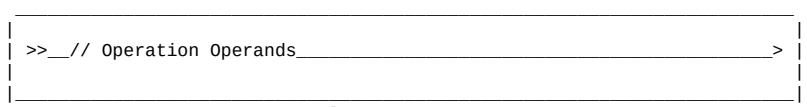


Figure 6. Format of job control statements

The format of the JOB job control statement is:

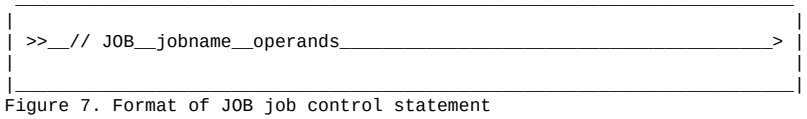


Figure 7. Format of JOB job control statement

The format of the EXEC job control statement is:

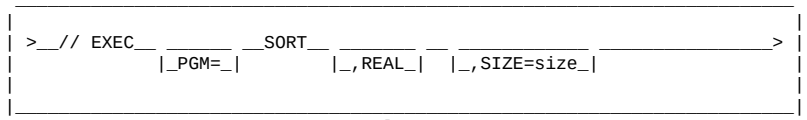


Figure 8. Format of EXEC job control statement

2.2 Defining Files

All files that are to be used in a sort, merge, or copy application must be defined according to VSE/ESA standard.

- Except for VSAM or SAM ESDS files, the file SYS number must be assigned to a device address (ASSGN job control statement).
- If the SYS number default values shown in Figure 9 (or changed for installation) are not to be used, the values must be specified in the appropriate

SORTIN or SORTOUT operand of the OPTION program control statement.

- The file name must be included in the DLBL or TLBL job control statement.
- If the default file names are not to be used, the file names must be specified in the FILNM operand of the OPTION program control statement.
- Except with VSAM or previously defined SAM ESDS files, at least one EXTENT job control statement is required for each DLBL job control statement.
- If input is from an FBA device, the DLBL job control statement must include a value with the CISIZE parameter.
- If output is to an FBA device, the DLBL job control statement should include a value with the CISIZE parameter. If it does not, DFSORT/VSE will use the minimum valid value that will hold the specified or defaulted block size in the OUTFIL program control statement.
- If a value on the CISIZE parameter is specified for a work file, it is ignored.
- DISP=(NEW,DELETE) should be specified in the DLBL job control statement for SAM ESDS work files.
- VSAM files and SAM ESDS files should be previously defined using the VSAM Access Method Services program. Only SAM ESDS output and work files can be defined implicitly with RECORDS and RECSIZE parameters in the DLBL job control statement. This will, of course, add to the overall time taken by DFSORT/VSE. See *VSE/VSAM Library* for more information.

  Implicitly defined SAM ESDS output files will be defined with record format undefined (RECFM=UNDEF). These files can be read using SAM as either fixed or variable, as appropriate. If these files are read using VSAM, the access method will provide an entire block to the reading program, which must then do its own deblocking. If this is not desired, these files must be defined explicitly with RECFM=FB or RECFM=VB as required.

If a File Management System is installed at your site and the FMS installation option is in effect, then input, output, and work files must be defined according to the requirements of the File Management System. In this case, DFSORT/VSE will attempt to interact with the File Management System installed at your site:

- When an input, output or work file is not completely defined, DFSORT/VSE will try to continue processing by assuming that file definition will be dynamically resolved at open time.
- If the allocated work file space is exhausted, DFSORT/VSE will try to allocate a secondary extent dynamically.

If the FMS installation option is in effect, DFSORT/VSE will attempt to take advantage of the benefits provided by the File Management System installed at your site, such as:

- Dynamic logical and physical device assignment
- Dynamic primary and secondary extent allocation

File default names are shown in Figure 9. If a default SYS number is occupied by another file (as specified in the OPTION program control statement), DFSORT/VSE will use the next free number.

Figure 9. File Names and SYS Numbers Allocated by Default

		Symbolic Unit Name

File Type	File Name	DFSORT/VSE Reads Input and Writes Output	E15 User Exit Routine Reads Input	E35 User Exit Routine Writes Output	E15 and E35 User Exit Routines Read Input and Write Output
Output	SORTOUT	SYS001	SYS001		
Input	SORTIN1 . . . SORTIN9	SYS002 . . . SYS(n+1)		SYS001 . . . SYS(n)	
Work	SORTWK1 . . . SORTWK9	SYS(n+2) . . . SYS(n+m+1)	SYS002 . . . SYS(m+1)	SYS(n+1) . . . SYS(n+m)	SYS001 . . . SYS(m)
Check-point	SORTCKP	SYS000	SYS000	SYS000	SYS000

Notes:

1. Where:

   n = the number of input files, as specified in the FILES operand of the SORT or MERGE program control statement. The maximum value is 9.

   m = the number of work files, as specified in the WORK operand of the SORT program control statement. The maximum value is 9.

2. Users of COBOL, PL/I, and RPG II should refer to their respective programming guides for further information about specifying symbolic unit names.

3. The values in Figure 9 are only needed for tape input and output files and for checkpoint file.

4. If you want to use standard labels for your work files and you have both directly and program invoked sort applications, use a SYS number in the EXTENT job control statement that is not normally used within the sort application. This allows both directly and program invoked sort applications to use only one label for each work file. A program invoked sort application normally uses SYS001 through SYS009 for work files, while a directly invoked sort application uses SYS003 through SYS011.

5. If the defaults for the ILUINST macro parameters SORTIN and SORTOUT are changed, the SYS numbers given in Figure 9 can also change.

Subtopics:

2.2.1 Input File Statements

When the file name is of the form SORTINn, n can be any value from 1 to 9 for a sort, merge, or copy application, depending on the number of input files. The file ID of the input file to be read must be included on each TLBL or DLBL job control statement. When the input file is a direct access multiextent file, only the first EXTENT job control statement need contain the specified or defaulted SYS number for the input file. Other EXTENT job control statements can each specify any valid SYS number.

2.2.2 Output File Statements

An output file on disk is defined by SORTOUT DLBL and EXTENT job control statements. Multivolume or multiextent output (or both) for a disk file is accomplished by use of VSE/ESA standards: *one* DLBL job control statement is supplied for the entire file, followed by one EXTENT job control statement for *each* separate extent that the file occupies on disk. When the output file is a direct access multiextent file, only the first EXTENT job control statement need contain the specified or defaulted SYS number. Other EXTENT job control statements can each specify any valid SYS number.

2.2.3 Work File Statements

Work files are defined by SORTWKn DLBL and EXTENT job control statements.
| The files can be defined as SD, DA, or SAM ESDS files, if supported by the
| operating system. There can be up to nine work files. The number of work files must be specified in the WORK operand of the SORT program control statement.

Users of SAM ESDS files should use the special file-ID prefix to cause a single extent to be allocated for the primary allocation, and specify DISP=(,DELETE). If the primary extent is filled, a second extent will be allocated. After a secondary allocation, its size will be near 500×80 bytes. See *VSE/VSAM Library*.

Any valid SYS number which correctly defines the device can be used for sort work files.

Subtopics:

- 2.2.3.1 Example

2.2.3.1 Example

```
// ASSGN SYS005,X'191'
// ASSGN SYS001,X'192'
// DLBL SORTWK1,,1,SD
// EXTENT SYS005,,,,50,100
// DLBL SORTWK2,,1,DA
// EXTENT SYS001,,,,100,100
// EXTENT SYS001,,,,300,100
// DLBL SORTWK3,'DOS.WORKFILE.SYS007',0,VSAM,
          RECORDS=(1000,500),RECSIZE=80,DISP=(,DELETE)
```

This example assumes WORK=3 in the SORT program control statement.

Notes:

1. See Note 4 under Figure 9 for more information on assigning SYS numbers to SORTWKn files, if your installation uses both directly and program invoked sort applications.

2. The SAM ESDS file is defined implicitly. No EXTENT job control statement is needed if the default model for the volume is used.

3. DISP=(,DELETE) does not have to be specified if the work file is defined with a DTF containing TYPEFLE=WORK and DELETFL is not specified.

3.0 Chapter 3. Using DFSORT/VSE Program Control Statements

Subtopics:

3.1 Introduction

Program control statements direct DFSORT/VSE 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 application is to be performed.
- Describe the control fields to be used.
- Indicate program user exits for transferring control to your own routines.
- Describe DFSORT/VSE functions you want to have invoked.
- Describe input, output, and work files.
- Indicate various options you want to use during processing.

You can supply program control statements to DFSORT/VSE from one of the following:

- A SYSIPT file
- A parameter list

Each control statement is checked for validity by DFSORT/VSE. If DFSORT/VSE finds an error in a statement, it issues an appropriate message and will usually skip the rest of the statement (including any continuation lines) and continue checking the next statement. If an error has been found, DFSORT/VSE usually terminates after it finishes checking all of the statements.

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

3.2 Control Statement Summary

Subtopics:

3.2.1 Describing the Primary Task

The only required program control statement in a DFSORT/VSE application is a SORT or MERGE control statement that specifies whether you want to sort, merge, or copy records (copying can be specified using either of these statements), and a RECORD control statement that specifies record length and type information.

SORT
Describes control fields and the number of input and work files 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 and the number of input files if you are coding a merge application, or specifies a copy application. Indicates whether you want ascending or descending order for the merge.

RECORD
Describes record length and type information.

3.2.2 Including or Omitting Records

You can specify whether certain records are included in the output file or omitted from it.

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

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

3.2.3 Reformatting and Editing Records

You can modify individual records by deleting and reordering fields and inserting blanks, or binary zeros.

INREC
Specifies how records are reformatted before they are processed (sorted or merged).

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

3.2.4 Describing Input and Output Files

You can define the characteristics of your input and output files.

INPFIL
Describes the input files and specifies the procedures to be followed when tape input files are opened and closed. An INPFIL control statement is required if any default value for input processing is not applicable (for example, if SAM input files are blocked).

OUTFIL
Describes the output file and specifies the procedures to be followed when a tape output file is opened and closed. An OUTFIL control statement is required if any default value for output processing is not applicable (for example, if SAM output file is blocked).

3.2.5 Invoking Additional Functions and Options

You can use the remaining control statements to perform a variety of tasks.

ALTSEQ
Modifies the default collating sequence that was set at installation time. The modified sequence is used for any control field whose format is specified as AQ.

ANALYZE
Specifies that DFSORT/VSE is to make optimization calculations and determine how many records can be sorted with the specified configuration, if more virtual storage should be allocated, and so on. No actual sorting, merging, or copying is done.

END
Causes DFSORT/VSE to discontinue accepting control statements.

MODS
Specifies use of one or more user exit routines in a DFSORT/VSE application. See Chapter 4, "Using Your Own User Exit Routines" in topic 4.0 for information about user exit routines.

OPTION
Overrides installation defaults (such as CHALT and DIAG) and supplies optional information (such as ADDROUT and LABEL).

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

3.3 General Coding Rules

All DFSORT/VSE control statements have the same general format, shown in Figure 10. The illustrated format does not apply to control statements you supply in a parameter list. See Chapter 5, "Invoking DFSORT/VSE from a Program" in topic 5.0 for information on the special rules that apply.

Figure 10. Control Statement Format

The control statements are free-form; that is, the operation definer, operand(s), and remark field can appear anywhere in a control 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 control statement type to DFSORT/VSE. In the example below, the operation definer, SORT, is in the operation field of the sample control statement.

Operand Field:
The operand field is composed of one or more operands separated by commas. This field must follow the operation field, and be separated from it by at least one blank. No blanks are allowed within the operands, but a blank is required at the end of all operands. 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/VSE). A value or values can be associated with a parameter. The three possible operand formats are:

- parameter
- parameter=value
- parameter=(value*1*,value*2*,...,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 control statement is continued on the next line. However, as long as the last character of the operand field on a line is a comma followed by a blank, DFSORT/VSE 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.

Subtopics:

3.3.1 Continuation Lines

The format of the DFSORT/VSE continuation line is shown in Figure 11.

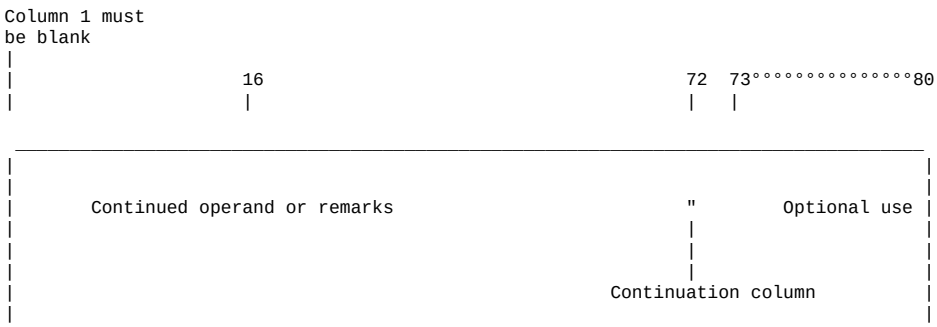


Figure 11. Continuation Line Format

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

A continuation line is treated as a logical extension of the preceding line. Either an operand or a 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 16.
- If an operand field is broken after a comma, the continuation column (72) can be left blank, and the continuation can begin in any column from 2 through 16. If the comma is in column 71 and column 72 contains a nonblank character, the continuation must begin in column 16.
- If an operand field is not broken after a comma, the operand field must be broken at column 71. Column 72 must contain a nonblank character. The continuation must begin in column 16.

Examples of Valid Continuation Lines

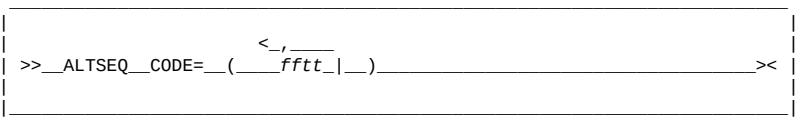
```
 _____________________________________________________________________
|                                                                     |
|                                                                     |
|                                                                     |
|   1             16                                              72| |
|   |             |                                                | | |
|   |             |                                                | | |
|                                                                   |
|    SORT FIELDS=(5,8,A,20,2,D),                                      |
|     FORMAT=CH                                                       |
|    OPTION PRINT=ALL,DIAG,      PRINT ALL MSGS   PRODUCE DIAGNOSTICS MSGS   |
|           ERASE                ERASE WORK FILES                     |
|    INCLUDE COND=(1,10,CH,EQ,C'STOCKHOLM',AND,21,8,ZD,GT,+50000,OR,31,4,CH* |
|                 ,NE,C'HERR')                                        |
|                                                                     |
|                                                                     |
|_____________________________________________________________________|
```

3.3.2 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.
- 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 and STOPAFT option) or storage requirements (such as those for STORAGE, GVSIZE, GVSRLOW, and GVSRANY options) can contain up to 10 digits (the maximum value is 2147483647). Values specified for LOCALE can contain up to 32 alphanumeric characters.
- Commas, equal signs, parentheses, and blanks can be used only as delimiters. They can be used in values only if the values are constants.
- Each type of control statement can appear only once for each execution of DFSORT/VSE.

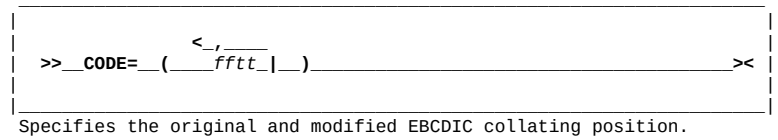
3.4 ALTSEQ Control Statement



The ALTSEQ control statement changes the default collating sequence of EBCDIC character data (ALTSEQ translate table); 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 installation, the ALTSEQ control statement overrides it.

When you supply an ALTSEQ control statement, DFSORT/VSE applies the modified collating sequence to any field whose format you specify in the SORT, MERGE, INCLUDE, or OMIT control statement as AQ. If you specify AQ without supplying an ALTSEQ control statement, DFSORT/VSE uses the default available at your installation, if there is one. Otherwise, DFSORT/VSE uses the standard EBCDIC collating sequence.

CODE



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.

Default: DFSORT/VSE installation default.

Subtopics:

- 3.4.1 ALTSEQ Control Statement Notes
- 3.4.2 Altering EBCDIC Collating Sequence--Examples

3.4.1 ALTSEQ Control Statement Notes

1. If CHALT option is in effect, control fields with format CH are collated using the ALTSEQ translate table, in addition to those with format AQ.

2. The moved character (ff) is considered equal to any character already occupying the position (tt).

3. A character can only be moved once.

4. The field to which the alternative sequence is to apply must be described as format AQ (or CH if CHALT option is in effect) in the SORT, MERGE, INCLUDE, or OMIT control statement that references it.

5. 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.

6. Each group of hexadecimal digits must contain exactly four digits.

7. Using the ALTSEQ control statement can degrade performance.

3.4.2 Altering EBCDIC Collating Sequence--Examples

Subtopics:

3.4.2.1 Example 1

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

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

3.4.2.2 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).

3.4.2.3 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 control statement does **not** cause collation of A before or after 1, or of B before or after 2.

3.4.2.4 Example 4

```
ALTSEQ   CODE=(C181,C282,8283,C384,8385,C486,8487,C588,8589,
          C68A,868B,C78C,878D,C88E,888F,C990,8991,D192,9193,
          D294,9295,D396,9397,D498,9499,D59A,959B,D69C,969D,
          D79E,979F,D8A0,98A1,D9A2,99A3,E2A4,A2A5,E3A6,A3A7,
          E4A8,A4A9,E5AA,A5AB,E6AC,A6AD,E7AE,A7AF,E8B0,A8B1,
          E9B2,A9B3)
```

Uppercase A is to collate before lowercase a, B before b, and so on through to Z and z.

3.4.2.5 Example 5

```
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 collating that is not case-sensitive.

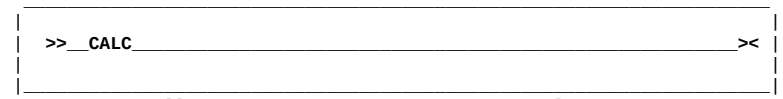
3.5 ANALYZE Control Statement

```
>>--ANALYZE--CALC-------------------------------------------------><
```

The ANALYZE control statement enables you to test your DFSORT/VSE control statements before running your sort, merge, or copy application. It causes

DFSORT/VSE to cancel the run, without actually sorting, merging, or copying, after analyzing the control statements and making its optimization calculations based on the information in the control statements.

CALC



Causes the following options to be forced, regardless of what has been specified in the OPTION control statement:

```
DIAG, NODUMP, ROUTE=LST (or ROUTE=xxx), PRINT=ALL
```

These options cause all diagnostic messages to be produced (for example, those relating to DFSORT/VSE storage requirements), after which message ILU029A ANALYZE END is issued and DFSORT/VSE terminates with a return code of 16.

If DFSORT/VSE is program invoked and a value for ROUTE=xxx has been supplied (explicitly or by default), that value will be used instead of ROUTE=LST.

Default: None; optional.

Subtopics:

- 3.5.1 Testing DFSORT/VSE Control Statements--Example

3.5.1 Testing DFSORT/VSE Control Statements--Example

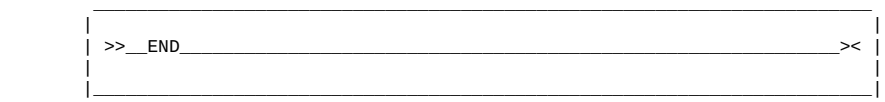
Subtopics:

- 3.5.1.1 Example

3.5.1.1 Example

Produces all diagnostic messages, issues message ILU029A ANALYZE END, and terminates DFSORT/VSE with a return code of 16.

3.6 END Control Statement



The END control statement allows DFSORT/VSE to discontinue accepting control statements.

Any statements (for example, control statements, JCL statements except /*, and so on) between the END and /* are read by DFSORT/VSE but not processed.

Notes:

1. DFSORT/VSE control statements in the SYSIPT file must be followed with /* JCL statement.

2. Starting with DFSORT/VSE Version 3 Release 2, if you do not specify /* after END, DFSORT/VSE ignores statements that were processed by the system with DOS/VS-VM/System Product Sort/Merge Version 2 Release 5 and DFSORT/VSE Version 3 Release 1.

Subtopics:

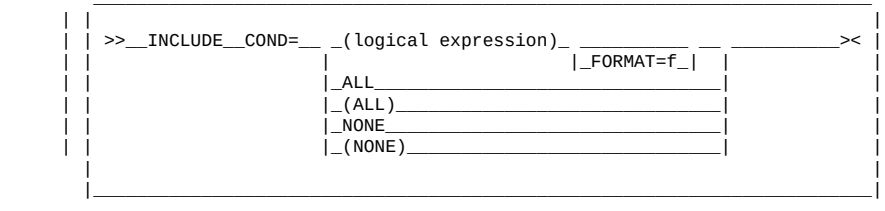
- 3.6.1 Discontinue Processing Control Statement--Example

3.6.1 Discontinue Processing Control Statement--Example

```
SORT FIELDS=(1,6,A,28,5,D),FORMAT=CH
RECORD TYPE=V,LENGTH=(200,,,,80)
END
OPTION GVSIZE=40M
```

Because the OPTION control statement appears after the END control statement, the OPTION control statement is read but not processed.

3.7 INCLUDE Control Statement

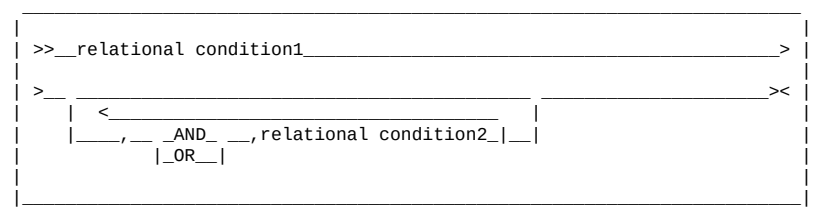


Use an INCLUDE statement if you want only certain records to appear in the output file. The INCLUDE statement selects the records you want to include.

You can specify either an INCLUDE statement or an OMIT statement in the same DFSORT/VSE run, but not both.

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:



If the logical expression is true for a given record, the record is included in the output file.

Three 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" in topic 3.7.2 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'AB,LM,DE' for the value in a 2-byte field, and include only those records for which C'AB', C'LM', or C'DE' appears in the field.

   See "Substring Comparison Tests" in topic 3.7.4 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" in topic 3.7.6 for information about bit logic tests.

By nesting relational conditions within parentheses, you can create logical expressions of higher complexity.

Although comparisons, substring comparison tests, and bit logic tests are explained separately below for clarity, they can be combined to form logical expressions.

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 file.

| **ALL or (ALL)**
specifies that all of the input records are to
| be included in the output file.

| **NONE or (NONE)**
specifies that none of the input records are to
| be included in the output file.

Default: None; must be specified.

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 Figure 12 in topic 3.7.2.1.  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.  If you have specified DATA=A in the
 INPFIL control statement, you may only use AC, AST and ASL.
```

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

Subtopics:

3.7.1 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:

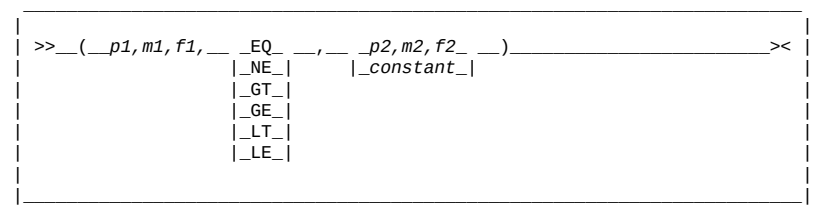
- AND statements are evaluated before OR statements unless parentheses are used to change the order of evaluation; expressions inside parentheses are always evaluated first. (Nesting of parentheses is limited only by the amount of storage available.)
- The symbols & (AND) and | (OR) can be used instead of the words.

3.7.2 Comparisons

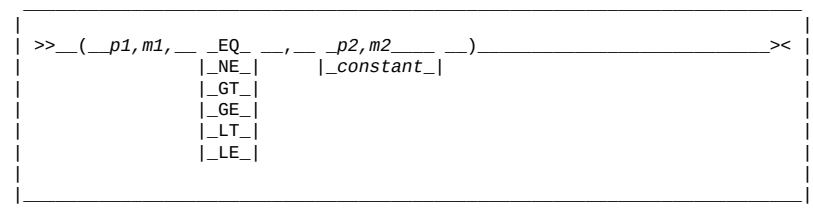
Subtopics:

3.7.2.1 Relational Condition Format

Two formats for the relational condition can be used:



Or, if FORMAT operand is used,



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. 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 record descriptor word). All compare fields must start on a byte boundary, and no compare field can extend beyond byte 4092.

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

- m1 specifies the length of the compare field. The acceptable lengths for different formats are given in Figure 12.

- f1 specifies the format of the data in the compare field. The permissible formats for comparisons are shown in Figure 13. If you have specified DATA=A in the INPFIL control statement, you may only use AC, AST, and ASL.

  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.

Figure 12. Compare Field Formats and Lengths

Format	Length	Description
CH	1 to 256 bytes	Character
AQ	1 to 256 bytes	Character with alternate collating sequence
ZD	1 to 18 bytes	Signed zoned decimal
PD	1 to 16 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
CSL	2 to 256 bytes	Signed numeric with leading separate sign
CST	2 to 256 bytes	Signed numeric with trailing separate sign
CLO	1 to 256 bytes	Signed numeric with leading overpunch sign
CTO	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

Notes:

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

2. See Appendix B, "Data Format Examples" in topic B.0 for detailed format descriptions.

p2,m2,f2: These parameters specify another field in the input record with which the p1, m1, and f1 field will be compared. Permissible comparisons between compare fields with different formats are shown in Figure 13.

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

Figure 13. Permissible Field-to-Field Comparisons for INCLUDE and OMIT Control Statements

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

Note: If LOCALE=NONE is not specified or defaulted, CH fields can only be compared with CH fields.

Constant: 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 Figure 14.

Figure 14. 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		
CSL	X		
CST	X		
CLO	X		
CTO	X		
AQ		X	X

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

```
 _____________________________________________________________
|                                                             |
|   [°][nn...n][.nn...n]                                      |
|                                                             |
|_____________________________________________________________|
```

Where *n* represents any decimal digit.

- Any number of digits can be specified except when the constant is to be compared with a field of FI format, when the constant may not be larger than 2147483647 nor smaller then -2147483648.
- The decimal point is not allowed in comparisons with fields of format FI, ZD, or PD.
- The decimal point, if specified, is counted in the length of the constant and included in it as a character.

Examples of valid and invalid decimal constants are:

Figure 15. Valid and Invalid Decimal Constants

Valid	Invalid	Explanation
15	++15	Too many sign characters
+15	15+	Sign in wrong place
-15	-15.	Invalid decimal point
14.09	1.4.09	Too many decimal points
18000000	1,800,000	Contains invalid characters

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

```
 _____________________________________________________________
|                                                             |
| C'cc...c'                                                   |
|                                                             |
|_____________________________________________________________|
```

The value c 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:

Figure 16. Valid and Invalid Character Constants

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

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 17. Valid and Invalid Strings with Double-Byte Data

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

```
X'xx...xx'
```

The value xx represents any pair of hexadecimal digits. You can specify up to 256 pairs of hexadecimal digits.

Examples of valid and invalid hexadecimal constants are:

Figure 18. Valid and Invalid Hexadecimal Constants

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

3.7.2.2 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 EBCDIC character string; blank character
- X'20' For ISCII/ASCII character string; blank character
- X'00' For a hexadecimal string; binary zeros.

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

3.7.2.3 Cultural Environment Considerations

DFSORT/VSE'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. DFSORT/VSE 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" in topic 1.3.3 or LOCALE in "OPTION Control Statement" in topic 3.13.

3.7.3 Including Records in the Output File--Comparison Examples

Subtopics:

- 3.7.3.1 Example 1
- 3.7.3.2 Example 2
- 3.7.3.3 Example 3

3.7.3.1 Example 1

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

This example illustrates how to only include records in which:

- The fixed-integer number in bytes 5 through 12 is greater than the fixed-integer number in bytes 13 through 20

  OR

- The fixed-integer number in bytes 105 through 108 is less than or equal to 1000.

Note that all three compare fields have the same format.

3.7.3.2 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 50000

  OR

- Bytes 31 through 34 do not contain HERR.

Note that the AND is evaluated before the OR. ("Example" in topic 3.12.1.1 illustrates how parentheses can be used to change the order of evaluation.) Also note that ending a line with a comma followed by a blank indicates that the parameters continue on the next line, starting in any position from columns 2 through 16.

3.7.3.3 Example 3

```
   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 to 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.

3.7.4 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.

Subtopics:

- 3.7.4.1 Relational Condition Format

3.7.4.1 Relational Condition Format

Two formats for the substring comparison relational condition can be used:

```
>>--(p1,m1,SS,--_EQ_--,--constant--)-------------------------------><
                |_NE_|
```

Or, if the FORMAT=SS operand is used:

```
>>--(p1,m1,--_EQ_--,--constant--)---------------------------------><
              |_NE_|
```

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. 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 record descriptor word). All fields to be tested must start on a byte boundary and must not extend beyond byte 4092.

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

- 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" in topic 3.7.2.1 and "Hexadecimal String Format" in topic 3.7.2.1 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 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.

3.7.5 Including Records in the Output File--Substring Comparison Example

Subtopics:

3.7.5.1 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.

3.7.6 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.

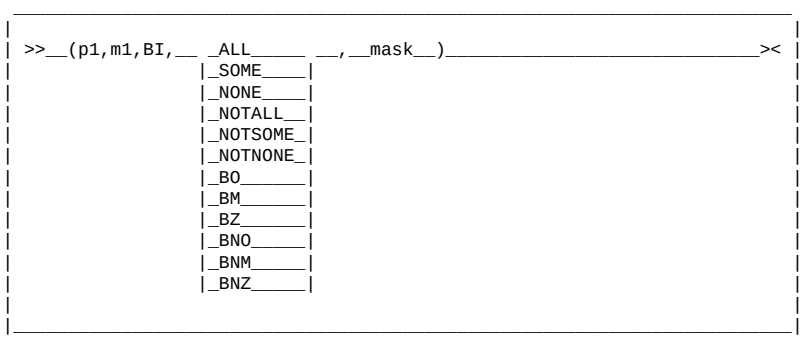
3.7.7 Method 1: Bit Operator Tests

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.

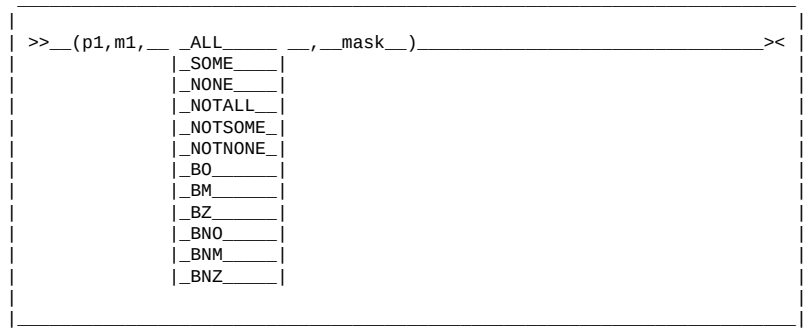
Subtopics:

3.7.7.1 Relational Condition Format

Two formats for the relational condition can be used:



Or, if the FORMAT=BI operand is used:



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) is intended for those who like meaningful mnemonics. The second set of operators (BO, BM, and so on) is intended for those familiar with the conditions associated with the TM instruction.

3.7.7.2 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. 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 record descriptor word). All fields to be tested must start on a byte boundary and must not extend beyond byte 4092.

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

- m1 specifies the length of the field to be tested. The length can be 1 to 256 bytes.

3.7.7.3 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.

3.7.8 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).

3.7.9 Including Records in the Output File--Bit Operator Test Examples

Subtopics:

3.7.9.1 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.

3.7.9.2 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:

Figure 19. Bit Operator 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

3.7.9.3 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:

Figure 20. Bit Operator 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

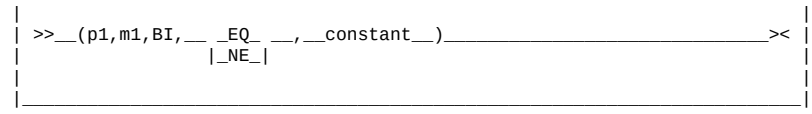
3.7.10 Method 2: Bit Comparison Tests

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.

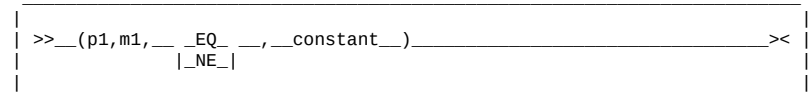
Subtopics:

3.7.10.1 Relational Condition Format

Two formats for the relational condition can be used:



Or, if the FORMAT=BI operand is used:



```
|_____________________________________________________________________|
```

Bit comparison operators are as follows:

EQ
Equal to
NE
Not equal to

3.7.10.2 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. 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 record descriptor word). All fields to be tested must start on a byte boundary and must not extend beyond byte 4092.

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

- m1 specifies the length of the field to be tested. The length can be 1 to 256 bytes.

3.7.10.3 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 a 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 period (.) for 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).

3.7.10.4 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.

3.7.11 Including Records in the Output File--Bit Comparison Test Examples

Subtopics:

3.7.11.1 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.

3.7.11.2 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:

Figure 21. 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

3.7.11.3 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:

Figure 22. 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

3.7.12 INCLUDE and OMIT Control Statements Notes

1. Floating point compare fields cannot be referenced in INCLUDE or OMIT control statements.

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

3. In the compare fields and decimal self-defining term, +0, 0, and -0 are treated as the same number and compare equal.

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

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

Figure 23 in topic 3.7.12 shows how DFSORT/VSE reacts to the result of a relational condition comparison, depending on whether the control statement is INCLUDE or OMIT and whether the relational condition is followed by an AND or an OR logical operator.

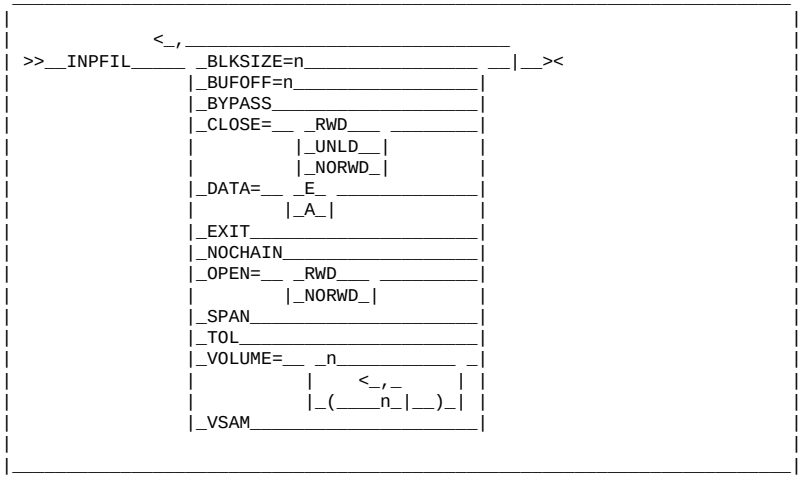
When writing complex control statements, the table in Figure 23 helps you get the result that you want.

Figure 23. Logic Table for INCLUDE and OMIT Control Statements

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

---

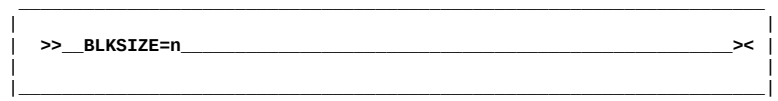
3.8 INPFIL Control Statement



The INPFIL control statement defines the input files to DFSORT/VSE and specifies the procedure to be followed when tape input files are opened or closed.

The INPFIL control statement is only required if a default value for input processing is not applicable.

BLKSIZE



- Input files are VSAM files, accessed as VSAM, or
- All input files are on FBA devices, or
- All input files are unblocked.

Otherwise, it must be supplied because DFSORT/VSE does not support the BLKSIZE parameter of the DLBL job control statement.

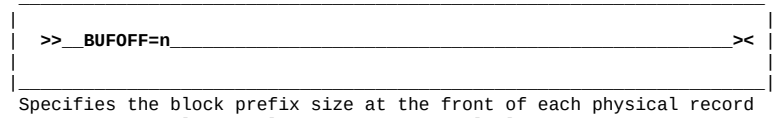
When specified for variable-length records, the BLKSIZE value must include the block descriptor word (4 bytes). For ISCII/ASCII data, it must include the BUFOFF value.

If input files have different block sizes, n must be equal to the largest block size. The maximum permissible block size for input files is shown in Figure 5 in topic 1.4.4.

Default:

- Fixed-length records: L1 value (specified from the RECORD control statement)
- Variable-length records: L1 value (specified from the RECORD control statement) plus 4 bytes
- ISCII/ASCII variable-length records: L1 value (specified from the RECORD control statement) plus BUFOFF value.

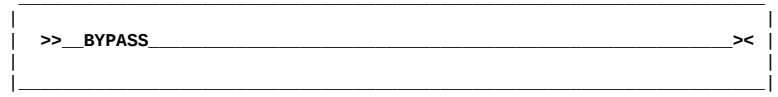
BUFOFF



n can be any value from 0 through 99.

Default: 0.

BYPASS



```
Specifies that DFSORT/VSE is to skip incorrectly read input data
blocks and wrong-length physical records (DFSORT/VSE prints an
informational message and continues).  If BYPASS is not specified,
DFSORT/VSE will issue an error message and terminate if it encounters
incorrectly read input data blocks or wrong-length physical records.
```

If BYPASS is specified for SAM ESDS files, the rest of the control interval with an error is bypassed.

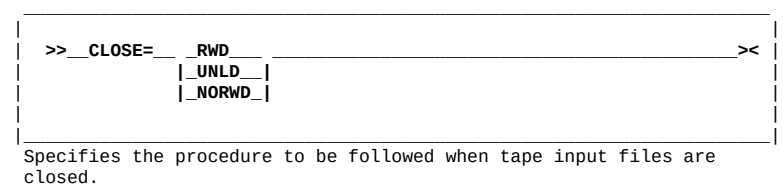
If BYPASS is specified for FBA input files, it is ignored.

If BYPASS is specified in conjunction with the SPAN operand, all complete and partial records in a block with an error will be skipped.

BYPASS prevents command chaining for the input files.

Default: None; optional.

CLOSE

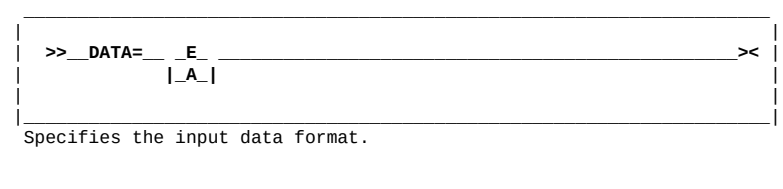


Specifies the procedure to be followed when tape input files are closed.

RWD specifies that tapes are to be rewound.
UNLD specifies that tapes are to be rewound and unloaded.
NORWD specifies that tapes are not to be rewound.

Default: RWD.

DATA



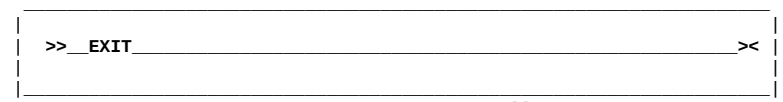
Specifies the input data format.

E
specifies that the input data format is EBCDIC.

A
specifies that the input data format is ISCII/ASCII. This value can only be specified for tape input files.

Default: E.

EXIT



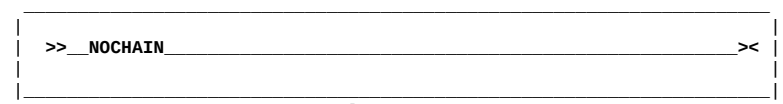
Specifies that your user exit routine reads all input data and passes each record to DFSORT/VSE. You must:

- Activate E15 (sort application) or E32 (merge or copy application) user exit by specifying its name in the MODS control statement.
- Provide a routine at E15 (sort application) or E32 (merge or copy application) user exit which will open the file(s), read them, and pass the records, one at a time, to DFSORT/VSE.

When EXIT is specified, all other INPFIL control statement operands except DATA are ignored. See "INPFIL Control Statement Notes" in topic 3.8.1 for further details.

Default: None; optional.

NOCHAIN



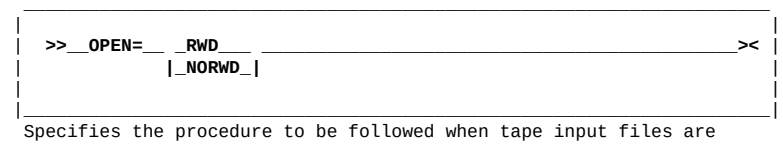
Specifies that DFSORT/VSE should not use command chaining when reading input files. NOCHAIN should *only* be used (for performance reasons) when input consists of blocked, fixed-length records on *tape files*, and you know that a large number of blocks are shorter than the specified block size, such as when:

- Input files have different block sizes, or
- One or more files contain a large number of short blocks.

See Chapter 7, "Improving Efficiency" in topic 7.0 for a discussion of performance.

Default: None; optional.

OPEN



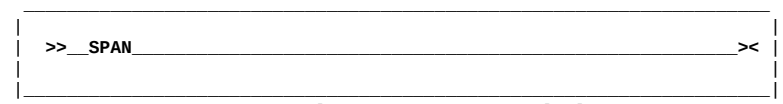
Specifies the procedure to be followed when tape input files are opened.

RWD specifies that the first volume of each input file is to be rewound before being read.

NORWD specifies that the first volume of each input file is not to be rewound before being read.

Default: RWD.

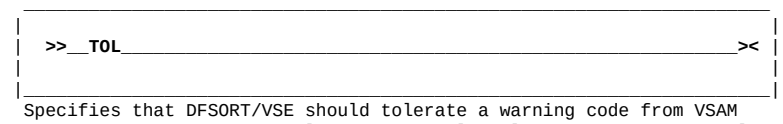
SPAN



Specifies that SAM input files consist of variable-length EBCDIC records and that the records may be spanned. The RECORD control statement must specify TYPE=V. This operand is not required if VSAM input files consist of spanned records.

Default: None; optional.

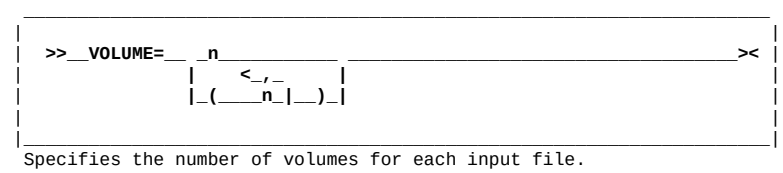
TOL



Specifies that DFSORT/VSE should tolerate a warning code from VSAM when opening VSAM input files. It is only valid for VSAM input files. See "INPFIL Control Statement Notes" in topic 3.8.1 Note 5.

Default: None; optional.

VOLUME

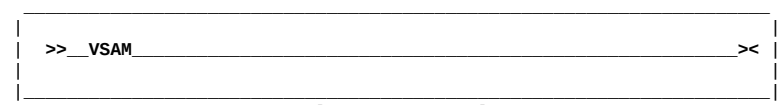


Specifies the number of volumes for each input file.

n can be a value between 1 and 255. The values must be specified in the order SORTIN1,SORTIN2,...,SORTIN9. VOLUME should only be specified for unlabeled or nonstandard labeled files; if specified for a labeled file, it is ignored.

Default: 1.

VSAM



Specifies that the input files are VSAM files. If the VSAM operand is not specified, SAM input files are assumed. The VSAM operand may also be used to access SAM ESDS files with VSAM.

A mixture of SAM and VSAM input files is not allowed, but SAM ESDS files can be mixed with either, provided all input files are to be accessed in the same way, that is, either all SAM or all VSAM.

The VSAM operand overrides all other INPFIL control statement operands except TOL and EXIT. EXIT overrides VSAM.

Default: None; optional.

Subtopics:

3.8.1 INPFIL Control Statement Notes

1. Any operands which precede EXIT in the INPFIL control statement are checked for syntax errors and flagged. Operands which follow EXIT are also checked for syntax errors and flagged, but all values are ignored.

2. The presence or absence of the EXIT operand affects the default symbolic unit names for the other files used by DFSORT/VSE. See Figure 9 in topic 2.2 for details.

3. If the EXIT operand and the ADDROUT or ADDROUT=D option are specified, DFSORT/VSE will terminate and issue an appropriate message.

4. DFSORT/VSE adds a record descriptor word (RDW) of 4 bytes to every variable-length record received from VSAM files. The user-written exit routines which handle variable-length records received from DFSORT/VSE need not, therefore, be designed separately for handling VSAM and SAM files.

5. If TOL is not specified for VSAM input files, a warning return code from VSAM will normally be recognized as an error by DFSORT/VSE; an error message will be issued, and DFSORT/VSE will terminate.

   If TOL is specified, DFSORT/VSE reads the input files without repairing it, and no error message is issued. Examples of conditions that produce warning messages are 'NOT PROPERLY CLOSED' or 'SYSTEM TIME STAMPS OF DATA AND INDEX DO NOT MATCH'.

   A *critical* return code from VSAM will always cause DFSORT/VSE to terminate, regardless of whether TOL has been specified, except for a null VSAM file (return code X'6E').

6. Since it can affect performance, BYPASS should not be used to omit certain extra short or long input records. The OMIT function should be used for this.

7. For SAM ESDS files, the BLKSIZE operand refers to the maximum logical block size. That is, it should match the RECORDSIZE (maximum) operand specified in the DEFINE command for the file when it was created with VSAM Access Method Services (AMS).

   Note also that fixed-length input blocks that are too long are truncated by data management and cannot be checked by DFSORT/VSE.

3.8.2 Defining Input Files--Examples

Subtopics:

3.8.2.1 Example 1

```
INPFIL VOLUME=(3,5,2,1),BLKSIZE=400,OPEN=NORWD,
       BYPASS,DATA=A,BUFOFF=99
```

VOLUME The four input files consist of three, five, two, and one unlabeled tape volumes respectively.

BLKSIZE The block size is 400 bytes.

OPEN The first volume of each input file is not to be rewound before being read.

BYPASS Incorrectly read input data blocks are to be ignored.

DATA Input data is ISCII/ASCII.

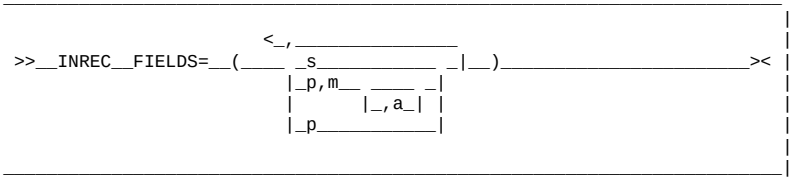
BUFOFF Each block has a buffer offset of 99 bytes.

3.8.2.2 Example 2

```
INPFIL  EXIT
```

EXIT All input data is received by DFSORT/VSE from either an E15 (sort application) or E32 (merge or copy application) user exit routine.

3.9 INREC Control Statement



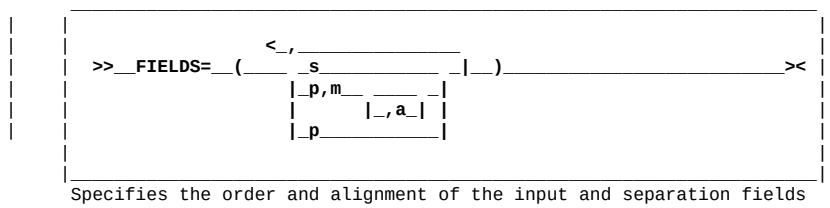
The INREC control statement allows you to reformat the input records before they are processed. That is, to define which parts of the input record are to be included in the reformatted input record, in what order they are to appear, and how they are to be aligned.

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

You can also insert blanks and binary zeros as separators (separation fields) before, between, and after the fields in the reformatted input records.

For information concerning the interaction of INREC and OUTREC control statements, see "INREC Control Statement Notes" in topic 3.9.1 and Chapter 7, "Improving Efficiency" in topic 7.0.

FIELDS



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

s

Specifies a separation field to be inserted into the reformatted input record in the position you code it relative to the other fields. It can be specified before or after any input field. Consecutive separation fields 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, and must meet the requirement to include at least one byte of fixed data in the reformatted input record. Permissible values are nX and nZ.

nX

Blank separation. n bytes of EBCDIC blanks (X'40') are inserted in the reformatted input records. n can range from 1 to 256. If n is omitted, 1 is used.

If you specify DATA=A in the INPFIL control statement, n bytes of ISCII/ASCII blanks (X'20') are inserted in the reformatted input record.

Examples of valid and invalid blank separation are shown in Figure 24.

Figure 24. Examples of Valid and Invalid Blank Separation

Validity	Specified	Result
Valid	1X or X	1 blank
Valid	256X	256 blanks
Invalid	500X	Too many repetitions. Use two adjacent separation fields instead (256X,244X, for example)
Invalid	0X	0 is not allowed

nZ

Binary zero separation. n bytes of binary zeros (X'00') are inserted in the reformatted input records. n can range from 1 to 256. If n is omitted, 1 is used.

Examples of valid and invalid binary zero separation are shown in Figure 25.

Figure 25. Examples of Valid and Invalid Binary Zero Separation

Validity	Specified	Result
Valid	1Z or Z	1 binary zero
Valid	256Z	256 binary zeros
Invalid	500Z	Too many repetitions. Use two adjacent separation fields instead (256Z,244Z, for example)
Invalid	0Z	0 is not allowed

p,m,a

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

p

specifies the first byte of the input field relative to the beginning of the input record. 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 record descriptor word). Fields can start anywhere within the record on a byte boundary. For special rules concerning variable-length records, see "INREC Control Statement Notes" in topic 3.9.1.

Note: If your E15 user exit routine reformats the record, p must refer to the record as reformatted by the routine.

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 Control Statement Notes" in topic 3.9.1 for more information.

a

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

The 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 is always padded with binary zeros.

| p

| specifies 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.

| A value must be specified for p that is less than or equal to the
| minimum input record length (see L4 value in "RECORD Control
| Statement" in topic 3.16) plus 1 byte.

Default: None; must be specified.

Subtopics:

3.9.1 INREC Control Statement Notes

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

2. In general, the OUTREC control statement should be used rather than the INREC control statement so your SORT and SUM control statements can refer to fields in the original input records.

3. If you use locale processing (LOCALE=NONE is not specified or defaulted), you must not use INREC. Use the OUTREC control statement instead of INREC.

4. When the INREC control statement is specified, you must be aware of the change in record size and layout of the resulting reformatted input records.

5. Performance can be improved if you can significantly reduce the length of your records with the INREC control statement. The INREC and OUTREC control statements should not be used unless they are actually needed to reformat your records.

6. For variable-length records, the first input field in the FIELDS operand must specify or include the 4-byte record descriptor word (RDW). DFSORT/VSE 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 first input field in the FIELDS operand can specify both RDW and data field in one. Otherwise, the RDW must be specifically included in the reformatted input record.

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

If both the INREC and OUTREC control statements 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 record includes only the RDW and the variable part of the input record, "null" records containing only an RDW could result.

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

9. Fields referenced in the INREC control statement can overlap each other and control fields or both.

10. If input is variable-length records, the output is also variable. This means that each record is given the correct RDW by DFSORT/VSE before output.

11. If overflow might occur during summation, the INREC control statement can be used to create a larger summary field in the reformatted input record (perhaps resulting in a larger record for sorting or merging) so that overflow does not occur.

12. DFSORT/VSE issues a message and terminates if an INREC control statement is specified with a MERGE control statement or a SORT FIELDS=COPY control statement.

13. DFSORT/VSE will calculate the L3 value (see L3 of the RECORD control statement).

14. You must consider the calculated value of L3 when specifying the BLKSIZE operand in the OUTFIL control statement.

15. The ADDROUT or ADDROUT=D operand in the OPTION control statement is ignored when an INREC control statement has been specified.

3.9.2 Reformatting Records Before Processing--Examples

Subtopics:

3.9.2.1 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),WORK=1,FILES=3
 SUM FIELDS=(17,4,BI)
 RECORD TYPE=F,LENGTH=80
 INPFIL BLKSIZE=240
 OUTFIL BLKSIZE=230
```

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),WORK=1,FILES=3
   SUM FIELDS=(38,4,BI)
   RECORD TYPE=F,LENGTH=80
   INPFIL BLKSIZE=240
   OUTFIL BLKSIZE=230
```

The above examples illustrate how three fixed-length input files are sorted and reformatted for output. Unnecessary fields are eliminated from the output records using INREC or OUTREC control statement. The input record size is 80 bytes.

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

The reformatted input records are fixed length with a record size of 23 bytes. The reformatted records, after INREC or OUTREC processing, are shown below.

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 the INREC or OUTREC statement. However, use of the OUTREC statement makes it easier to code the SORT and SUM statements. In either case, the INCLUDE COND operands must refer to the fields of the original input records. However, with the INREC statement, the SUM and SORT FIELDS operands must refer to the fields of the *reformatted* input records, while with the OUTREC statement, the SUM and SORT FIELDS operands must refer to the fields of the *original* input records.

3.9.2.2 Example 2

```
   INREC FIELDS=(1,35,2Z,36,45)
   SORT FIELDS=(20,4,CH,D,10,3,CH,D),WORK=1,FILES=3
   SUM FIELDS=(36,4,BI,40,8,PD)
   RECORD TYPE=F,LENGTH=80
   INPFIL BLKSIZE=240
   OUTFIL BLKSIZE=246
```

This example illustrates how overflow of a summary field can be prevented when three fixed-length files are sorted and reformatted for output. The input record size is 80 bytes.

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 are shown below.

Position Contents

1-35
: Input positions 1 through 35

36-37
: Binary zeros (to prevent overflow)

38-82
: Input positions 36 through 80

The SORT 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* sorting. This prevents overflow of this summary field. Note that, if the OUTREC statement were used instead of the INREC statement, the records would be reformatted *after* sorting, 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.

3.9.2.3 Example 3

```
 _____________________________________________________________________
|                                                                     |
|   INREC FIELDS=(1,4,15,15,47,50,101)                                |
|   SORT  FIELDS=(32,4,CH,A),WORK=1,FILES=1                           |
|   RECORD TYPE=V,LENGTH=(200,,,100,130)                              |
|   OUTREC FIELDS=(1,4,10Z,5,15,17Z,20,50,4X,70)                      |
|   INPFIL BLKSIZE=2004                                               |
|   OUTFIL BLKSIZE=2004                                               |
|                                                                     |
|_____________________________________________________________________|
```

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

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

Position Contents

1-4
: RDW (input positions 1 through 4)

5-19
: Input positions 15 through 29

20-69
: Input positions 47 through 96

70-n
: Input positions 101 through n (variable part of input records)

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

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

Since padding fields will be removed by the INREC statement and reinserted by the OUTREC statement, the output records will be identical to the original input records; that is, variable length, with a minimum record size of 100 bytes and maximum record size of 200 bytes. They are shown below.

Position Contents

1-4
: RDW

5-14
: Binary zeros

15-29
: Input positions 15 through 29

30-46
: Binary zeros

47-96
: Input positions 47 through 96

97-100
: EBCDIC blanks

101-n
: Input positions 101 through n (variable part of input records)

Thus, the use of the INREC and OUTREC statements allowed sorting of smaller records, even though the output records were not actually reformatted.

3.9.2.4 Example 4

```
 _____________________________________________________________________
|                                                                     |
|   INREC FIELDS=(20,4,12,3)                                          |
|   SORT FIELDS=(1,4,D,5,3,D),FORMAT=CH,WORK=1,FILES=3                |
|   OUTREC FIELDS=(5X,1,4,H,8X,1,2,5,3,80Z)                           |
|   RECORD TYPE=F,LENGTH=80                                           |
|   INPFIL BLKSIZE=240                                                |
|   OUTFIL BLKSIZE=206                                                |
|                                                                     |
|_____________________________________________________________________|
```

This example illustrates how three fixed-length input files can be sorted and reformatted for output. A more efficient sort is achieved by using the INREC statement to reduce the input records as much as possible before sorting and using the OUTREC statement to repeat fields and insert padding after sorting. The input record size 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 are shown below.

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, and have a record size of 103 bytes. They are shown below.

Position Contents

1-5
: EBCDIC blanks

6
: Binary zero (for H alignment)

7-10
: Input positions 20 through 23

11-18
: EBCDIC blanks

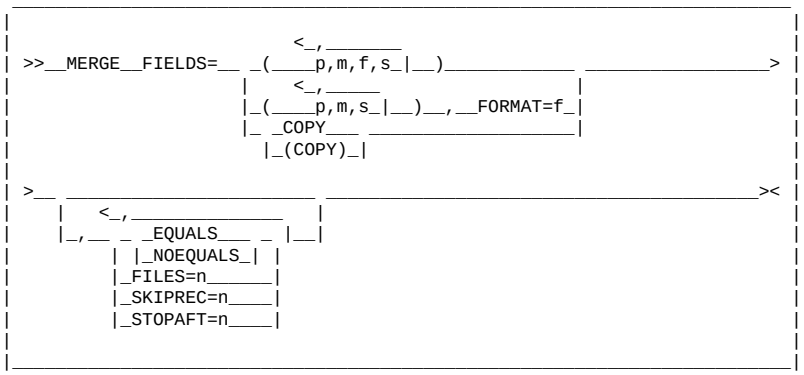
19-20
: Input positions 20 through 21

21-23
: Input positions 12 through 14

24-103
: Binary zeros.

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

3.10 MERGE Control Statement



The MERGE control statement must be used when a merge application is to be performed; this statement describes the control fields in the input records on which the files have previously been sorted.

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

DFSORT/VSE's collating and ordering 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. DFSORT/VSE will produce 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 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) control fields.

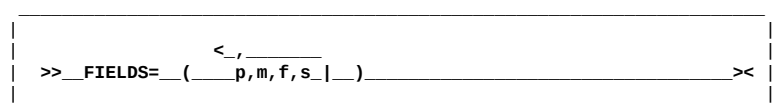
For more information on locale processing, see "Cultural Environment Considerations" in topic 1.3.3 or LOCALE in "OPTION Control Statement" in topic 3.13.

Notes:

1. Up to 9 files can be merged or copied.

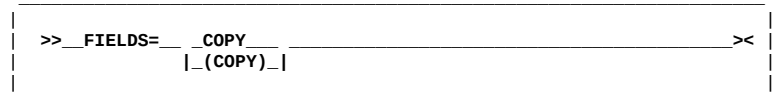
2. If you use an E35 user exit routine, sequence checking is not performed at the time the records are passed to the E35 user exit; therefore, you must ensure that input records are in correct sequence.

FIELDS



The FIELDS operand is written exactly the same way for a merge as it is for a sort. The meanings of p, m, f, and s are described in the discussion of the SORT statement. The defaults for this and the following parameters are also given there. See "SORT Control Statement" in topic 3.17.

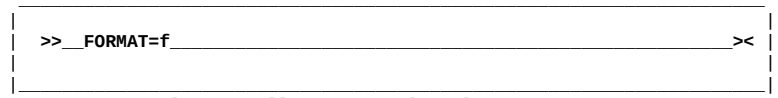
FIELDS=COPY or FIELDS=(COPY)



Can be used to copy one or more input files to an output file. All other MERGE operands (except SKIPREC and STOPAFT) are ignored when FIELDS=COPY is specified. If FIELDS=COPY is specified with a SUM or INREC control statement, DFSORT/VSE will terminate.

Default: None; optional.

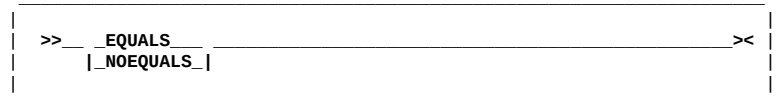
FORMAT



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 operand.

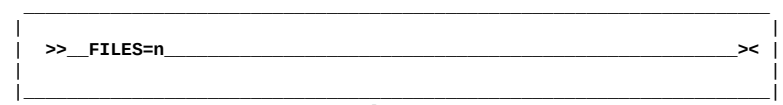
Default: None; must be specified if not included in FIELDS operand.

EQUALS or NOEQUALS



See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" in topic 3.13.

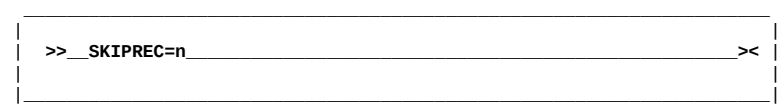
FILES



Specifies the number of input files to be merged or copied, where n can be any number from 1 through 9.

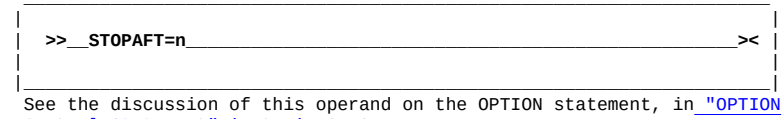
Default: None, must be specified for merge applications. 1 for copy applications.

SKIPREC



See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" in topic 3.13.

STOPAFT



See the discussion of this operand on the OPTION statement, in "OPTION Control Statement" in topic 3.13.

Subtopics:

3.10.1 MERGE Control Statement Notes

1. In control fields, +0, 0, and -0 are treated as the same number and compare equal.
2. Input files must be all SAM or all VSAM.
3. If any of the input files are multivolume and unlabeled or nonstandard labeled, you must specify the number of volumes using the VOLUME operand of the INPFIL control statement.
4. DFSORT/VSE issues a message and terminates if an INREC control statement is specified with a MERGE control statement.

3.10.2 Merging and Copying--Examples

Subtopics:

3.10.2.1 Example 1

```
MERGE   FIELDS=(2,5,CH,A),FILES=3
```

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

FILES Three input files are to be merged.

3.10.2.2 Example 2

```
MERGE   FIELDS=(25,4,A,48,8,A),FORMAT=ZD,FILES=9
```

FIELDS The major control field begins on byte 25 of each record, is 4 bytes long, and contains zoned decimal data that has also been presorted in ascending order.

The second control field begins on byte 48, is 8 bytes long, has the same data format as first field, and has also been presorted in ascending order.

FORMAT The FORMAT operand can be used because both control fields have the same data format (zoned decimal).

It would also be correct to write this MERGE control statement as follows:

```
MERGE  FIELDS=(25,4,ZD,A,48,8,ZD,A),FILES=9
```

FILES Nine input files are to be merged.

3.10.2.3 Example 3

```
 _____________________________________________________________________
|                                                                     |
|   MERGE  FIELDS=COPY                                                |
|                                                                     |
|_____________________________________________________________________|
```

FIELDS The input file is copied to the output file without merging.

3.11 MODS Control Statement

```
 _____________________________________________________________________
|                                                                     |
|           <_,_____________________________________                  |
|                                        <_,____     |                |
| >>__MODS____PHx=__(__ ___ __,__ ___ __,____exit_|__)_|_____________><|
|                      |_n_|     |_m_|                               |
|                                                                     |
|_____________________________________________________________________|
```

The MODS control statement is needed only when DFSORT/VSE passes control to your routines at user exits. The MODS control statement associates your routines with specific DFSORT/VSE user exits and provides DFSORT/VSE with descriptions of these routines. For details about DFSORT/VSE user exits and how your routines can be used, see Chapter 4, "Using Your Own User Exit Routines" in topic 4.0.

To use one of the user exits, you associate its three-character name (for example, E31) with the phase in which it will be used.

PHx

```
 _____________________________________________________________________
|                                                                     |
|      <_,_____________________________________                       |
|                                        <_,____     |                |
|  >>____PHx=__(__ ___ __,__ ___ __,____exit_|__)_|_____________><|
|                  |_n_|     |_m_|                                   |
|                                                                     |
|_____________________________________________________________________|
Specifies the DFSORT/VSE phase; PH1 specifies the initial sorting
phase (phase 1), and PH3 specifies the final merging phase (phase 3).
The values that follow PHx describe your routines, which can be used
in DFSORT/VSE phase x.  These values are:
```

n specifies the cataloged name of your routines to be executed at
| the user exits specified. n must be omitted if the routines
| are already in virtual storage. You can use any valid operating system name for your routines. This allows you to keep several alternative routines with different names in the same library.

m specifies loading information, and describes one of two ways in

| which the user routines may be loaded into virtual storage by
| DFSORT/VSE.

The value can be the absolute loading address (expressed in decimal), or the length of the user routines to be executed (expressed in bytes (decimal) and prefixed with L).

| m must be omitted if the routines are already in virtual
| storage.

exit specifies the user exits that are to be used in the specified DFSORT/VSE phase. The values are expressed as E31, E38, and so on. They can be in any order.

Default: None; must be specified if you use user exit routines.

Subtopics:

3.11.1 MODS Control Statement Notes

1. The absolute loading address must be a virtual address if DFSORT/VSE is to run in virtual mode, or a real address if DFSORT/VSE is to run in real mode.

2. If the absolute loading address is used, the length of the module
   | containing your routines is unknown to DFSORT/VSE. This means that
   | all virtual storage in the partition program area beyond the address given is unavailable to DFSORT/VSE for other uses in that phase.

3. If length of the user exit routines is specified, the routines must be:

   - Self-relocating (unmodified address constants cannot be used), or
   - Eligible for relocation by the system loader program.

   This enables DFSORT/VSE to load your routines in the most suitable
   | position to leave maximum virtual storage available for other uses.

3.11.2 Identifying User Exit Routines--Examples

Subtopics:

3.11.2.1 Example 1

```
 _____________________________________________________________________
|                                                                     |
|   MODS   PH1=(USERV,L500,E15),PH3=(USERK,L800,E31,E37)              |
|                                                                     |
|_____________________________________________________________________|
```

Specifies user exit routines for the initial sorting phase and the final merging phase. The USERV module contains a routine 500 bytes long which is executed at E15 user exit. The USERK module contains routines 800 bytes long which are executed at E31 and E37 user exits.

3.11.2.2 Example 2

```
 _____________________________________________________________________
|                                                                     |
|   MODS   PH3=(,,E31,E35)                                            |
|                                                                     |
|_____________________________________________________________________|
```

| Specifies user exit routines for the final merging phase. The user exit
| routines are already in virtual storage, so the name and loading information are not included. Preloaded user exit routines (which are executed at E31 and E35 user exits) are used only for program invoked sort, merge or copy applications.

3.12 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 file. 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/VSE run, but not both.

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 file.

Three 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.

For further details on this control statement, see "INCLUDE Control Statement" in topic 3.7 and "INCLUDE and OMIT Control Statements Notes" in topic 3.7.12.

COND

```
     _____________________________________________________________________
|    |                                                                     |
|    |  >>__COND=__ _(logical expression)_ ___________________________><  |
|    |              |_ALL_____________________|                            |
|    |              |_(ALL)___________________|                            |
|    |              |_NONE____________________|                            |
|    |              |_(NONE)__________________|                            |
|    |                                                                     |
|    |_____________________________________________________________________|
```

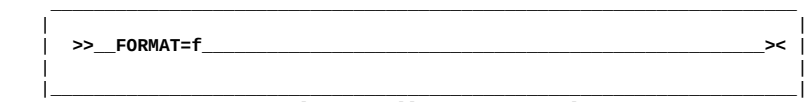
logical expression
specifies one or more relational conditions logically combined, based on fields in the input record. If the logical expression is true for a given record, the record is omitted from the output file.

| **ALL or (ALL)**
specifies that all of the input records are to
| be omitted from the output file.

| **NONE or (NONE)**
specifies that none of the input records are to
| be omitted from the output file.

Default: None: must be specified.

FORMAT



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 Figure 12 in topic 3.7.2.1. 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. If you have specified DATA=A in the INPFIL control statement, you may only use AC, AST and ASL.

Default: None: must be specified if not included in the COND operand.

Subtopics:

- 3.12.1 Omitting Records from the Output File--Example

3.12.1 Omitting Records from the Output File--Example

Subtopics:

- 3.12.1.1 Example

3.12.1.1 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:

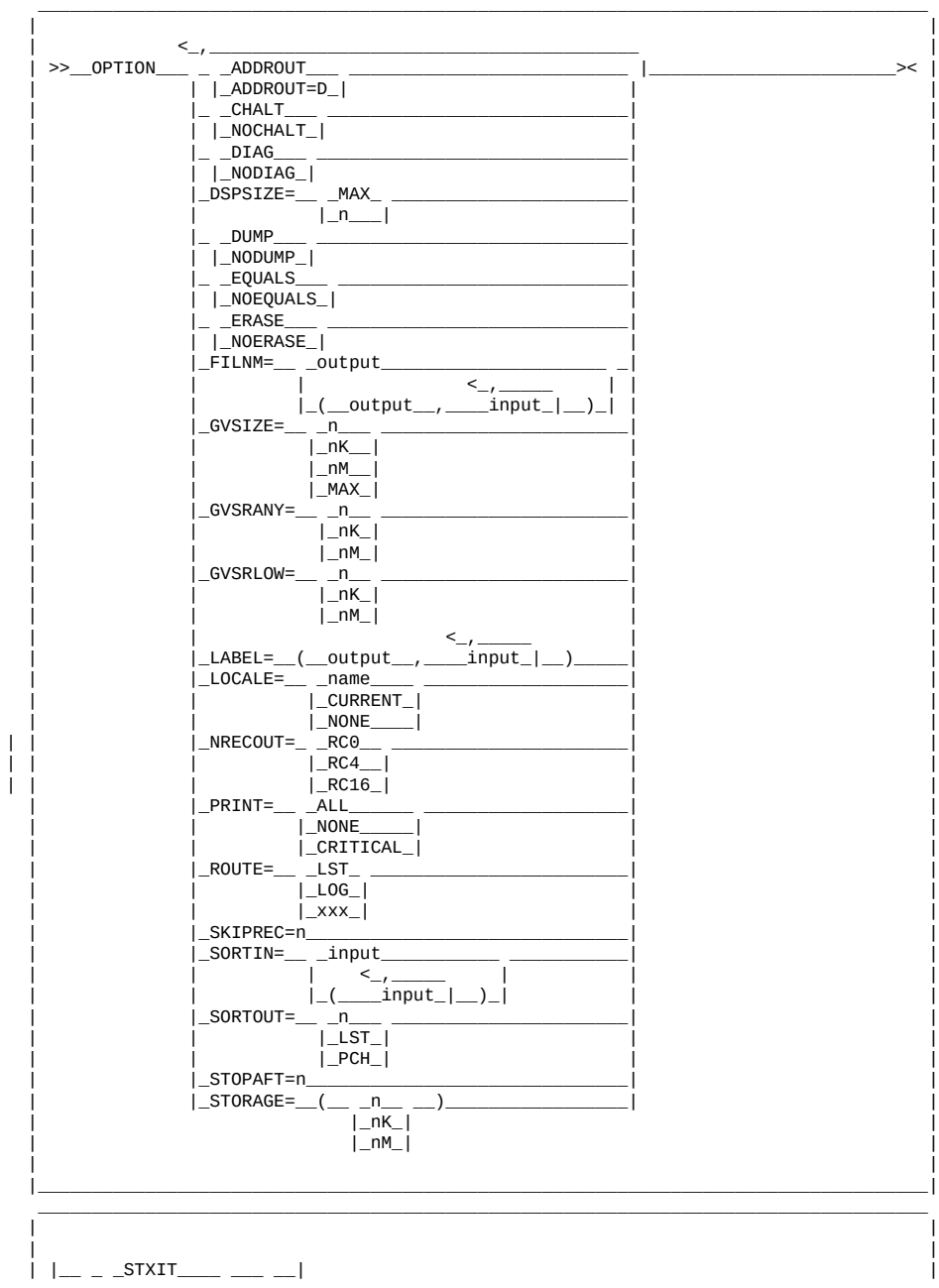
- Byte 27 contains D
    - AND

- 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" in topic 3.7.

3.13 OPTION Control Statement



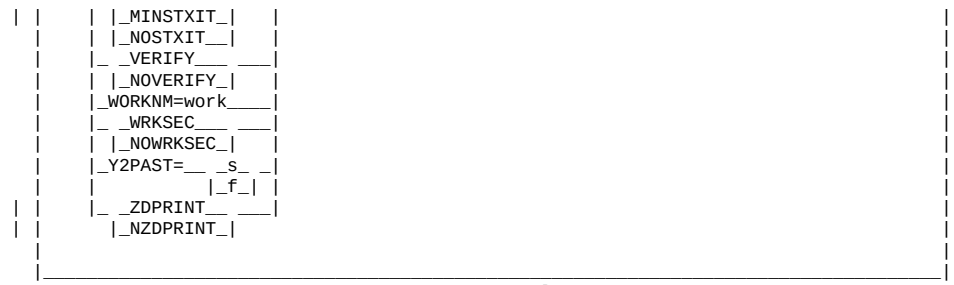
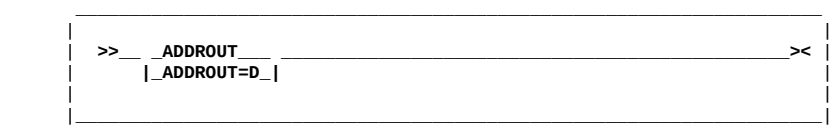


Figure 26. Syntax Diagram for the Option Control Statement

The OPTION control statement allows you to override some of the options available at installation time (such as DIAG and CHALT) and to supply other optional information (such as ADDROUT and LABEL).

ADDROUT or ADDROUT=D



ADDROUT

specifies that each output record should contain the direct access address of its corresponding sorted input record. The addresses can be used to retrieve the input records in ordered sequence. The application must be a sort, not a merge or copy.

Input can be either SAM or VSAM files. SAM files must be on CKD direct access devices; they must not be on more than one volume, and they must not be SAM ESDS files. However, ADDROUT option can be used with SAM ESDS files if they are defined to DFSORT/VSE as VSAM files; that is, if the VSAM operand is specified in the INPFIL control statement.

Input records must not be spanned. That is, the SPAN operand in the INPFIL control statement must not be specified nor must input consist of a VSAM KSDS defined with the SPANNED attribute. Addresses produced in these cases would be meaningless.

If an OUTREC, INREC, or SUM control statement is provided, ADDROUT is ignored.

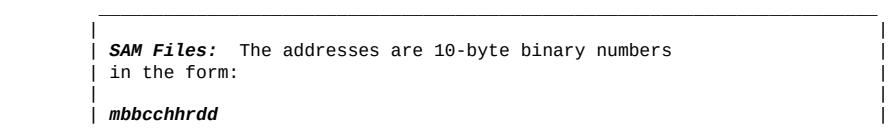
If both the ADDROUT option and the EXIT operand of the INPFIL control statement are specified, DFSORT/VSE terminates.

If the output file is on tape, the output block size must be 20 bytes or more. DFSORT/VSE ensures that the last block contains at least 18 bytes by padding with blanks as necessary.

Output records are fixed-length regardless of the type of input records. L2 and L3 values are calculated by DFSORT/VSE (see "RECORD Control Statement Notes" in topic 3.16.1, Note 5).

See also "Use Options That Improve Performance" in topic 7.3.5 for information on how ADDROUT and ADDROUT=D options can affect performance.

Figure 27 shows the formats for SAM and VSAM file addresses.



```
|                                                                      |
| where                                                                |
|                                                                      |
| m         0                                                          |
|                                                                      |
| bb        bin number (always 00)                                     |
|                                                                      |
| cc        cylinder number                                            |
|                                                                      |
| hh        head number                                                |
|                                                                      |
| r         record (block) number on the input file track              |
|                                                                      |
| dd        displacement within block: 00 for unblocked                |
|           fixed-length records, or the displacement,                 |
|           in bytes (relative to zero),                               |
|           of the record within the block.                            |
|           04 for unblocked variable-length                           |
|           records, or the displacement within the block.             |
|                                                                      |
|                                                                      |
| The output block size must be a multiple of 10.                      |
|                                                                      |
| VSAM Files:  The addresses are 5-byte binary numbers                 |
| in the form:                                                         |
|                                                                      |
| nyyyy                                                                |
|                                                                      |
| where                                                                |
|                                                                      |
| n         input file number (1-9)                                    |
|           n=1 for records from SORTIN1,                              |
|           n=2 from SORTIN2, and so on.                               |
|                                                                      |
| yyyy      relative byte address (RBA) for key sequenced data set (KSDS)|
|           or entry sequenced data set (ESDS) files, record number    |
|           for relative record data set (RRDS) files.                 |
|                                                                      |
|                                                                      |
| The output block size must be a multiple of 5.                       |
|                                                                      |
|______________________________________________________________________|
```

Figure 27. SAM and VSAM Files Address Format

ADDROUT=D

specifies that each output record should contain the direct access address of its corresponding sorted input record *followed* by the control fields of the input record (in the order in which you specified the fields in the SORT control statement, with no intervening space between them).

The details for using the ADDROUT=D option are identical to those for the ADDROUT option, with the following exceptions:

- Each output record consists of the direct access address and control fields of its corresponding sorted input record.
- The output block size must be a multiple of the L3 value as calculated by DFSORT/VSE (see "RECORD Control Statement Notes" in topic 3.16.1, Note 5).

Default: None; optional.

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.
```

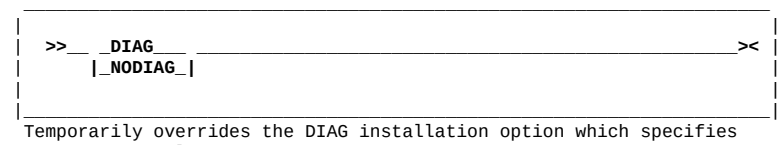
CHALT specifies that DFSORT/VSE translates character fields with formats CH and AQ using the alternate collating sequence.

NOCHALT specifies that format CH fields are not translated.

Note: If you use locale processing, you must not use CHALT. If you need alternate sequence processing for a particular field, use format AQ.

Default: DFSORT/VSE installation default.

DIAG or NODIAG




```
Temporarily overrides the DIAG installation option which specifies
whether special diagnostic messages are to be produced.
```

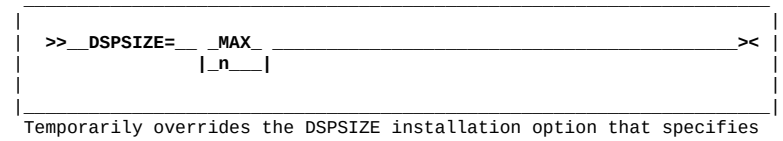
DIAG specifies that special diagnostic messages are to be produced. This option is useful when tuning the performance of sort applications. (For further details, see "Using the DIAG Option" in topic 7.8.)

NODIAG specifies that special diagnostic messages will not be produced.

| **Note:** If the DIAGINF installation option is in effect for all jobs or
| for a specific job, DIAG is forced, regardless of the options in
| effect at run time.

Default: DFSORT/VSE installation default.

DSPSIZE




```
Temporarily overrides the DSPSIZE installation option that specifies
the maximum amount of data space to be used with dataspace sorting.
```

MAX specifies that DFSORT/VSE dynamically determines the maximum amount of data space to be used for dataspace sorting. In this case, DFSORT/VSE bases its data space usage on:

- The information from the SYSDEF job control statement for creation of a single data space
- The amount of real storage

- The paging activity of the system

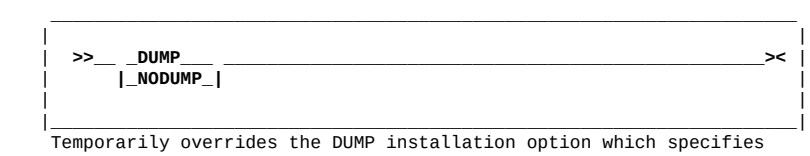
n specifies the amount, in MB, of data space to be used for dataspace sorting. n must be a value between 0 and 2048.

If n=0 or the requested storage is not available, dataspace sorting will not be used. If DFSORT/VSE determines that the value n may cause a real storage overcommitment, DFSORT/VSE will decrease the value of n.

Note: In general, DFSORT/VSE uses dataspace sorting if it is enabled. However, if the requested amount of data space is not available, DFSORT/VSE will not use dataspace sorting. If dataspace sorting is not used, DFSORT/VSE uses getvis sorting, if it is enabled. However, if the requested amount of GETVIS area is not available, DFSORT/VSE will not use getvis sorting. If neither dataspace sorting nor getvis sorting is used, DFSORT/VSE uses partition program area sorting.

Default: DFSORT/VSE installation default.

DUMP or NODUMP



Temporarily overrides the DUMP installation option which specifies
| whether a dump of virtual storage is produced whenever DFSORT/VSE
terminates abnormally.

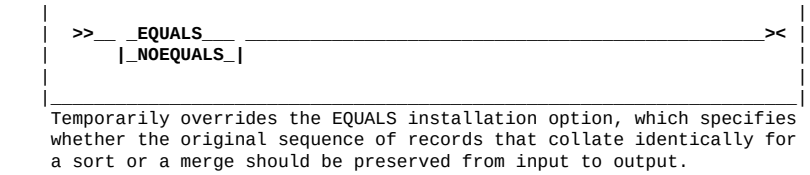
| **DUMP** specifies that a dump of virtual storage is produced. When
| DFSORT/VSE's STXIT is in effect, a dump will be produced for
| any DFSORT/VSE abnormal termination. When DFSORT/VSE's STXIT
| is not in effect, a dump will be produced for DFSORT/VSE detected errors.

NODUMP specifies that a dump is not to be produced if DFSORT/VSE terminates abnormally.

| **Note:** If the DIAGINF installation option is in effect for all jobs or
| for a specific job, DUMP is forced, regardless of the options in
| effect at run time.

Default: DFSORT/VSE installation default.

EQUALS or 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.

The EQUALS and NOEQUALS operands may be specified in the OPTION, SORT, or MERGE control statement. EQUALS or NOEQUALS operand specification in the OPTION control statement overrides EQUALS or NOEQUALS in the SORT or MERGE control statement.

For sort applications, the sequence of the output records depends upon the order of:

- The input files (SORTIN1, SORTIN2, and so on). Records that collate identically are output in the order of their file numbers. For example, records from the SORTIN1 file are output before any records that collate identically from the SORTIN2 file.
- The records in each input file.
- The records inserted by an E15 user exit routine.

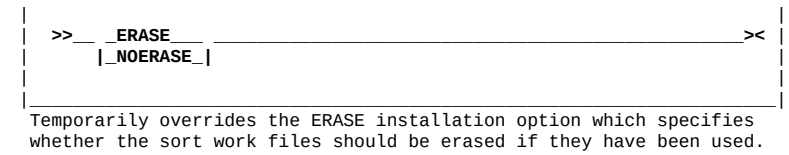
For merge applications, the sequence of the output records depends upon the order of:

- The input files (SORTIN1, SORTIN2, and so on). Records that collate identically are output in the order of their file numbers. For example, records from the SORTIN1 file are output before any records that collate identically from the SORTIN2 file.
- The records in each input file.
- The records inserted by an E32 user exit routine. Records that collate identically from an E32 user exit routine are output in the order of their file increment codes. For example, records from the file with increment code 0 are output before any records that collate identically from the file with increment code 4.

Note: Use of EQUALS can degrade performance.

Default: DFSORT/VSE installation default.

ERASE or NOERASE



Temporarily overrides the ERASE installation option which specifies whether the sort work files should be erased if they have been used.

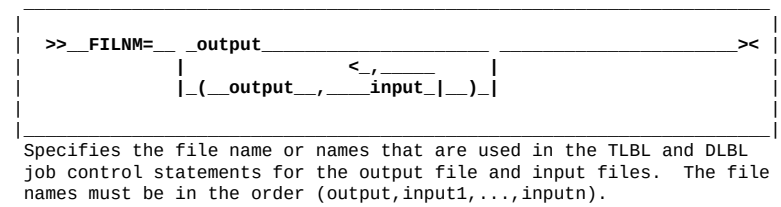
ERASE specifies that the work files will be erased if they have been used. For CKD devices, this is done by means of an end-of-file record written on every used track of each work file. For FBA devices, the used part of each work file is rewritten with zeros.

NOERASE specifies that work files are not erased.

Note: Use of ERASE can degrade performance.

Default: DFSORT/VSE installation default.

FILNM

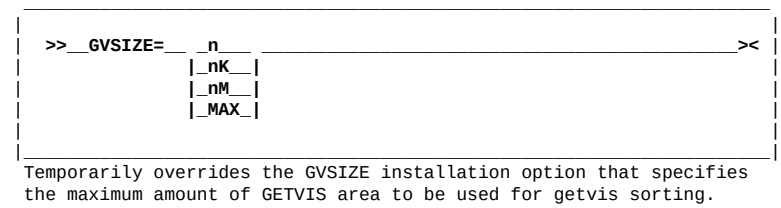


To be valid, a file name must begin with an alphabetic character. For input and output files, the name can be a maximum of seven alphanumeric characters.

For compatibility reasons, the work file name prefix can also be specified here instead of in the WORKNM operand. If a work file name prefix is supplied, it must be specified last and in the eleventh position; omitted input parameters must be indicated by commas. If a work file name prefix is specified in both operands (FILNM and WORKNM), the one specified first is overridden and the one specified second is used.

Default: If the FILNM option is omitted, the default file names are assumed for all output and input files (SORTOUT,SORTIN1-SORTIN9). See Figure 9 in topic 2.2.

GVSIZE



n specifies that n bytes of GETVIS area are to be used for getvis sorting.

nK specifies that n times 1024 bytes of GETVIS area are to be used for getvis sorting.

nM specifies that n times 1048576 bytes of GETVIS area are to be used for getvis sorting.

MAX specifies that DFSORT/VSE dynamically determines the maximum amount of GETVIS area to be used for getvis sorting. In this case, DFSORT/VSE bases its GETVIS area usage on:

- The size of the available GETVIS area
- The amount of reserved 24-bit GETVIS area and 31-bit GETVIS area for the operating system and user applications
- The amount of real storage

- The paging activity of the system

Notes:

1. n limit: 10 digits (the maximum value for n is 2047M (2146435072)).

2. The specified value of n will be decreased by DFSORT/VSE, if it is required to avoid real storage overcommitment.

3. If n is less than 32K (32768) or the requested GETVIS area is not available, getvis sorting will not be used.

4. In general, DFSORT/VSE uses dataspace sorting if it is enabled. However, if the requested amount of data space is not available, DFSORT/VSE will not use dataspace sorting. If dataspace sorting is not used, DFSORT/VSE uses getvis sorting, if it is enabled. However, if the requested amount of GETVIS area is not available, DFSORT/VSE will not use getvis sorting. If neither dataspace sorting nor getvis sorting is used, DFSORT/VSE uses partition program area sorting.

Default: DFSORT/VSE installation default.

GVSRANY

Temporarily overrides the GVSRANY installation option that specifies the amount of 31-bit GETVIS area to be reserved for the operating system and user applications when the GVSIZE=MAX option is in effect.

n specifies that n bytes of 31-bit GETVIS area are to be reserved.

nK specifies that n times 1024 bytes of 31-bit GETVIS area are to be reserved.

nM specifies that n times 1048576 bytes of 31-bit GETVIS area are to be reserved.

Notes:

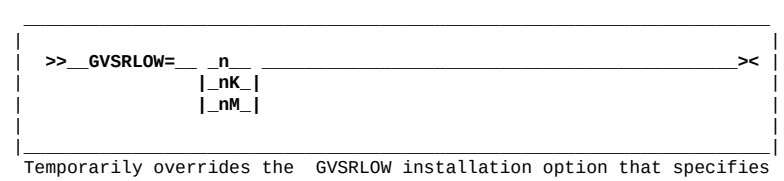
1. n limit: 10 digits.

2. n must be a value between 32K (32768) and 2047M (2146435072).

3. If GVSIZE=MAX option is not in effect, the GVSRANY operand will be ignored.

Default: DFSORT/VSE installation default.

GVSRLOW



n specifies that n bytes of 24-bit GETVIS area are to be reserved.

nK specifies that n times 1024 bytes of 24-bit GETVIS area are to be reserved.

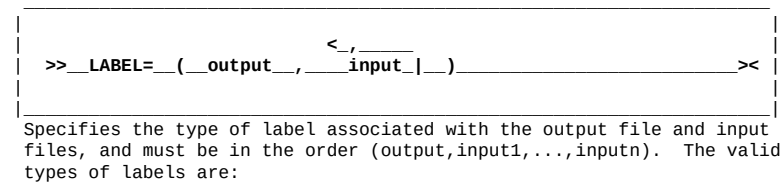
nM specifies that n times 1048576 bytes of 24-bit GETVIS area are to be reserved.

Notes:

1. n limit: 10 digits.
2. n must be a value between 64K (65536) and 16M (16777216).
3. If GVSIZE=MAX option is not in effect, the GVSRLOW operand will be ignored.

Default: DFSORT/VSE installation default.

LABEL



N specifies nonstandard labels (including user standard labels).

S specifies standard labels.

U specifies unlabeled.

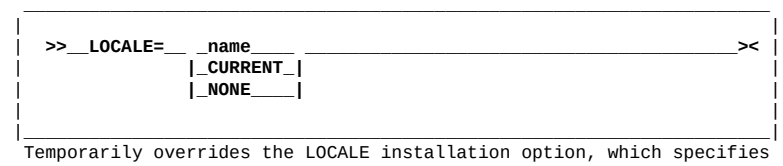
If the LABEL option is omitted, standard labels are assumed for all input and output files.

If you specify N for nonstandard label files (including standard labels with additional user header or trailer labels), you must provide your routines at the label checking user exits to open and close the files, and process the labels, as described in Chapter 4, "Using Your Own User Exit Routines" in topic 4.0.

A positional parameter can be replaced by a comma if the default value is applicable.

Default: S.

LOCALE



Temporarily overrides the LOCALE installation option, which specifies whether locale processing is to be used and, if so, designates the active locale.

DFSORT/VSE's collating behavior can be modified according to your cultural environment. Your cultural environment is defined to DFSORT/VSE 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/VSE'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/VSE 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. Up to 32 characters can be specified for the descriptive locale name. The locale names themselves are not case sensitive. See *LE/VSE Library* for a complete description of the naming conventions for locales.

You can use IBM-supplied and user-defined locales. See Appendix E, "Locales Supplied with C Run-Time Library" in topic E.0 for examples of some IBM-supplied locales.

The state of the active locale prior to DFSORT/VSE being entered will be restored when DFSORT/VSE completes processing.

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

NONE
specifies that locale processing is not to be used. DFSORT/VSE will use the binary encoding for collating and comparing.

Notes:

1. If you require locale processing, then you must install one of the following:

VSE/ESA 1.4 through 2.1 and the Language Environment for VSE/ESA 1.4.0 (with the C-language run-time support installed) or a subsequent release of Language Environment for VSE/ESA

or

VSE/ESA 2.2 (with the C-language run-time support installed from the VSE/ESA Extended Base Products tape) or a subsequent release.

2. To use an IBM-supplied locale, DFSORT/VSE must have access to the sublibrary containing the LE/VSE dynamically loadable routines and compiled locale modules. For example, this sublibrary might be called PRD2.SCEEBASE. If you are not sure of where the compiled locale modules are installed at your location, contact your system administrator. To use a user-defined locale, DFSORT/VSE must have access to the sublibrary containing the locale.

3. If locale processing is requested:

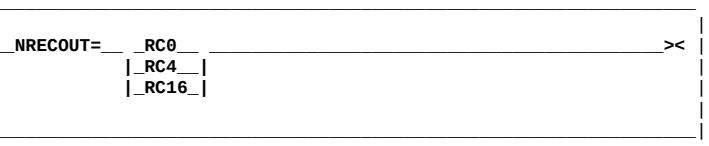
- CHALT must not be used.
- INREC must not be used.

4. Locale processing for SORT, MERGE, INCLUDE, and OMIT functions can improve performance relative to applications which perform preprocessing or postprocessing of data, or both to produce the desired collating results. However, locale processing should only be used when required since it can show degraded performance relative to collation using character encoding values.

5. The amount of storage needed to use locale processing for DFSORT/VSE can vary significantly according to the needs of LE/VSE, the locale used, and so on. 2 MB of the partition GETVIS area should be enough in most cases. The storage needed by LE/VSE will be reduced if LE/VSE phases CEEBINIT, CEEPLPKA, and CEEV003 were placed into the SVA. See *LE/VSE Library* for more information. If you are planning to use locale processing with getvis sorting, then increase the amount of the partition GETVIS area by the GVSIZE value.

Default: DFSORT/VSE installation default.

| **NRECOUT**



| Temporarily overrides the NRECOUT installation option which specifies
| the actions to be taken by DFSORT/VSE when it does not write any
| records to the output file.

| **RC0** specifies that DFSORT/VSE should issue message ILU400I, set a
| return code of 0 and continue processing.

| **RC4** specifies that DFSORT/VSE should issue message ILU400I, set a

| return code of 4 and continue processing.

| **RC16** specifies that DFSORT/VSE should issue message ILU401A, set a
| return code of 16 and terminate processing.

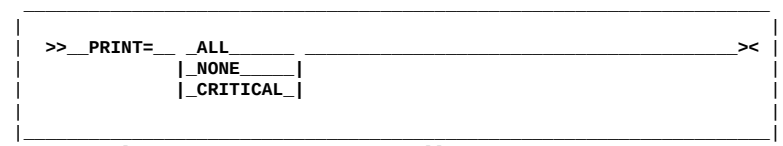
| **Notes:**

| 1. The return code of 0 or 4 set when DFSORT/VSE does not write any
| records to the output file can be overridden by a higher return
| code set for some other reason.

| 2. If OUTFIL EXIT is specified, NRECOUT is ignored.

Default: DFSORT/VSE installation default.

PRINT



Temporarily overrides the PRINT installation option that specifies which messages are to be produced by DFSORT/VSE.

ALL specifies that all DFSORT/VSE messages are to be produced, including error and end-of-application messages, various size calculation messages, and other informational messages.

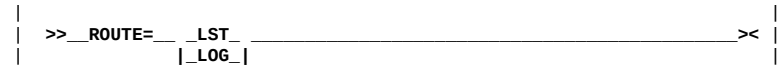
NONE specifies that DFSORT/VSE messages are not to be produced. The NONE option forces the use of the NODUMP option.

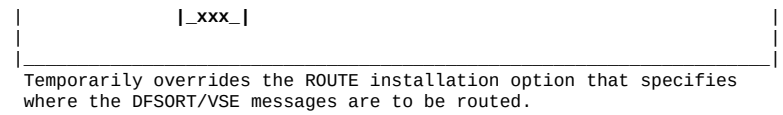
CRITICAL specifies that only critical error messages are to be produced; that is, error messages signaling conditions that can cause DFSORT/VSE to terminate.

| **Note:** If the DIAGINF installation option is in effect for all jobs or
| for a specific job, PRINT=ALL is forced, regardless of the options in
| effect at run time.

Default: DFSORT/VSE installation default.

ROUTE





LST specifies that all DFSORT/VSE messages are to be routed to SYSLST. In addition, critical messages are to be routed to the system console.

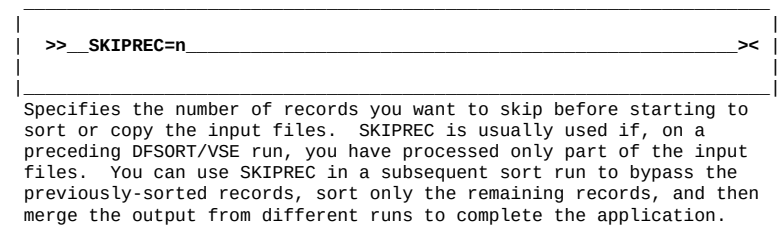
LOG specifies that all DFSORT/VSE messages are to be routed to the system console.

xxx specifies that all DFSORT/VSE messages are to be routed to SYSxxx, where xxx can be any valid SYS number between 000 and 221. Critical messages will also be routed to SYSLOG, but the system dump (if any) is produced on SYSLST. ROUTE=xxx is in effect for program invoked DFSORT/VSE applications only. If ROUTE=xxx is specified for a directly invoked DFSORT/VSE application, it will be ignored and the default ROUTE value (LST or LOG) will be used instead.

| **Note:** If the DIAGINF installation option is in effect for all jobs or
| for a specific job, ROUTE=LST is forced, regardless of the options in
| effect at run time.

Default: DFSORT/VSE installation default.

SKIPREC



SKIPREC may be specified in the OPTION, SORT, or MERGE control statement. The SKIPREC specification in the OPTION control statement overrides any SKIPREC specification on the SORT or MERGE control statement.

n
specifies the number of records to be skipped.

Notes:

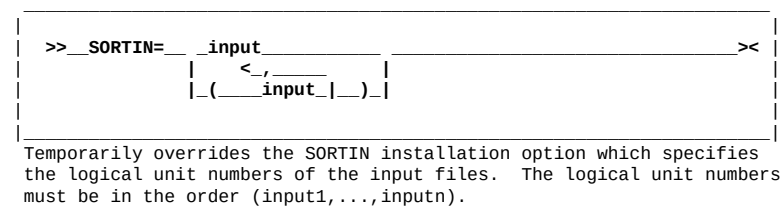
1. SKIPREC applies only to records read from SORTIN (not from an E15 user exit for a sort application or from an E32 user exit for a copy application).

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

3. n limit: 10 digits (the maximum value for n is 2147483647).

Default: None; optional.

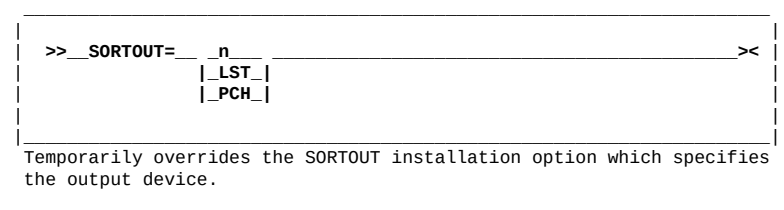
SORTIN



The value for a logical unit number can be any number from 1 through 221, or a comma (for the default logical unit number). The SORTIN operand is only needed for tape files.

Default: DFSORT/VSE installation default.

SORTOUT



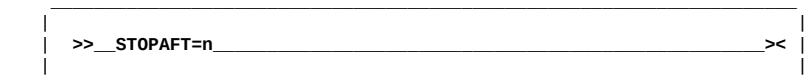
n
specifies the logical unit number of the output device. The value for the logical unit number can be any number from 1 through 221. The logical unit number is only needed for tape, printer, and punch devices.

LST
specifies that output is to be written to SYSLST.

PCH
specifies that output is to be written to SYSPCH.

Default: DFSORT/VSE installation default.

STOPAFT



Specifies the maximum number of records you want accepted for sorting or copying. For sorting, that number includes the records read from SORTIN or inserted by the E15 user exit routine and not deleted by SKIPREC, the E15 user exit routine, or INCLUDE or OMIT function. For copying, that number includes records read from SORTIN or inserted by the E32 user exit routine and not deleted by SKIPREC, INCLUDE, or OMIT functions.

When n records have been accepted, no more records are read from SORTIN; an E15 user exit routine for a sort application or an E32 user exit routine for a copy application continues to be entered as if end of file was encountered until a return code of 8 is sent, but no more records are accepted for processing. If end of file is encountered for the last input file before n records are accepted, only those records accepted up to that point are sorted or copied.

STOPAFT may be specified in the OPTION, SORT, or MERGE control statement. STOPAFT specification in the OPTION control statement overrides any STOPAFT specification in the SORT or MERGE control statement.

n
specifies the maximum number of records to be accepted.

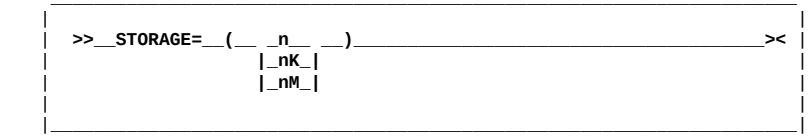
Notes:

1. If STOPAFT=0 is in effect, STOPAFT is not used.

2. n limit: 10 digits (the maximum value for n is 2147483647).

Default: None; optional.

STORAGE



Temporarily overrides the STORAGE installation option, which specifies the amount of virtual storage (the partition program area) to be used, if available, by DFSORT/VSE.

n specifies that n bytes are to be used.

nK specifies that n times 1024 bytes are to be used.

nM specifies that n times 1048576 bytes are to be used.

Notes:

1. n must be a value between 32K (32768) and 16M (16777216).

2. n limit: 10 digits.

3. *Minimum:* The value specified for n must not be less than 32K.

4. *Maximum:* The value specified for n must not be greater than the smallest of the following:

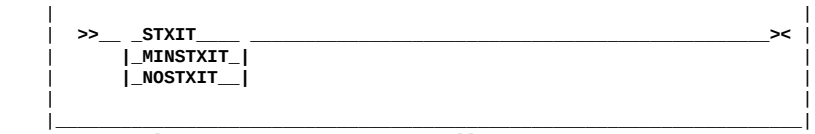
- The difference between the DFSORT/VSE load point and any calling program return address (in register 14), which is in the partition program area.
- The difference between the DFSORT/VSE load point and any preloaded user exit routine address, which is in the partition program area (above the DFSORT/VSE load point).

5. If *n* is greater than the maximum permitted value, it is ignored and the maximum is used instead.

6. If the final value for storage available to DFSORT/VSE is less than 32K, DFSORT/VSE terminates. A given application can require more than the minimum of 32K. For additional considerations, see "OPTION Control Statement Notes" in topic 3.13.1, Chapter 7, "Improving Efficiency" in topic 7.0, and Appendix A, "Estimating Storage Requirements" in topic A.0.

Default: DFSORT/VSE installation default.

| **STXIT, MINSTXIT, or NOSTXIT**



Temporarily overrides the STXIT installation option which specifies
| whether DFSORT/VSE should use its STXIT routine for abend recovery
| processing.

| **STXIT** specifies that DFSORT/VSE should use its STXIT routine for
| abend recovery processing, restoring its STXIT every time
| control is returned from a user exit routine.

| DFSORT/VSE will save the caller's STXIT, establish its
| STXIT, restore its STXIT each time a user exit routine (if
| any) returns control to DFSORT/VSE, and restore the caller's
| STXIT before returning control to the caller.

| If an abend occurs, DFSORT/VSE will return control to the

| caller with a return code of 16 (unsuccessful completion) at
| the end of its abend recovery processing.

| Performance can be degraded when COBOL or PL/I programs
| invoke DFSORT/VSE and use E15/E35 user exit routines to
| process records.

| **MINSTXIT** specifies that DFSORT/VSE should use its STXIT routine for
| abend recovery processing and not restore its STXIT every
| time control is returned from a user exit routine.

| DFSORT/VSE will save the caller's STXIT, establish its
| STXIT, and restore the caller's STXIT before returning
| control to the caller.

| If an abend occurs, DFSORT/VSE will return control to the
| caller with a return code of 16 (unsuccessful completion) at
| the end of its abend recovery processing.

| DFSORT/VSE will not restore its STXIT when it receives
| control from the user exit routine (if any), so if the user
| exit routine established its own STXIT, it will remain in
| effect even though DFSORT/VSE has control. Then if an abend
| occurs, control will be returned to the user exit routine's
| STXIT routine and therefore DFSORT/VSE will not be able to
| do its abend recovery processing.

| Unlike STXIT, MINSTXIT will not degrade performance when
| COBOL or PL/I programs invoke DFSORT/VSE and use E15/E35
| user exit routines to process records.

NOSTXIT specifies that DFSORT/VSE should not use its STXIT routine. DFSORT/VSE will not be able to do its abend recovery processing.

| **Notes:**

| 1. If you have COBOL or PL/I programs that invoke DFSORT/VSE and use
| E15/E35 user exit routines to process records, we recommend using
| MINSTXIT.

| 2. If you do not have COBOL or PL/I programs that invoke DFSORT/VSE
| and use E15/E35 user exit routines to process records, we
| recommend using STXIT.

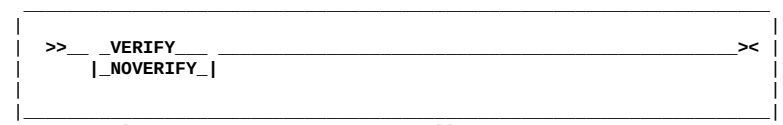
3. See Appendix D, "DFSORT/VSE Abend Processing" in topic D.0 for more information on the DFSORT/VSE STXIT routine.

| 4. If an E31 user exit routine is used and both MINSTXIT and CKPT are

| in effect, DFSORT/VSE's STXIT is restored when the user exit
| routine returns control to DFSORT/VSE.

Default: DFSORT/VSE installation default.

VERIFY or NOVERIFY



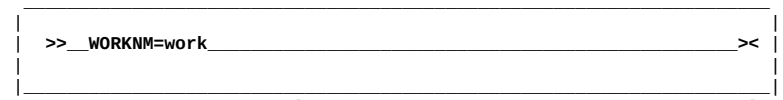
Temporarily overrides the VERIFY installation option which specifies whether, when a direct access device is used to store the output file, each block is checked to ensure that it was written correctly.

VERIFY specifies that, when a direct access device is used to store the output file, each block is checked to ensure that it was written correctly. VERIFY is ignored for a VSAM output file. Its use degrades DFSORT/VSE performance.

NOVERIFY specifies that blocks are not checked.

Default: DFSORT/VSE installation default.

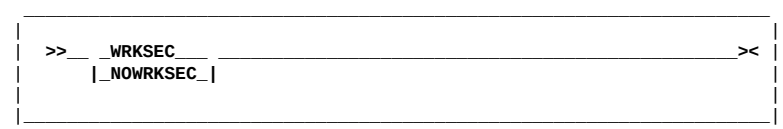
WORKNM



Specifies the first four letters of the name in the DLBL job control statement for the work files (work file name prefix). This work file name prefix replaces SORT in the names SORTWK1, SORTWK2, and so on.

Default: SORT.

WRKSEC or NOWRKSEC



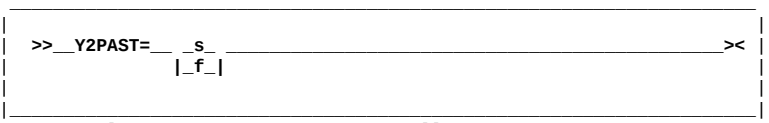
| Temporarily overrides the WRKSEC installation option that specifies
| whether secondary allocation for SAM ESDS or SD work files is to be used.

| **WRKSEC** specifies that secondary allocation for SAM ESDS or SD work
| files is to be used.

| **NOWRKSEC** specifies that secondary allocation for SAM ESDS for SD work
| files is not to be used.

Default: DFSORT/VSE installation default.

Y2PAST



Temporarily overrides the Y2PAST installation option, which specifies the sliding (s) or fixed (f) century window. The century window is

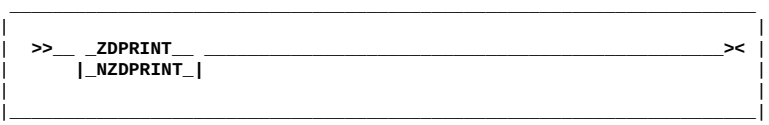
| used with DFSORT/VSE's Y2C, Y2Z, Y2S, Y2P, Y2D, and Y2B formats to correctly interpret two-digit year data values as four-digit year data values.

s specifies the number of years DFSORT/VSE 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.

Default: DFSORT/VSE installation default.

| **ZDPRINT or NZDPRINT**



| Temporarily overrides the ZDPRINT installation option that specifies
| whether positive zoned decimal (ZD) fields resulting from summation
| 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" in topic 3.18 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.

| *Default:* DFSORT/VSE installation default.

Subtopics:

3.13.1 OPTION Control Statement Notes

1. If you want to use the PRINT operand, ROUTE operand, or both you are advised to put the OPTION control statement before all other DFSORT/VSE control statements and to put those operands on the first line of the control statement (that is, not on a continuation line).

2. When running in virtual mode, if neither the SIZE parameter of the EXEC job control statement or the SIZE job control statement nor the STORAGE option of the OPTION control statement is specified, the whole partition program area is used for DFSORT/VSE by default.

3. Specifying the ERASE option provides data security when sorting files which contain sensitive information, but it increases execution time.

   If the CKPT operand has been specified in the SORT control statement, and if DFSORT/VSE terminates abnormally, the ERASE option will be ignored.

   The ERASE option has no effect on the DSCB entries in the VTOC. If the work files are given a nonzero retention period, the DSCBs will remain in the VTOC after DFSORT/VSE has completed, even though the work file areas themselves have been cleared or erased.

3.13.2 Specifying DFSORT/VSE Options--Examples

Subtopics:

3.13.2.1 Example 1

```
 _____________________________________________________________________
|                                                                     |
|   OPTION PRINT=CRITICAL,ROUTE=LOG,STORAGE=49152,LABEL=(,N,N)        |
|                                                                     |
|_____________________________________________________________________|
```

PRINT Only critical messages will be produced.

ROUTE The messages will be routed to the system console.

| **STORAGE** 49152 bytes of virtual storage (partition program area) will be used, if available, by DFSORT/VSE.

LABEL The output file is to have a standard label (by default) and the two input files have nonstandard labels.

3.13.2.2 Example 2

```
   OPTION STORAGE=64K,VERIFY,ERASE,
          SORTIN=(003,004)
```

STORAGE 64 KB of virtual storage (partition program area) will be used, if available, by DFSORT/VSE.

VERIFY When producing the output file, each block is checked to ensure that it has been written correctly.

ERASE The work files used by DFSORT/VSE are to be erased upon completion of the sort application.

SORTIN Specifies two logical unit numbers 003 and 004 of input files.

3.13.2.3 Example 3

```
   OPTION FILNM=(SORT2,IN,,IN3),WORKNM=JOB2
```

If we assume FILES=3 is specified in the SORT control statement then, in this example, the output file is given the name SORT2, the input files are named IN, SORTIN2 (by default), and IN3, and the work files have the names JOB2WK1-JOB2WKm.

Furthermore, if we assume that a multiextent direct access work file has been specified in the DLBL job control statement (DA), only the logical unit number of the first extent is necessary. In this case, the WORK operand of the SORT statement must be specified or defaulted as WORK=DA.

If the work file name prefix is specified in the FILNM operand instead of in the WORKNM operand, then the work file name prefix must be the 11th value and missing values must be indicated by commas:

```
   OPTION FILNM=(SORT2,IN,,IN3,,,,,,,JOB2)
```

3.13.2.4 Example 4

```
   SORT FIELDS=(20,8,CH,D,8,4,FI,A),WORK=1,FILES=1
   RECORD TYPE=F,LENGTH=80
   OPTION ADDROUT=D
   INPFIL BLKSIZE=800
   OUTFIL BLKSIZE=220
```

This example illustrates the use of the ADDROUT=D option. The output record will have a record size of 22 bytes and will look as follows:

Position Contents

1-10
: mbbcchhrdd (direct access address of the corresponding sorted input record)

11-18
: Input positions 20 through 27 (first control field--major control field)

19-22
: Input positions 8 through 11 (second control field--minor control field)

3.13.2.5 Example 5

```
   OPTION DSPSIZE=2,GVSIZE=2048K
```

This example illustrates the use of DSPSIZE and GVSIZE options:

DSPSIZE If requested (2 MB) data space is available, dataspace sorting is used.

GVSIZE If requested data space is not available and requested (2048 KB) GETVIS area is available, getvis sorting is used.

If requested data space and GETVIS area are not available, partition program area is used.

3.13.2.6 Example 6

```
 OPTION NOSTXIT
```

This example illustrates the use of the NOSTXIT option. DFSORT/VSE will not use its STXIT routine.

3.13.2.7 Example 7

```
 OPTION STXIT
```

This example illustrates the use of the STXIT option. DFSORT/VSE will use its STXIT routine.

3.13.2.8 Example 8

```
 OPTION GVSIZE=MAX,GVSRLOW=512K,GVSRANY=5M
```

GVSIZE DFSORT/VSE dynamically determines the maximum amount of GETVIS area to be used for getvis sorting.

GVSRLOW 512 KB of 24-bit GETVIS area are reserved for the operating system and user applications.

GVSRANY 5 MB of 31-bit GETVIS area are reserved for the operating system and user applications.

3.13.2.9 Example 9

```
OPTION SKIPREC=1000,STOPAFT=5000
```

SKIPREC 1000 records of the input files are skipped before starting of processing (sorting or copying).

STOPAFT 5000 records of the input files, the files inserted by user exit routines, or both are accepted for processing (sorting or copying).

3.13.2.10 Example 10

```
OPTION WRKSEC
```

| **WRKSEC** DFSORT/VSE will use secondary allocation for SAM ESDS or SD work
| files.

3.13.2.11 Example 11

```
OPTION NOWRKSEC
```

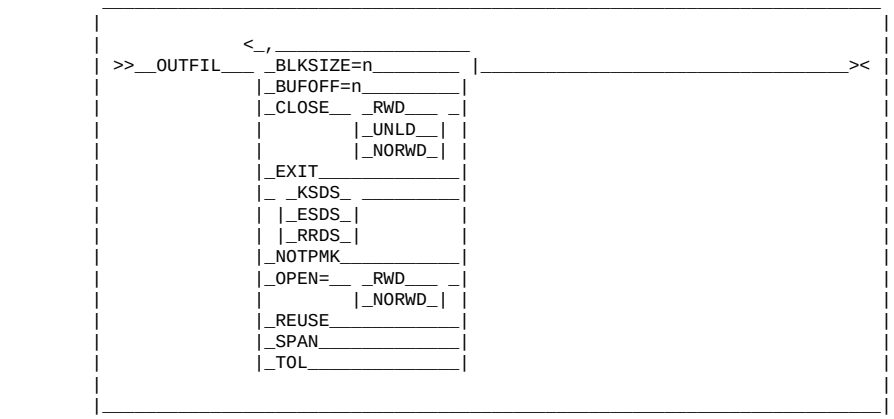
| **NOWRKSEC** DFSORT/VSE will not use secondary allocation for SAM ESDS or
| SD work files.

| 3.13.2.12 Example 12

```
   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/VSE
| uses an F sign for the last digit.

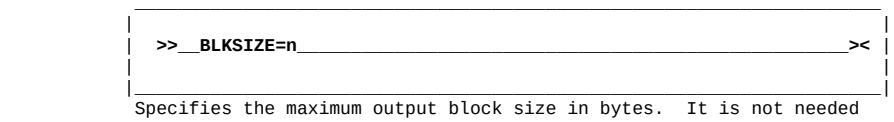
3.14 OUTFIL Control Statement



The OUTFIL control statement defines the output file to DFSORT/VSE and specifies the procedure to be followed when a tape output file is opened or closed.

The OUTFIL control statement is required only if default values for output processing are not applicable.

BLKSIZE



Specifies the maximum output block size in bytes. It is not needed if:

- The output file is a VSAM file, accessed as VSAM, or
- The output file is to be unblocked.

Otherwise, it must be supplied because DFSORT/VSE does not support the BLKSIZE parameter of the DLBL job control statement.

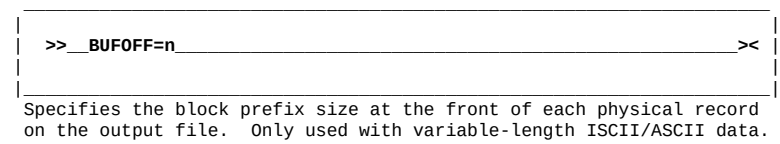
When specified for variable-length records, the BLKSIZE value must include the block descriptor word (4 bytes). For ISCII/ASCII data, it must include the BUFOFF value.

If INREC, OUTREC, ADDROUT, or ADDROUT=D is specified, the BLKSIZE value you specify must be consistent with the effective value of L3, as calculated by DFSORT/VSE (see "RECORD Control Statement Notes" in topic 3.16.1, Notes 5 and 6).

Default:

- Fixed-length records: L3 value (specified from the RECORD control statement or calculated)
- Variable-length records: L3 value (specified from the RECORD control statement or calculated) plus 4 bytes
- ISCII/ASCII variable-length records: L3 value (specified from the RECORD control statement or calculated) plus BUFOFF value.

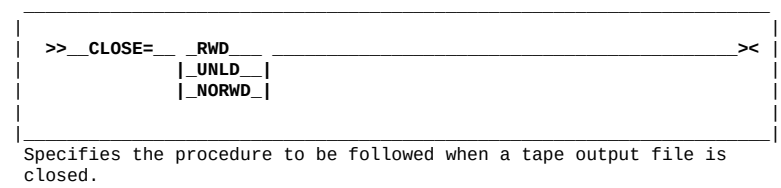
BUFOFF



The value of n may only be 0 or 4.

Default: 0.

CLOSE



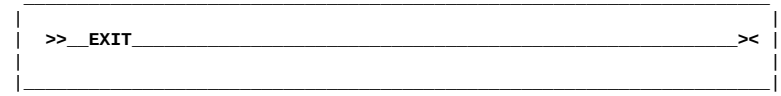
RWD specifies that the tape is to be rewound.

UNLD specifies that the tape is to be rewound and unloaded.

NORWD specifies that the tape is not to be rewound.

Default: RWD.

EXIT



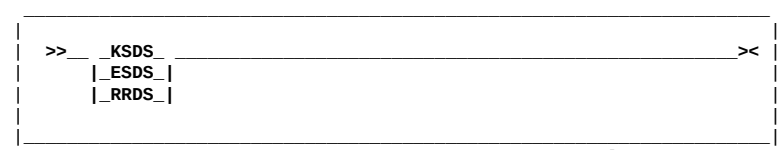
Specifies that, instead of DFSORT/VSE, your own E35 user exit routine will take responsibility for the output file. You must:

- Activate your E35 user exit by specifying its name in the MODS control statement.
- Provide an E35 user exit routine which will receive each output record from DFSORT/VSE. This routine must take complete responsibility for the output file.

When EXIT is specified, all other OUTFIL control statement operands are ignored.

Default: None; optional.

KSDS or ESDS or RRDS



Any one of these operands specifies that the output file is a VSAM file. If none of these operands are specified, a SAM output file is assumed. It may also be used to access a SAM ESDS file with VSAM access.

Any of these operands overrides all other OUTFIL control statement operands except TOL, REUSE, and EXIT. The EXIT operand overrides KSDS, ESDS, and RRDS.

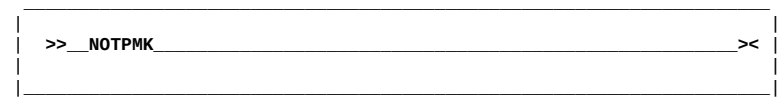
KSDS specifies that the VSAM file is to be a key sequenced file. The records must have been sorted into ascending key sequence on the primary VSAM key.

ESDS specifies that the VSAM file is to be an entry sequenced file.

RRDS specifies that the VSAM file is to be a relative record file.

Default: None; optional.

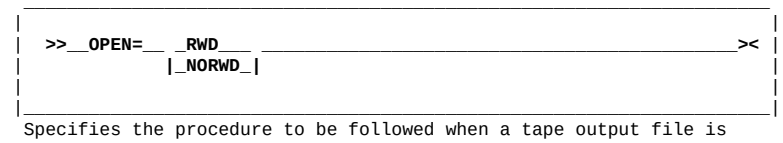
NOTPMK



Specifies that no tape mark is to be written before the first data record on each volume in the output file. This operand can only be specified for an unlabeled tape output file.

Default: None; optional.

OPEN



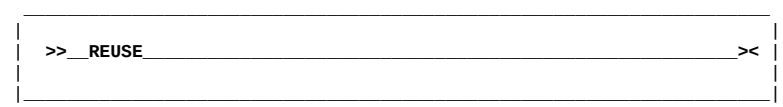
Specifies the procedure to be followed when a tape output file is opened.

RWD specifies that the tape should be rewound before being written.

NORWD specifies that the tape should not be rewound before being written.

Default: RWD.

REUSE

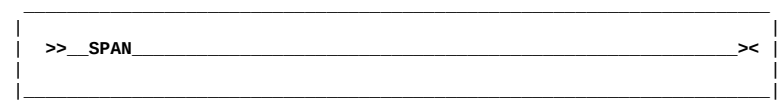


Specifies that you want to write over an existing non-empty VSAM file defined with the REUSE attribute. For a SAM file, this operand is ignored.

Note: The DISP parameter in the output DLBL job control statement will override the REUSE operand if they conflict.

Default: None; optional.

SPAN

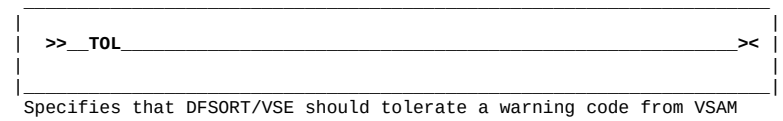


Specifies that the output SAM file is to consist of spanned variable-length EBCDIC records. The RECORD control statement must specify TYPE=V.

If a BLKSIZE operand is also provided, the records will be spanned blocked. Otherwise, they will be spanned unblocked, with a default block size of the maximum allowed for the output device.

Default: None; optional.

TOL



Specifies that DFSORT/VSE should tolerate a warning code from VSAM when opening a VSAM output file. It is only valid for a VSAM output

```
file.  See "OUTFIL Control Statement Notes" in topic 3.14.1.
```

Default: None; optional.

Subtopics:

- 3.14.1 OUTFIL Control Statement Notes
- 3.14.2 Defining the Output File--Examples

3.14.1 OUTFIL Control Statement Notes

1. Any operands which precede EXIT in the OUTFIL control statement are checked only for syntax errors and flagged. Operands which follow EXIT are also checked for syntax errors and flagged but all values are ignored.

2. The presence or absence of the EXIT operand affects the default symbolic unit names for the other files used by DFSORT/VSE. See Figure 9 in topic 2.2 for details.

3. A VSAM output file must be previously created with the VSAM access method services utility program. See *VSE/VSAM Library* for a discussion of a VSAM organization, and for instructions on how to define a VSAM file.

4. Before writing variable-length records to a VSAM output file, DFSORT/VSE removes the 4-byte record descriptor word (RDW). Thus, those user-written routines at E35 user exit which pass variable-length records back to DFSORT/VSE, do not need to be designed separately for handling VSAM and SAM files.

5. If TOL is not specified for a VSAM output file, a warning return code from VSAM will normally be recognized as an error by DFSORT/VSE; an error message will be issued, and DFSORT/VSE will terminate.

   If TOL is specified, DFSORT/VSE writes the output file without repairing it, and no error message will be issued. Examples of conditions that produce warning messages are: 'NOT PROPERLY CLOSED' or 'SYSTEM TIME STAMPS OF DATA AND INDEX DO NOT MATCH'.

   A *critical* return code from VSAM will always cause DFSORT/VSE to terminate, regardless of whether TOL has been specified.

6. For a SAM ESDS file, the BLKSIZE operand refers to the maximum logical block size. That is, it should match the RECORDSIZE (maximum) operand specified in the DEFINE command for the file when it was created with VSAM Access Method Services (AMS). If the file is to be implicitly defined, VSAM will use this value to calculate the control interval size for the file.

7. For FBA devices, DFSORT/VSE will write the maximum number of records possible within the control interval specified or defaulted, which can result in a short block within the control interval.

3.14.2 Defining the Output File--Examples

Subtopics:

- 3.14.2.1 Example 1
- 3.14.2.2 Example 2

3.14.2.1 Example 1

```
OUTFIL BLKSIZE=300,CLOSE=UNLD,BUFOFF=4
```

BLKSIZE The block size is 300 bytes.

CLOSE The output tape will be rewound (by default) before writing begins and it will be rewound and unloaded at end of file.

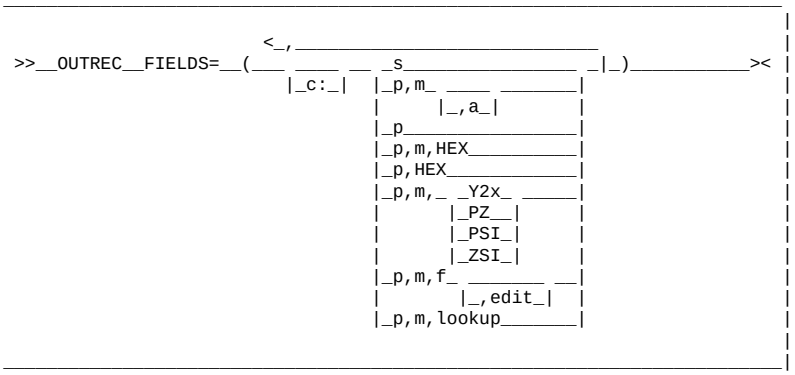
BUFOFF Data records are variable-length ISCII/ASCII, and each output block will have a block prefix of length 4.

3.14.2.2 Example 2

```
OUTFIL  EXIT
```

EXIT DFSORT/VSE provides the output records, one at a time, to be handled by the E35 user exit routine.

3.15 OUTREC Control Statement



The OUTREC control statement allows you to reformat the input records
| before they are output. That is, to define which parts of the input
| record are included in the reformatted output record, in what order they

| are to appear, how they are to be aligned, and how they are to be edited
| or changed. You can also insert separators before, between, and after the
| input fields.

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

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

You can transform two-digit character, zoned decimal, packed decimal or decimal year data to correct four-digit character year data using the century window.

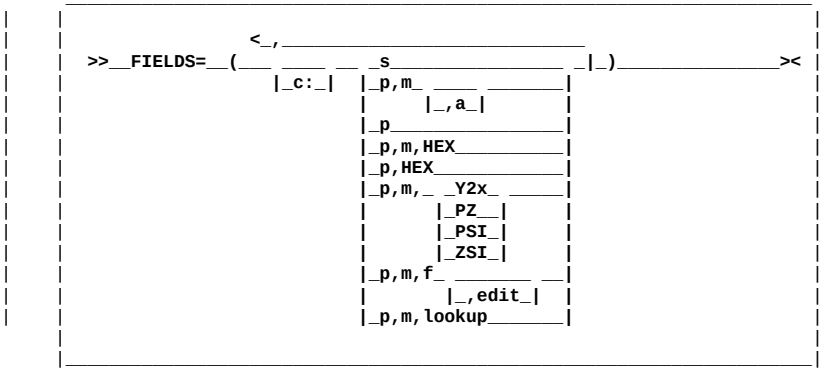
| The OUTREC control statement also supports these output file tasks:

- | 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).

For information concerning the interaction of INREC and OUTREC control statements, see "OUTREC Control Statement Notes" in topic 3.15.1 and Chapter 7, "Improving Efficiency" in topic 7.0.

FIELDS



| Specifies which parts of the input record are included in the
| reformatted output record, in what order the parts appear, how they
| are aligned, and how they are edited or changed. You can also insert
| separators before, between, and after the input fields.

| **c:** specifies the column in which the first position of the
| associated input or separation field is to appear,
| relative to the start of the reformatted output record.
| Count the RDW (variable-length output records only) 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. If you specify DATA=A in the INPFIL
| control statement, unused space is padded with ISCII/ASCII
| blanks (X'20'). The following rules apply:

- | c must be a number between 1 and 65535.
- | 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 (,) for
  | continuation to another line.

| Both valid and invalid examples are shown in Figure 28.

Figure 28. 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	65536:21,8	Invalid -- column value must be less than 65536.
Invalid	5:10:2,5	Column values cannot be adjacent.
Invalid	20,10,6:C'AB'	Column value overlaps previous field.

s specifies a separation field to be inserted into the reformatted output record in the position you code it relative to the other fields. It can be specified before or after any input field. Consecutive separation fields may be specified. For variable-length records, s must not be specified before the first input field (the record descriptor word) or after the variable part of the input record, and must meet the requirement to include at least one byte of fixed data in the reformatted output record. Permissible values are nX, nZ, nC'xx...x', and nX'yy...yy'.

nX

Blank separation. n bytes of EBCDIC blanks (X'40') are inserted in the reformatted output records. n can range from 1 to 256. If n is omitted, 1 is used.

If you specify DATA=A in the INPFIL control statement, n bytes of ISCII/ASCII blanks (X'20') are inserted in the reformatted output record.

See Figure 24 in topic 3.9 for examples of valid and invalid blank separation.

nZ

Binary zero separation. n bytes of binary zeros (X'00') are inserted in the reformatted output records. n can range from 1 to 256. If n is omitted, 1 is used.

See Figure 25 in topic 3.9 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 256. If n is omitted, 1 is used. x can be any EBCDIC character. Specify from 1 to 256 characters.

If you specify DATA=A in the INPFIL control statement, each EBCDIC x value is translated appropriately to an ISCII/ASCII value in the reformatted output record.

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 Figure 29.

Figure 29. 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	200C' '	400 blanks	400
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	300C'1'	Too many repetitions. Use two adjacent separation fields instead (200C'1',100C'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 output records. n can range from 1 to 256. If n is omitted, 1 is used.

The value yy represents any pair of hexadecimal digits. Specify from 1 to 256 pairs of hexadecimal digits. Examples of valid and invalid hexadecimal string separation are shown in Figure 30.

Figure 30. 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	150X'FFFF'	FF repeated 300 times	300
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	260X'01'	Too many repetitions. Use two adjacent separation fields instead (256X'01',4X'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 output record. the other

p specifies the first byte of the input field relative to the beginning of the 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 4 bytes contain the RDW). Fields can start anywhere within the record on a byte boundary. For special rules concerning variable-length records, see "OUTREC Control Statement Notes" in topic 3.15.1.

Note: If INREC control statement is specified, p must refer to the record as reformatted by INREC control statement. If your E15 user exit routine reformats the record, and INREC control statement is not specified, p must refer to the record as reformatted by the routine.

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 "OUTREC Control Statement Notes" in topic 3.15.1 for more information.

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

The permissible values of **a** are:

H

Halfword aligned. The displacement (p-1) of the field from the beginning of the reformatted output 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 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 input record length (see L4 value in
| "RECORD Control Statement" in topic 3.16) plus 1 byte.

| **p,m,HEX** specifies the hexadecimal representation (EBCDIC) of an
| input field is to appear in the reformatted 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 32767.

| **HEX** requests hexadecimal representation (EBCDIC) 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 the hexadecimal representation (EBCDIC) 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 record includes only the RDW and the variable
| part of the 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 (EBCDIC) 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,m,Y2x specifies that the four-digit character year representation of a two-digit year input field is to appear in the reformatted output record.

p See p under p,m,a.

m specifies the length in bytes of the two-digit year input field.

Y2x requests four-digit character year representation of the input field for one of the two-digit Y2 field formats shown in Figure 31.

Figure 31. Four-Digit Character Year Representation

Format	Length	Description
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
Y2B	1 byte	Two-digit binary year data

The century window established by the Y2PAST option in effect will be used to produce the correct four-digit character year for output. For example, the control statements:

```
SORT   FIELDS=(COPY)
RECORD TYPE=F,LENGTH=80
OUTREC FIELDS=(2X,1,2,C' = ',1,2,Y2C,69X)
```

in conjunction with a century window of 1915-2014, would result in the following output records:

```
23 = 1923
98 = 1998
00 = 2000
02 = 2002
15 = 1915
14 = 2014
```

For Y2S, the four characters produced for output are based on the two- byte input field as follows:

- X'00xx' produces X'000000xx'.
- X'40xx' produces X'404040xx' (X'20xx' produces X'202020xx', if DATA=A is specified in the INPFIL control statement).
- X'xyxy' produces C'yyyy' using the century window.

- | X'FFxx' produces X'FFFFFFxx'.

If you specify DATA=A in the INPFIL control statement, the input field is translated appropriately to an ISCII/ASCII character representation in the reformatted output record.

p,m,PZ specifies that the character representation of a packed decimal input field, with the sign and first digit ignored, is to appear in the reformatted output record.

p See p under p,m,a.

m specifies the length in bytes of the packed decimal input field. Specify a value from 2 to 8.

PZ requests character representation of the packed decimal input field with the sign and first digit ignored. An invalid digit (A-F) other than the first results in a data exception or incorrect numeric output. The output field will be 2m-2 characters long.

PZ can be used to make parts of PD fields printable or readable. For example, if a date appears in position 1 as P'ddmmyy' (X'0ddmmyyC'), PZ and Y2P could be used in an OUTREC control statement as follows to transform the date to C'dd/mm/yyyy':

```
OUTREC FIELDS=(1,2,PZ,C'/',     transform dd
               2,2,PZ,C'/',     transform mm
               3,2,Y2P)         transform yy
```

If you specify DATA=A in the INPFIL control statement, the input field is translated appropriately to an ISCII/ASCII character representation in the reformatted output record.

p,m,PSI specifies that the character representation of a packed decimal input field, with the sign ignored, is to appear in the reformatted output record.

p See p under p,m,a.

m specifies the length in bytes of the packed decimal input field. Specify a value from 1 to 8.

PSI requests character representation of the packed decimal input field with the sign ignored. An invalid digit (A-F) results in a data exception or incorrect numeric output. The output field will be 2m-1 characters long.

PSI can be used to make positive PD fields printable or readable. For example, if a date appears in position 1 as P'yyddd' (X'yydddC'), Y2D and PSI could be used in an OUTREC control statement as follows to transform the date to C'yyyy/ddd':

```
OUTREC FIELDS=(1,2,Y2D,C'/',    transform yy
               2,2,PSI)         transform ddd
```

If you specify DATA=A in the INPFIL control statement, the input field is translated appropriately to an ISCII/ASCII character representation in the reformatted output record.

p,m,ZSI specifies that the character representation of a zoned decimal input field, with the sign ignored, is to appear in the reformatted output record.

p See p under p,m,a.

m specifies the length in bytes of the zoned decimal input field. Specify a value from 1 to 15.

ZSI requests character representation of the zoned decimal input field with the sign ignored. An invalid digit (A-F) results in a data exception or incorrect numeric output. The output field will be m characters long.

ZSI can be used to make positive ZD fields printable or readable. For example, if a date appears in position 1 as Z'yymm', Y2C and ZSI could be used in an OUTREC control statement as follows to transform the date to C'yyyy/mm':

```
OUTREC FIELDS=(1,2,Y2C,C'/',    transform yy
               3,2,ZSI)         transform mm
```

If you specify DATA=A in the INPFIL control statement, the input field is translated appropriately to an ISCII/ASCII character representation in the reformatted output record.

| **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, and ZD fields using either pre-defined edit masks
| (M0-M25) or specific edit patterns you define. You can
| control the way the output fields look 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.

| The edit feature cannot be used if the input data is
| specified as ISCII/ASCII by DATA=A in the INPFIL control
| statement.

| **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 Figure 32 for permissible
| length values.

| **f** specifies the format of the numeric field:

Figure 32. 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
Note: See Appendix B, "Data Format Examples" in topic B.0 for detailed format descriptions.		

| 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.

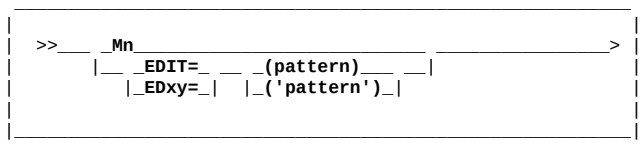
| For a PD0 format field:

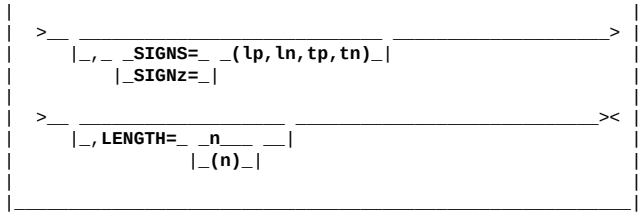
- | The first digit is ignored.

- | 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**





| 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:

Figure 33. Edit Mask Patterns

Mask	Pattern	Examples	
		Value	Result
M0	IIIIIIIIIIIIIITS	+01234 -00001	1234 1-
M1	TTTTTTTTTTTTTTTS	-00123 +00123	00123- 00123
M2	I,III,III,III,IIT.TTS	+123450 -000020	1,234.50 0.20-
M3	I,III,III,III,IIT.TTCR	-001234 +123456	12.34CR 1,234.56
M4	SI,III,III,III,IIT.TT	+0123456 -1234567	+1,234.56 -12,345.67
M5	SI,III,III,III,IIT.TTS	-001234 +123450	(12.34) 1,234.50
M6	III-TTT-TTTT	00123456 12345678	012-3456 1-234-5678
M7	TTT-TT-TTTT	00123456 12345678	000-12-3456 012-34-5678
M8	IT:TT:TT	030553 121736	3:05:53 12:17:36
M9	IT/TT/TT	123094 083194	12/30/94 8/31/94
M10	IIIIIIIIIIIIIIT	01234 00000	1234 0
M11	TTTTTTTTTTTTTTT	00010 01234	00010 01234
M12	SIII,III,III,III,IIT	+1234567 -0012345	1,234,567 -12,345
M13	SIII.III.III.III.IIT	+1234567 -0012345	1.234.567 -12.345
M14	SIII III III III IITS	+1234567 -0012345	1 234 567 (12 345)
M15	III III III III IITS	+1234567 -0012345	1 234 567 12 345-
M16	SIII III III III IIT	+1234567 -0012345	1 234 567 -12 345
M17	SIII'III'III'III'IIT	+1234567 -0012345	1'234'567 -12'345
M18	SI,III,III,III,IIT.TT	+0123456 -1234567	1,234.56 -12,345.67
M19	SI.III.III.III.IIT,TT	+0123456 -1234567	1.234,56 -12.345,67
M20	SI III III III IIT,TTS	+0123456 -1234567	1 234,56 (12 345,67)
M21	I III III III IIT,TTS	+0123456 -1234567	1 234,56 12 345,67-
M22	SI III III III IIT,TT	+0123456 -1234567	1 234,56 -12 345,67

M23	SI'III'III'III'IIT.TT	+0123456 -1234567	1'234.56 -12'345.67
M24	SI'III'III'III'IIT,TT	+0123456 -1234567	1'234,56 -12'345,67
M25	SIIIIIIIIIIIIIT	+01234 -00001	1234 -1

| The elements used in the representation of the
| edit masks in Figure 33 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:

Figure 34. 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
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:

Figure 35. Digits Needed for Numeric Fields		
Format	**Input Length**	**Digits Needed**
ZD	m	m
PD	m	2m-1
PD0	m	2m-2
BI, FI	m	2m+(m+1)/2

| 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
| Figure 35, and the result is rounded down to the
| nearest integer:

Figure 36. 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	PD0,5	11
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	PD0,8	15

| To illustrate conceptually how DFSORT/VSE
| produces the edited output from the numeric
| value, consider the following example:

```
|                              OUTREC FIELDS=(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/VSE 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
  | Figure 34. 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/VSE produces the
| edited output from the numeric value, consider the
| following example:

```
|                    OUTREC FIELDS=(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/VSE 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).

| **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:

```
|                       OUTREC FIELDS=(5,1,ZD,M4)
```

```
|                     the pattern will be C'ST' rather than C'ST.TT'
|                     because the digit length is 1.  Specifying:

|                         OUTREC FIELDS=(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 after editing.
| For example, if you specify:

```
|                         OUTREC FIELDS=(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:

```
|                         OUTREC FIELDS=(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:*

```
|     OUTREC FIELDS=(5:21,8,ZD,M19,25:46,5,ZD,M13)
|     OUTREC FIELDS=(5,2,BI,C' * ',18,2,BI,80:X)
|     OUTREC FIELDS=(11:35,6,PD0,SIGNS=(,,+,-),LENGTH=11,
|               31:8,4,PD,EDIT=(**II,IIT.TTXS),SIGNS=(,,+,-))
```

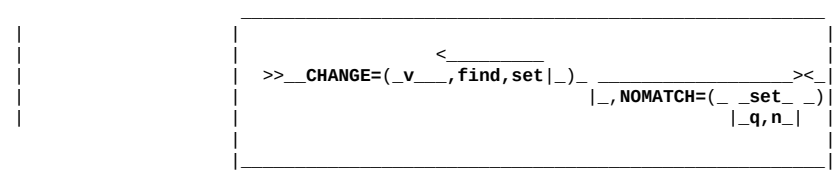
| **p,m,lookup** specifies that a character or hexadecimal string from a lookup
| table is to appear in the reformatted output record. You can
| use p,m,lookup to select a specified character or hexadecimal
| string for the output field based on matching an input value
| against character, hexadecimal, or bit constants.

| The lookup feature cannot be used if the input data is
| specified as ISCII/ASCII by DATA=A in the INPFIL control
| statement.

| **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** 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 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" in
| topic 3.7 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"
  | in topic 3.7 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" in
| topic 3.7 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:

```
|                                OUTREC FIELDS=(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/VSE 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:*

```
|      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'))
```

Default: None; must be specified.

Subtopics:

- 3.15.1 OUTREC Control Statement Notes
- 3.15.2 Reformatting the Output Records--Examples

3.15.1 OUTREC Control Statement Notes

1. If input records are reformatted by the INREC control statement, an E15 (for a sort application) or an E32 (for merge and copy applications) user exit routine, the OUTREC control statement must refer to fields in the appropriate reformatted record (see "INREC Control Statement Notes" in topic 3.9.1).

2. When you specify the OUTREC control statement, you must be aware of the change in record size and layout of the resulting reformatted output records.

3. For variable-length records, the first input field in the FIELDS operand must specify or include the 4-byte RDW. DFSORT/VSE 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 first input field in the FIELDS operand can specify both RDW and data field in one. Otherwise, the RDW must be specifically included in the reformatted output record.

4. 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 that is less than or equal to the minimum record length (see L4 value of the RECORD control statement) plus 1 byte; m and a must be omitted.

   If both the INREC and OUTREC control statements 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 output record includes only the RDW and the variable part of the input record, "null" records containing only an RDW might result.

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

6. Fields referenced in an OUTREC control statement can overlap each other and control fields or both.

7. If input is variable-length records, the output is also variable. This means that each record is given the correct RDW by DFSORT/VSE before output.

8. When the OUTREC control statement is specified, your E35 user exit routine must refer to fields in the reformatted output record.

9. DFSORT/VSE will calculate the L3 value (see L3 of the RECORD control statement).

10. You must consider the calculated value of L3 (see L3 of the RECORD control statement) when specifying the BLKSIZE operand on the OUTFIL control statement.

11. The ADDROUT or ADDROUT=D operand of the OPTION control statement is ignored when an OUTREC statement has been specified.

3.15.2 Reformatting the Output Records--Examples

See "Reformatting Records Before Processing--Examples" in topic 3.9.2, "Example 1," "Example 3," and "Example 4" for applications in which both INREC and OUTREC control statements are used in the same application.

Subtopics:

3.15.2.1 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.

3.15.2.2 Example 2

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

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

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

3.15.2.3 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.

3.15.2.4 Example 4

```
   SORT FIELDS=(20,4,CH,D,10,3,CH,D),WORK=1,FILES=3
   RECORD TYPE=F,LENGTH=80
   OUTREC FIELDS=(5X,20,4,H,8X,20,2,10,3,1Z,1,9,13,7,24,57,6Z)
   INPFIL BLKSIZE=240
   OUTFIL BLKSIZE=206
```

This example illustrates how three fixed-length input files can be sorted and reformatted for output. The input record size is 80 bytes.

The reformatted output records are fixed-length with a record size of 103 bytes and are shown below:

Position Contents

1-5
: EBCDIC blanks for column alignment

6
: Binary zero for halfword alignment

7-10
: Input positions 20 through 23

11-18
: EBCDIC blanks

19-20
: Input positions 20 through 21

21-23
: Input positions 10 through 12

24
: Binary zero

25-33
: Input positions 1 through 9

34-40
: Input positions 13 through 19

41-97
: Input positions 24 through 80

98-103
Binary zeros

3.15.2.5 Example 5

```
   SORT FIELDS=COPY
   RECORD TYPE=V,LENGTH=(200,,,100,130)
   OUTREC FIELDS=(1,7,5Z,5X,28,8,6X,101)
   INPFIL BLKSIZE=2004
   OUTFIL BLKSIZE=1314
```

This example illustrates how a variable-length input file can be copied 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 is 200 bytes.

The reformatted output records are variable-length with a maximum record size of 131 bytes. The reformatted records are shown below:

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)

3.15.2.6 Example 6

```
   MERGE FIELDS=(28,4,BI,A),FILES=2
   RECORD TYPE=V,LENGTH=(200,,,100,185)
   OUTREC FIELDS=(1,4,5Z,5X,5,3,28,8,6Z)
```

This example illustrates how two unblocked variable-length 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 minimum input record size is 100 bytes, and the maximum input record size is 200 bytes.

The reformatted output records are variable length with a maximum record size of 31 bytes and are shown below.

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

| 3.15.2.7 Example 7

```
| |   OPTION Y2PAST=26                                                  |
| |   SORT   FIELDS=COPY                                                |
| |   RECORD TYPE=F,LENGTH=80                                           |
| |   OUTREC FIELDS=(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 file with a packed
| decimal date field of the form P'yymmdd' (X'0yymmddC') in bytes 20-23 into
| a new file 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 file has a record length of 80 and the output file will have a
| record length 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 1998, the
| century window will be 1972 to 2071. 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
```

| 3.15.2.8 Example 8

```
OPTION Y2PAST=1998
SORT   FIELDS=COPY
RECORD TYPE=F,LENGTH=40
OUTREC FIELDS=(1,14,   copy positions 1-14
            19,2,Y2S,  transform yy to yyyy - allow blanks
            15,4,      copy mmdd
            21,20)     copy positions 21-40
```

| This example illustrates how to transform an existing file with a
| character date field of the form C'mmddyy' and blank special indicators in
| bytes 15-20, into a new file with a character date field of the form
| C'yyyymmdd' and blank special indicators in bytes 15-22.

| The input file has a record length of 40 and the output file will have a
| record length of 42.

| The Y2PAST=1998 option sets the century window to 1998-2097. 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      051298  CA
BOCA RATON    062800  FL
DENVER        111597  CO
```

| The output records would be as follows:

```
MORGAN HILL             CA
SAN JOSE      19980512  CA
BOCA RATON    20000628  FL
DENVER        20971115  CO
```

| 3.15.2.9 Example 9

```
SORT FIELDS=(15,6,ZD,A)
RECORD TYPE=F,LENGTH=80
OUTREC FIELDS=(5:15,6,ZD,M11,
            20:3,4,ZD,M12,LENGTH=9,
            35:38,8,ZD,M18,LENGTH=12)
```

| The OUTREC statement's editing capabilities can be used to produce numeric
| formats commonly used in different countries. This example illustrates
| how to produce output records that have numeric formats commonly used in
| the United States. The next example illustrates how to produce output
| records that have numeric formats commonly used in France.

| The OUTREC control statement 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 the United States.
  | M12 uses a comma 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 the United States. M18 uses a comma for the
  | thousands separator and a period for the decimal separator.

| Figure 33 in topic 3.15 shows the 26 pre-defined edit masks (M0-M25) that
| you can choose from.

| The output records might look as follows:

```
|           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
```

| 3.15.2.10 Example 10

```
  ____________________________________________________________________
  |                                                                  |
| |   SORT FIELDS=(15,6,ZD,A)                                        |
| |   RECORD TYPE=F,LENGTH=80                                        |
| |   OUTREC FIELDS=(5:15,6,ZD,M11,                                  |
| |              20:3,4,ZD,M16,LENGTH=9,                             |
| |              35:38,8,ZD,M22,LENGTH=12)                           |
  |                                                                  |
  |__________________________________________________________________|
```

| This example illustrates how to produce output records that have numeric
| formats commonly used in France.

| The OUTREC control statement 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 France. M16 uses a
  | blank 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 France. M22 uses a blank for the thousands separator
  | and a comma for the decimal separator.

| Figure 33 in topic 3.15 shows the 26 pre-defined edit masks (M0-M25) that
| you can choose from.

| The output records might look as follows:

```
|        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
```

| 3.15.2.11 Example 11

```
  _____________________________________________________________________
 |                                                                     |
| |   SORT FIELDS=(6,5,CH,A)                                          |
| |   RECORD TYPE=V,LENGTH=80                                         |
| |   OUTREC FIELDS=(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)                                             |
 |                                                                     |
 |_____________________________________________________________________|
```

| This example illustrates how output records can be produced from sorted
| variable-length input records, using a lookup table.

| The OUTREC control statement 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.

| The first few output records might look as follows:

```
|            00082     HOLD              36
|            00123     SHIP             106
|            00300     *CHECK CODE*      95
|            10321     TRANSFER          18
|            12140     SHIP             120
```

| 3.15.2.12 Example 12

```
  ____________________________________________________________________________
 |                                                                            |
| |   SORT FIELDS=(20,4,CH,D,10,3,CH,D)                                      |
| |   RECORD TYPE=F,LENGTH=80                                                |
| |   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 file can be sorted and
| reformatted for output. The SORTIN record length is 80 bytes.

| The reformatted output records are fixed-length with a record size of 103
| bytes and are shown below.

| **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

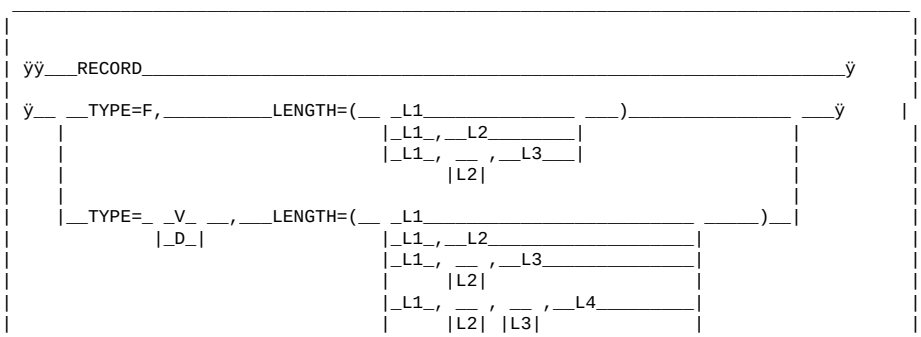
| **25-33**
Input positions 1 through 9

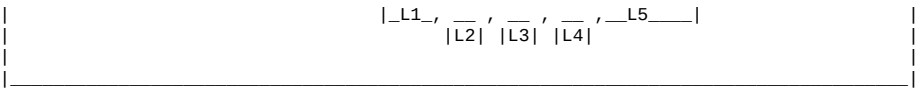
| **34-40**
Input positions 13 through 19

| **41-97**
Input positions 24 through 80

| **98-103**
Hexadecimal string: X'FFFFFFFFFFFF'

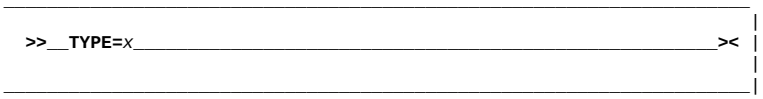
3.16 RECORD Control Statement





The RECORD control statement describes the format and lengths of the records being processed. It must always be provided. Both the LENGTH and TYPE operands must be specified.

TYPE



Specifies the format of the records being processed. x can take the following values:

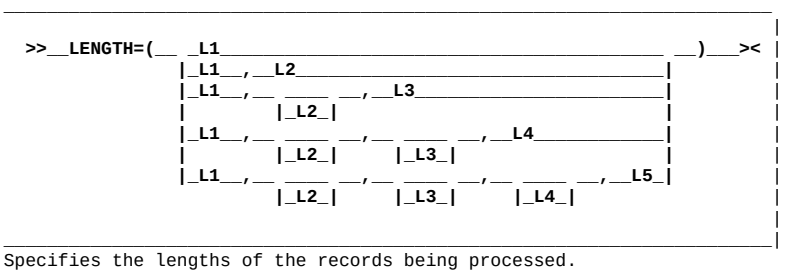
F
specifies that the records are fixed-length records.

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

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

Default: None; must be specified.

LENGTH



Specifies the lengths of the records being processed.

Note: L4 through L5 apply only to variable-length records.

L1
Input record length. For variable-length records, maximum input record length.

Default: None; must be specified.

L2
Record length after E15 user exit routine. For variable-length records, maximum record length after E15 user exit routine.

Notes:

1. L2 is ignored if E15 user exit routine is not used.

2. An accurate value for L2 must be specified if E15 user exit routine changes the record length.

Default: L1.

L3

Output record length after E35 user exit routine. For variable-length records, maximum record length after E35 user exit routine.

Notes:

1. L3 is ignored if E35 user exit routine is not used.

2. An accurate value for L3 must be specified if E35 user exit routine changes the record length.

Default: L2 for a sort, or L1 for a merge and copy.

L4

Minimum input record length.

Notes:

1. L4 is only used for variable-length record sorts.

2. If E15 user exit routine changes the record length, specifies the minimum record length after E15 user exit routine.

3. Specifying L4 may improve performance.

Default: The minimum length needed to contain all control fields. This number must be at least 14 bytes, otherwise, DFSORT/VSE sets L4 to 14 bytes.

L5

Average (modal) input record length.

Notes:

1. L5 is only used for variable-length record sorts.

2. If E15 user exit routine changes the record length, specifies the average record length after E15 user exit routine.

3. Specifying L5 may improve performance.

Default: The average of L2 and L4: L5=(L2+L4)/2.

Subtopics:

3.16.1 RECORD Control Statement Notes

1. L1 must be specified.

2. You can drop values from the right. For example,

   ```
   LENGTH=(80,70,70,70).
   ```

3. You can omit values from the middle or left, provided you indicate their omission by a comma. For example,

   ```
   LENGTH=(80,,30,70).
   ```

4. Parentheses are optional when L1 alone is specified. If any of L2 through L5 is specified, parentheses are required.

5. If an ADDROUT or ADDROUT=D operand is specified in the OPTION control statement, it is unnecessary to specify L2 or L3. DFSORT/VSE will calculate these values as follows:

   - L2=10+CF for SAM files, or 5+CF for VSAM files.
   - L3=10 for SAM files, or 5 for VSAM files (if ADDROUT is specified).
   - L3=10+CF for SAM files, or 5+CF for VSAM files (if ADDROUT=D is specified).

   Where CF is the sum of the control field lengths. DFSORT/VSE will terminate if L2 or L3 is specified incorrectly. L4 and L5 will be calculated by DFSORT/VSE.

6. If an INREC or OUTREC control statement is specified, it is unnecessary to specify L3. DFSORT/VSE will calculate L3 as the length of the reformatted output record. DFSORT/VSE will terminate if L3 is specified incorrectly.

7. The lengths specified for variable-length records must include the 4-byte RDW. This applies even for VSAM variable-length records which normally have no RDW, since DFSORT/VSE has to build an RDW for these records when reading them in.

8. Record size must not exceed the track size of any CKD direct access devices used for work files.

3.16.2 Describing the Record Format and Length--Examples

Subtopics:

3.16.2.1 Example 1

```
RECORD   TYPE=F,LENGTH=80
```

TYPE The input records are fixed-length.

LENGTH The records in the input file are each 80 bytes long.

3.16.2.2 Example 2

```
RECORD   TYPE=F,LENGTH=(80,,50)
```

TYPE The input records are fixed-length.

LENGTH The records in the input file are each 80 bytes long. An E35 user exit routine changes the record lengths to 50 bytes. Note that L2 value is omitted.

3.16.2.3 Example 3

```
RECORD   TYPE=V,LENGTH=(104,,,44,84)
```

TYPE The records in the input file are EBCDIC variable-length.

LENGTH The maximum length of the records in the input file is 104 bytes. The minimum record length is 44 bytes and the average record length is 84 bytes.

3.16.2.4 Example 4

```
RECORD   TYPE=F,LENGTH=(215,22,5)
```

This example assumes that this statement is used in conjunction with the ADDROUT operand in the OPTION control statement and that the input file is a VSAM file.

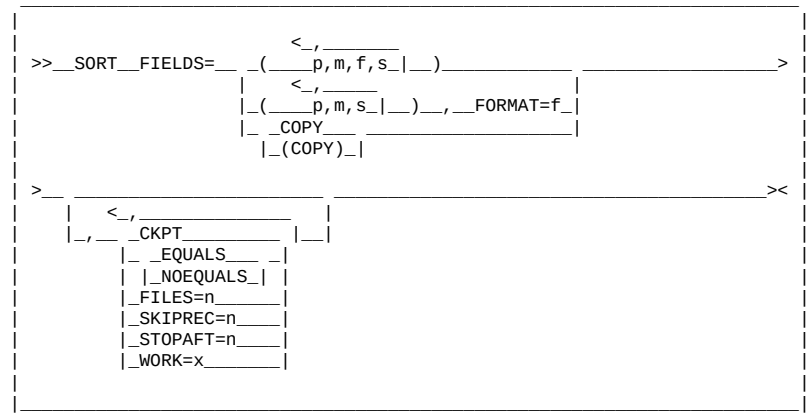
TYPE The input records are fixed-length.

LENGTH The records in the input file are each 215 (the L1 value) bytes long. The L2 value is 22 bytes, and consists of 5 bytes for the disk address plus a 17-byte control word for each record. Each output record contains a 5-byte address of input record rather than data. The L3 value must be 5 for a VSAM file with the ADDROUT option specified. However, you do not need to give this value for L2 and L3, as they can be supplied by default; for example, the statement

```
RECORD   TYPE=F,LENGTH=215
```

is also correct for the above example.

3.17 SORT Control Statement



The SORT control statement must be used when a sort application is performed; this statement describes the control fields in the input records on which DFSORT/VSE sorts.

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

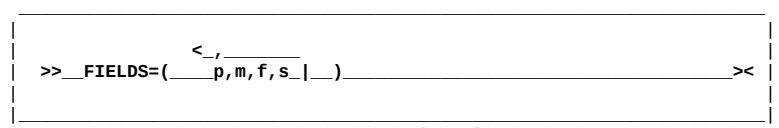
DFSORT/VSE's collating and ordering behavior can be modified according to your cultural environment. The cultural environment is established by selecting the active locale. The active locale's collating rules affect SORT processing. DFSORT/VSE will produce sorted records for output according to the collating rules defined in the active locale. This provides sorting for single- or multi-byte character data based on defined collating rules 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) control fields.

For more information on locale processing, see "Cultural Environment Considerations" in topic 1.3.3 or LOCALE in "OPTION Control Statement" in topic 3.13.

Note: Up to 9 files can be sorted or copied.

FIELDS



All control fields (up to 64) must be located within the first 4092 bytes of a record.

Control fields must not extend beyond the shortest record to be sorted. The collected control fields (comprising the control word) must not exceed 3072 bytes.

The FIELDS operand can be written in two ways.

Use the first FIELDS operand format to describe control fields that contain different data formats; use the second format (explained in the discussion of the FORMAT operand later in this section) to describe control fields that contain data of the same format. The second format is optional; if you prefer, you can always use the first format.

DFSORT/VSE 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. 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: 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.

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/VSE supplies the necessary RDW on input to the program and removes it again at output (if output is to a VSAM file). 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 37:

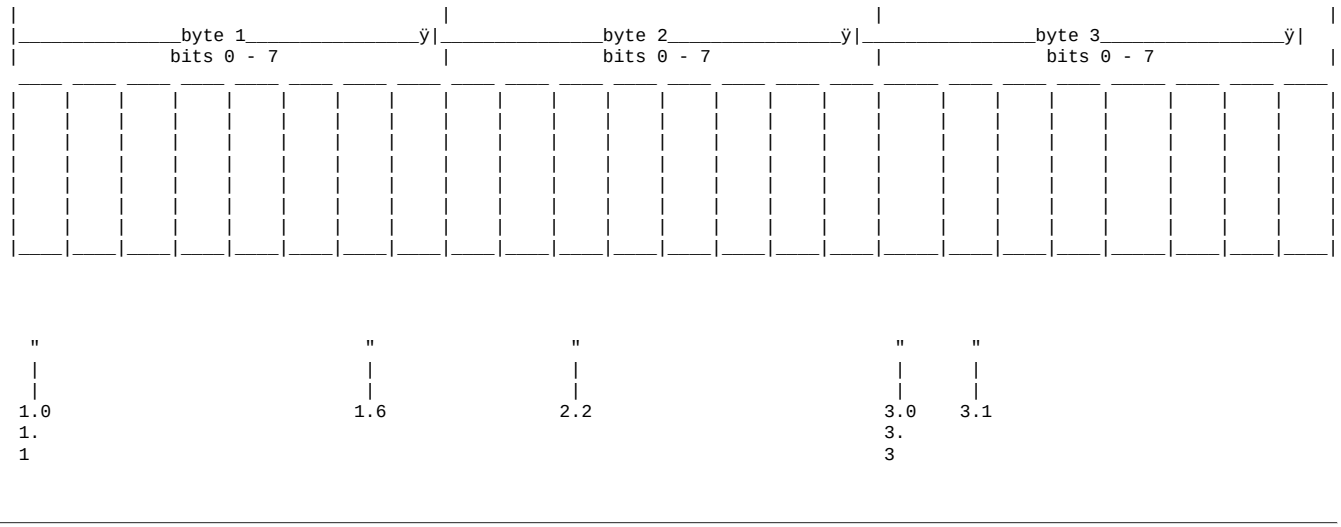


Figure 37. Examples of Notation for Binary Fields

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 3072. When you determine the total, count a binary field as occupying an entire byte if it occupies any part of it. For example, a binary field that begins on byte 2.6 and is 3 bits long occupies two bytes. All fields must be completely contained within the first 4092 bytes of the record.

f

specifies the format of the data in the control field. Acceptable control field lengths (in bytes) and available formats are shown in Figure 38.

Figure 38. Control Field Formats and Lengths

Format	Length	Description
CH	1 to 256 bytes	Character
AQ	1 to 256 bytes	Character with alternate collating sequence
ZD	1 to 256 bytes	Signed zoned decimal
PD	1 to 32 bytes	Signed packed decimal
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 256 bytes	Unsigned binary
FL	1 to 256 bytes	Signed floating-point
AC	1 to 256 bytes	ISCII/ASCII character
CSL	2 to 256 bytes	Signed numeric with leading separate sign
CST	2 to 256 bytes	Signed numeric with trailing separate sign
CLO	1 to 256 bytes	Signed numeric with leading overpunch sign
CTO	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
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
Y2B	1 byte	Two-digit binary year data

Note: See Appendix B, "Data Format Examples" in topic B.0 for detailed format descriptions.

The century window established by the Y2PAST option in effect will be used to produce the correct order for Y2 format fields. For example, the control statement:

```
SORT FIELDS=(1,2,Y2C,A)
```

in conjunction with a century window of 1915-2014, would result in the following correct two-character year data ordering:

```
15, 28, 95, 99, 00, 05, 14
```

| The century window established by the Y2PAST option in effect will
| be used to interpret Y2S format two-digit year values, but will
| not be used to interpret Y2S format special indicators. For
| example, the control statement:

```
|              SORT FIELDS=(1,2,Y2S,A)


|         in conjunction with a century window of 1915-2014, would result in

|         the following correct two-digit character year and special

|         indicator data ordering:


|             X'0000', X'4040', X'F1F5', X'F9C7', X'F1F4', X'FFFF'
```

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 DFSORT/VSE can collate it properly. You can use an E15 user exit routine to do this during processing.

For information on how to add user exits, see Chapter 4, "Using Your Own User Exit Routines" in topic 4.0.

s

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

A
ascending order

D
descending order

Default: None; must be specified.

FORMAT

```
 _____________________________________________________________
|                                                             |
|  >>__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 operand.
```

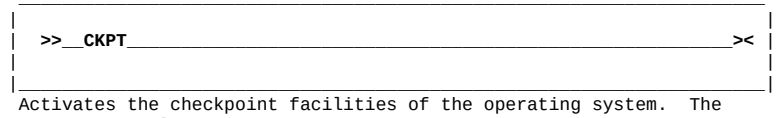
Default: None; must be specified if not included in FIELDS operand.

FIELDS=COPY or FIELDS=(COPY)

```
 _____________________________________________________________
|                                                             |
|  >>__FIELDS=__ _COPY___ ___________________________________><|
|              |_(COPY)_|                                     |
|                                                             |
|_____________________________________________________________|
 Can be used to copy one or more input files to an output file.  All
 other SORT operands (except SKIPREC and STOPAFT) are ignored when
 FIELDS=COPY is specified.  If FIELDS=COPY is specified with a SUM or
 INREC control statement, DFSORT/VSE will terminate.
```

Default: None; optional.

CKPT



Activates the checkpoint facilities of the operating system. The checkpoint file must:

- Be assigned to SYS000 (a tape or a direct access device, CKD or FBA)
- Have standard labels
- Have the file name SORTCKP.

Only one checkpoint is taken by DFSORT/VSE.

If DFSORT/VSE was program invoked or user exits are in use, the entire partition is checkpointed. Otherwise, only the area being used by DFSORT/VSE is checkpointed. Checkpointing a large partition will adversely affect DFSORT/VSE elapsed time.

You must use VSE/ESA restart facilities to continue an interrupted application. The general procedure is as follows:

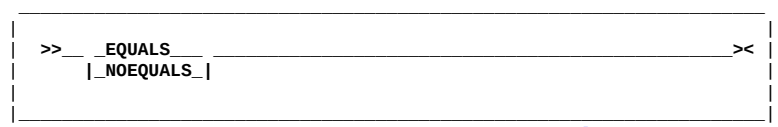
1. Submit a RSTRT job control statement to request the system restart facilities.

2. Submit all ASSGN job control statements that were in effect at the time checkpoint was taken.

Checkpoint is ignored if work files are not used, DFSORT/VSE is subtasked, or E31 user exit is specified in the MODS control statement (in this case, DFSORT/VSE will pass a device list to your routine at E31 user exit so that you can take a checkpoint there). See *VSE/ESA System Macros Reference* and "Checkpoint/Restart" in topic D.2 to determine if checkpoint can be used.

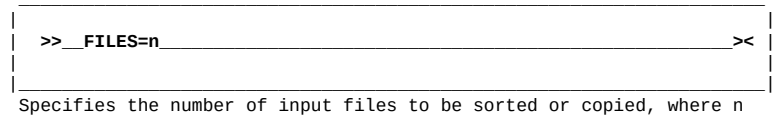
Default: None; optional.

EQUALS or NOEQUALS



See the discussion of this operand in "OPTION Control Statement" in topic 3.13.

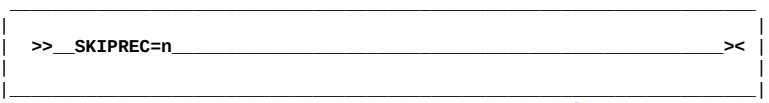
FILES



Specifies the number of input files to be sorted or copied, where n can be any number from 1 through 9.

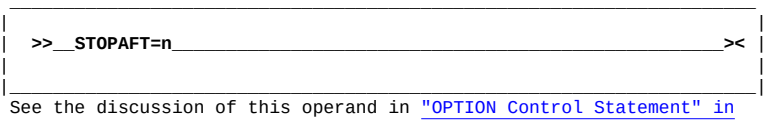
Default: 1.

SKIPREC



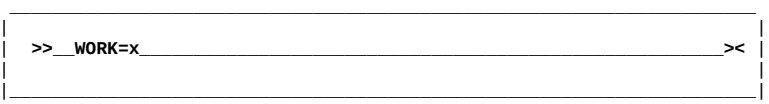
See the discussion of this operand in "OPTION Control Statement" in topic 3.13.

STOPAFT



See the discussion of this operand in "OPTION Control Statement" in topic 3.13.

WORK



Describes the work files. Permissible values of x are:

DA
specifies that the work file DLBL job control statement defines the file as multiextent (DA). (Not allowed for FBA work files.)

n
specifies the number of work file DLBL job control statements supplied, where n can be any number from 0 through 9. If 0 is specified, DFSORT/VSE will not use any work files and will attempt to perform an incore sort.

Default: DA.

Subtopics:

- 3.17.1 SORT Control Statement Notes
- 3.17.2 Sorting and Copying--Examples

3.17.1 SORT Control Statement Notes

1. If the records are reformatted by an INREC control statement or an E15 user exit routine, the SORT control statement must refer to fields in the appropriate reformatted record (see the description for p following FIELDS above).

2. WORK=DA means the same as WORK=1. DFSORT/VSE will determine if the files are DA, SD, or SAM ESDS.

3. In control fields, +0, 0, and -0 are treated as the same number and compare equal.

4. Input files must be all SAM or all VSAM.

5. If any of the input files are multivolume and unlabeled or nonstandard labeled, you must specify the number of volumes using the VOLUME operand of the INPFIL control statement.

6. If WORK=DA is specified or defaulted, the first extent must contain at least two tracks.

7. EQUALS is ignored when the SUM control statement is specified.

3.17.2 Sorting and Copying--Examples

Subtopics:

3.17.2.1 Example 1

```
   SORT   FIELDS=(2,5,CH,A),WORK=1
```

FIELDS The control field begins on the second byte of each record in the input file, is 5 bytes long, and contains character (EBCDIC) data. It is to be sorted in ascending order.

WORK One work file is used.

3.17.2.2 Example 2

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

FIELDS The major control field begins on byte 25 of each record, is 4 bytes long, contains zoned decimal data, and is to be sorted in ascending order.

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 operand can be used because both control fields have the same data (zoned decimal) format.

It would also be correct to write this SORT control statement as follows:

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

EQUALS specifies that the sequence of equal collating records is to be preserved from input to output.

WORK A multiextent work file is used.

3.17.2.3 Example 3

```
SORT   FIELDS=COPY,FILES=3
```

FIELDS Three input files are copied to the output file without sorting.

3.17.2.4 Example 4

```
SORT   FIELDS=(12,1,Y2D,A,13,2,PD,A)
```

FIELDS Sorts a packed decimal date field of the form P'yyddd' (X'yydddC'). yy represents the two-digit year, ddd represents the day of the year and C represents a positive sign.

The first control field begins on byte 12 of each record, is 1 byte long and is to be sorted in ascending sequence using the century window in effect to interpret the two-digit years as four-digit years. The second control field begins on byte 13 of each record, is 2 bytes long, contains packed decimal data, and is to be sorted in ascending sequence.

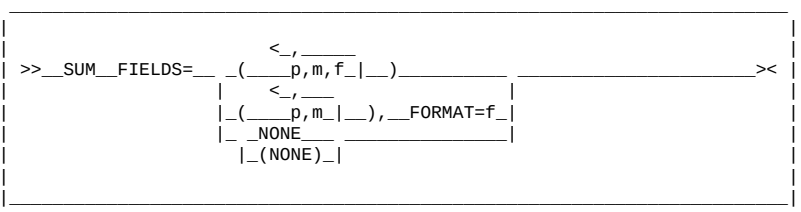
3.17.2.5 Example 5

```
 SORT   FIELDS=(7,3,PD,D,398.4,7.6,BI,D)
```

FIELDS The major control field begins on byte 7 of each record, is 3 bytes long, contains packed decimal data, and is to be sorted in descending order.

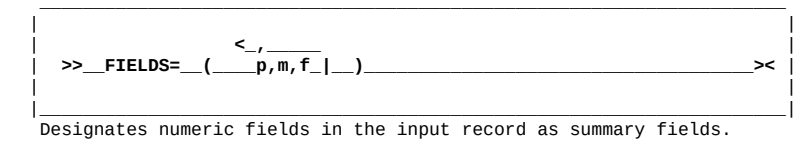
The second control field begins on the fifth bit (bits are numbered 0 through 7) of byte 398, is 7 bytes and 6 bits long (occupies 9 bytes), and contains binary data to be sorted in descending order.

3.18 SUM Control Statement



The SUM control statement specifies that, whenever two records are found with equal control fields (duplicate keys), 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



Requires three facts about each summary field in the input records: the position of the field within the record, the length of the field, and the format of the data in the field. These facts are communicated to DFSORT/VSE by the values of the FIELDS operand, represented by p, m, and f.

The FIELDS operand can be written in two ways.

Use the first FIELDS operand format to describe summary fields that contain different data formats; use the second format (explained in the discussion of the FORMAT operand later in this section) to describe summary fields that contain data of the same format. The second format is optional; if you prefer, you can always use the first format.

p specifies the first byte of the summary field relative to the beginning of the 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 contain the RDW). All fields must start on a byte boundary, and no field can extend beyond byte 4092.

Note: If INREC control statement is specified, p must refer to the record as reformatted by INREC control statement. If your E15 user exit routine reformats the record, and INREC control statement is not specified, p must refer to the record as reformatted by the routine.

m specifies the length in bytes of the summary field. See Figure 39 for permissible length values.

f specifies the format of the data in the summary field. Acceptable summary field lengths (in bytes) and available formats are shown in Figure 39.

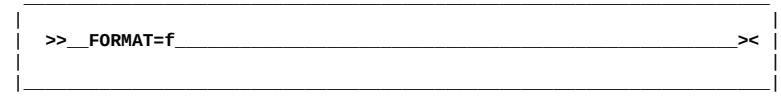
Figure 39. Summary Field Formats and Lengths

Format	Length	Description
BI	2, 4, or 8 bytes	Unsigned binary
FI	2, 4, or 8 bytes	Signed fixed-point
PD	1 to 16 bytes	Signed packed decimal
ZD	1 to 18 bytes	Signed zoned decimal

Note: See Appendix B, "Data Format Examples" in topic B.0 for detailed format descriptions.

Default: None; must be specified.

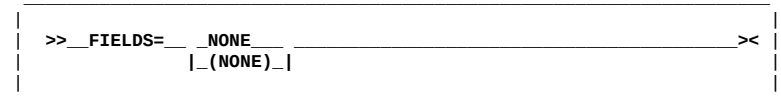
FORMAT



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 operand.

Default: None. Must be specified if not included in the FIELDS operand.

FIELDS=NONE or FIELDS=(NONE)



```
|_____________________________________________________________|
 Can be used to eliminate records with duplicate keys.  Only one record
 with each key is kept and no summing is performed.
```

Note: ICETOOL's SELECT operator can perform the same function as SUM FIELDS=NONE, as well as other functions related to records with duplicate keys.

Default: None; must be specified.

Subtopics:

3.18.1 SUM Control Statement Notes

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

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

3. Floating-point fields must not be summed.

4. When records are summed, the choice of which record is to receive the sum (and be retained) and which record is to be deleted is unpredictable.

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

6. If overflow occurs, the two records involved are left unsummed. That is, the contents of the records are left undisturbed, neither record is deleted, and the records are still available for summing. Overflow does not prevent further summing.

7. If both the SUM control statement and the EQUALS operand of the SORT control statement are specified, the EQUALS operand will be ignored.

8. If both the SUM control statement and the ADDROUT or ADDROUT=D operand of the OPTION control statement are specified, the ADDROUT or ADDROUT=D operand will be ignored.

9. Summing of data with invalid sign or digit codes results in a data exception program check. 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.

10. | Whether or not positive summed ZD results have printable numbers
    | depends on whether NZDPRINT or ZDPRINT is in effect (as set by the
    | ZDPRINT parameter of the ILUINST macro and the NZDPRINT or ZDPRINT
    | operand of the OPTION control 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 keep their original signs, regardless of
| whether NZDPRINT or ZDPRINT is in effect. For example, if X'F2F0C5'
| is not summed, it remains X'F2F0C5' (prints as '20E'). The OUTREC
| control statement can be used to ensure that all summed or unsummed ZD
| values are printable, as illustrated by Example 5 in "Specifying
| Summary Fields--Examples" in topic 3.18.2.

3.18.2 Specifying Summary Fields--Examples

Subtopics:

3.18.2.1 Example 1

```
SUM FIELDS=(21,8,PD,11,4,FI)
```

FIELDS Designates an 8-byte packed decimal field at byte 21, and a 4-byte fixed-integer field at byte 11, as summary fields.

3.18.2.2 Example 2

```
SUM FIELDS=(41,8,49,4),FORMAT=ZD
```

FIELDS Designates two zoned decimal fields, one 8 bytes long starting at byte 41, and the other 4 bytes long starting at byte 49.

FORMAT Can be used because both summary fields have the same data (zoned decimal) format.

It would also be correct to write this SUM control statement as follows:

```
SUM FIELDS=(41,8,ZD,49,4,ZD)
```

3.18.2.3 Example 3

```
SUM FIELDS=NONE
```

FIELDS Specifies that the elimination of records with duplicate keys without summing is to be provided.

| 3.18.2.4 Example 4

```
SUM FIELDS=(41,8,49,4),FORMAT=ZD
OPTION ZDPRINT
```

| **FIELDS** Designates two zoned decimal fields, one 8 bytes long starting at
| byte 41, and the other 4 bytes long starting at byte 49.

| **FORMAT** Can be used because both summary fields have the same data (zoned
| decimal) format.

| **ZDPRINT** 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.

| 3.18.2.5 Example 5

```
  _____________________________________________________________________
  |                                                                    |
| |   SUM FIELDS=(41,8,49,4),FORMAT=ZD                                 |
| |   OUTREC FIELDS=(1,40,41,8,ZD,M11,49,4,ZD,M11,53,28)               |
  |                                                                    |
  |____________________________________________________________________|
```

| These statements illustrate the use of the OUTREC control statement to
| ensure that all positive ZD summary fields in the output file are
| printable. Whereas the ZDPRINT option affects only positive summed ZD
| fields, OUTREC can be used to edit positive or negative BI, FI, PD or ZD
| values, whether they are summed or not.

| **Note:** For purposes of illustration, this example assumes that the input
| records are 80 bytes long.

4.0 Chapter 4. Using Your Own User Exit Routines

Subtopics:

4.1 Overview

DFSORT/VSE 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 summing records.

If you need to perform these tasks, you should be aware that DFSORT/VSE already provides extensive facilities for working with your data in the various DFSORT/VSE program control statements. See the discussions of the INCLUDE, OMIT, INREC, OUTREC, and SUM program control statements in Chapter 3, "Using DFSORT/VSE Program Control Statements" in topic 3.0. You might decide that using a program control statement to work with your records is more appropriate to your needs.

Although this chapter discusses only routines written in Assembler, 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 a parameter list.

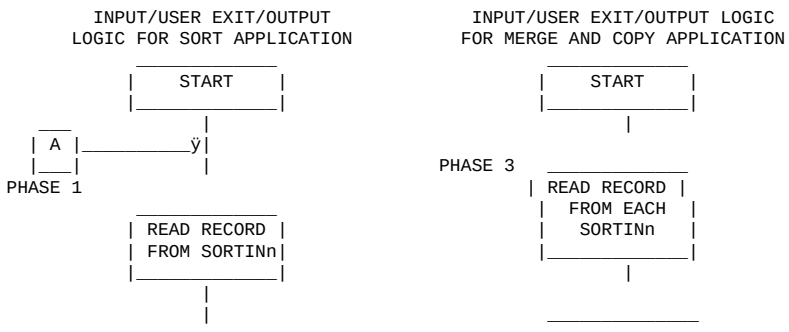
You can easily activate user exit routines at run time with the MODS program control statement (see "MODS Control Statement" in topic 3.11). Alternatively, under certain circumstances you can also activate user exit routines by passing the address of your routines in the invocation parameter list. See Chapter 5, "Invoking DFSORT/VSE from a Program" in topic 5.0 for details.

Parameters that affect the way user exit routines are handled include:

- The MODS program control statement, explained in "MODS Control Statement" in topic 3.11
- The INPFIL program control statement, explained in "INPFIL Control Statement" in topic 3.8
- The OUTFIL program control statement, explained in "OUTFIL Control Statement" in topic 3.14
- The parameter list, explained in Chapter 5, "Invoking DFSORT/VSE from a Program" in topic 5.0.

4.2 DFSORT/VSE Processing Phases

DFSORT/VSE performs specific tasks such as reading the input files (initial sorting phase--phase 1) and writing the output file (final merging phase--phase 3). Various user exits are contained in the initial sorting and final merging phases and are activated at a particular time during DFSORT/VSE processing. The initial sorting phase is used only for a sort application. When the final merging phase is completed, DFSORT/VSE returns control to the operating system or invoking program. Figure 40 is a representation of DFSORT/VSE input, user exit, and output logic.



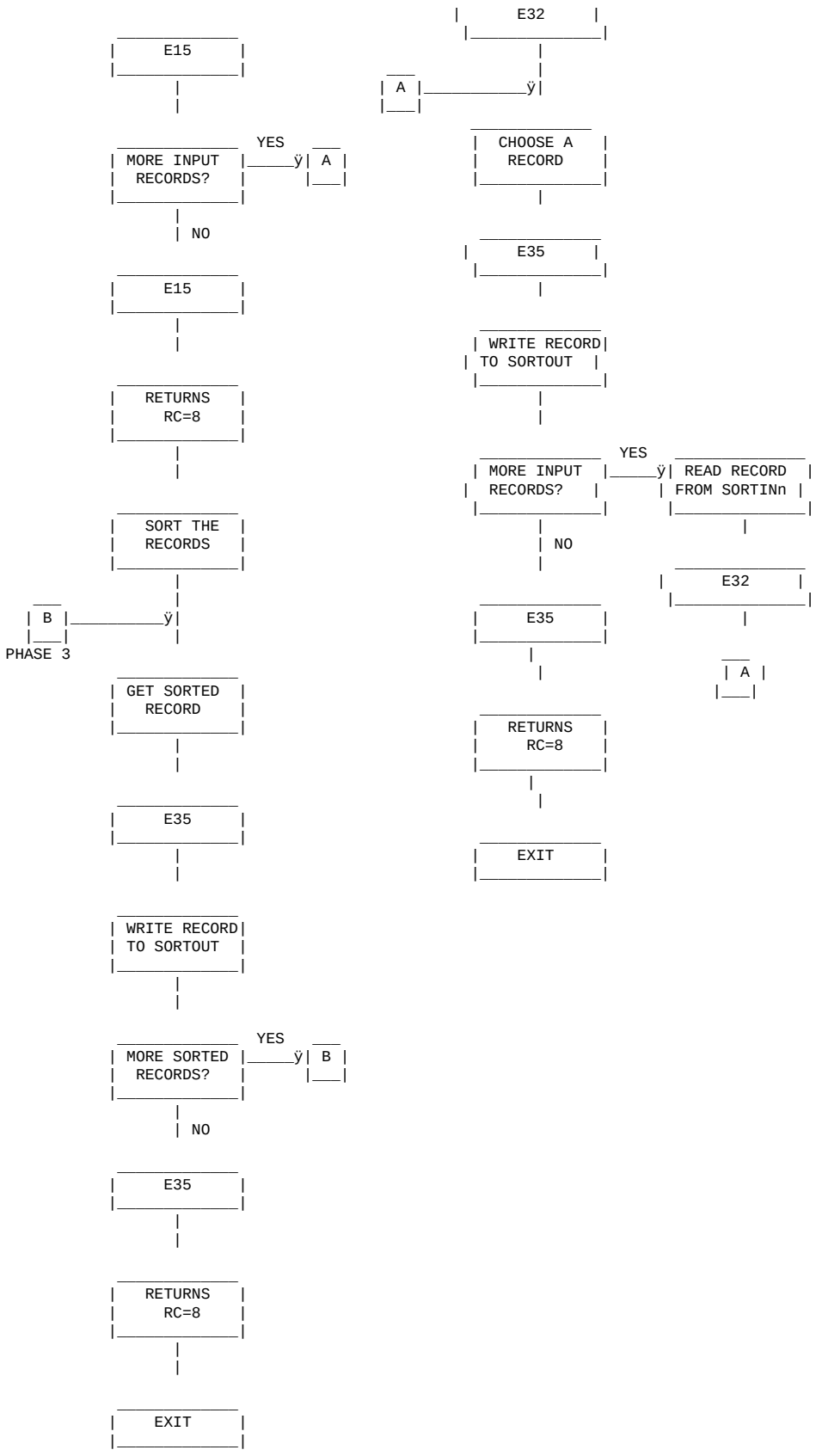


Figure 40. Examples of DFSORT/VSE Input/User Exit/Output Logic

4.3 Functions of Routines at User Exits

You can use user exit routines to accomplish a variety of tasks:

- Open and initialize files
- Process labels
- Control all input and output
- Take checkpoints
- Alter, delete, or insert records
- Sum records
- Modify VSAM processing
- Close files
- Terminate DFSORT/VSE.

Figure 40 in topic 4.2 and Figure 42 in topic 4.5 summarize the functions of user exit routines and the user exits and phases with which they can be associated.

Subtopics:

4.3.1 DFSORT/VSE Input, User Exit, Output Logic Examples

Figure 40 in topic 4.2 gives examples of the logic flow for sort, merge, and copy applications as it relates to SORTINn, E15, E32, or E35 user exits, and SORTOUT. The intent is to show how your E15, E32, and E35 user exit routines fit into the logic of an application. All possible paths are not covered. For simplicity, it is assumed that all of the applicable files and user exits are present and that records are not inserted or deleted. (For merge and copy applications, similar logic would be used if an E32 user exit routine supplied the records rather than SORTINn files.)

Figure 40 in topic 4.2 illustrates the following logic:

- E15 and E35 user exit routines continue to be entered until they pass back a return code of 8. If your user exit routine passes a return code of 8 to DFSORT/VSE, subsequent records are processed by DFSORT/VSE *without* being passed to your user exit routine.
- During sort processing, each record is read from SORTINn and passed to E15 user exit routine. When *all* of the records have been accepted in this manner, they are sorted by DFSORT/VSE, then each sorted record is passed to your E35 user exit routine or written to SORTOUT.
- During merge or copy processing, one record is initially read from each SORTINn file and passed to your E32 user exit routine. The record to be output is chosen, passed to your E35 user exit routine or written to SORTOUT. The chosen record is then replaced by reading a record from the same SORTINn file or your E32 user exit routine and the process continues.

Notes:

1. DFSORT/VSE first checks whether you supplied an EXIT operand in the INPFIL control statement for sort, merge, or copy application. If not, DFSORT/VSE reads the input records from SORTINn files.

- If an EXIT operand is present for a sort application, DFSORT/VSE does not read the input records from SORTINn files; you must use an E15 user exit routine to insert all the records.

- If an EXIT operand is present for a merge or copy application, DFSORT/VSE does not read the input records from SORTINn files; you must use an E32 user exit routine to insert all the records.

2. DFSORT/VSE writes each processed record to the SORTOUT file, if present, or, if an EXIT operand in the OUTFIL control statement is present, the E35 user exit routine must dispose of all the records because DFSORT/VSE treats these records as deleted.

You can use these user exits in the DFSORT/VSE initial sorting phase:

- E11
- E15
- E17
- E18

You can use these user exits in the DFSORT/VSE final merging phase (incore sort):

- E31
- E35
- E37
- E39

You can use these user exits in the DFSORT/VSE final merging phase:

- E31
- E32
- E35
- E37
- E38
- E39

These user exits are discussed in sequence. To determine whether a particular user exit can be used for your application, refer to Figure 42 in topic 4.5.

4.3.2 Opening and Initializing Files

You can write your own user exit routines to open files and perform any necessary initialization. For these purposes you should use E11 and E31 user exits.

4.3.3 Processing Labels

If your tapes or disks have standard labels, or if you use unlabeled tapes, you need do nothing about either label processing or opening and closing files. The work will be done by DFSORT/VSE using the standard facilities of the operating system.

DFSORT/VSE cannot, however, handle nonstandard labels, or user-standard labels (standard labels with extra header or trailer labels). You must process these labels and open and close the files at the appropriate user exits. Figure 41 summarizes how user exits can be used for file label handling.

You should be familiar with the label processing procedures described in *VSE/ESA System Macros Reference*, *VSE/ESA System Macros User's Guide*, *VSE/ESA System Control Statements*, and *VSE/ESA Guide to System Functions* as appropriate.

Figure 41 is equally applicable to tape and disk files with the exception of remarks on the subject of trailer labels: only tape files can have trailer labels.

SAM ESDS files must have standard labels.

Figure 41. Which User Exits to Use for File Label Handling

User Exits for File Label Handling	Input (sort)	Input (merge/copy)	Output
Open file, process header labels	E11	E31	E31
Process all trailer labels except the last	E11	E31	E31
Process last trailer label, close file	E17	E37	E37

4.3.4 Control All Input and Output

The EXIT operand in the INPFIL control statement (INPFIL EXIT specification) is a promise to DFSORT/VSE that you will take care of all input to the sort, merge, or copy application--not just the individual records, but the files as well. In the same way, OUTFIL EXIT specification means you will take complete charge of output.

With INPFIL EXIT specified, you must, therefore, use E15 user exit routine (for a sort application) or E32 user exit routine (for a merge or copy application) to read the input files and pass the records one at a time to DFSORT/VSE. For any application with OUTFIL EXIT specified, you must use E35 user exit routine to take the output records one at a time from DFSORT/VSE and write them to the output file. You can, if you wish, open and close the files at the same user exits. However, your routines will be simple to code, and more generalized, if you use the label handling user exits instead, as shown in Figure 41.

Individual input records can be passed to DFSORT/VSE by your E15 user exit routine for a sort application or your E32 user exit routine for a merge or copy application regardless of whether you have specified the EXIT operand in the INPFIL control statement.

4.3.5 Checkpointing

VSE/ESA checkpoint/restart facility has some restrictions:

- Checkpoint/restart does not support the 31-bit environment; the CKPT request is cancelled when issued from a partition that crosses 16 MB virtual.
- Checkpoint/restart does not support data spaces; that is, data spaces that a program may access are not recorded during CKPT requests.

- Checkpoint/restart does not support dynamic partitions.

Only one checkpoint can be taken during a sort application according to *VSE/ESA System Macros Reference* and none during a merge or copy application. If you want DFSORT/VSE to take a checkpoint, you must
| specify the CKPT operand in the SORT control statement. This will cause
| DFSORT/VSE to take a standard checkpoint of virtual storage and work files at the beginning of the final merging phase.

If DFSORT/VSE is subtasked, the CKPT operand will be ignored.

You can handle checkpointing yourself by specifying the CKPT operand in the SORT control statement and by supplying a checkpoint routine at E31 user exit. DFSORT/VSE will set up a checkpoint parameter list but take no checkpoints itself. You must take full responsibility for the checkpoints in your own user exit routine.

DFSORT/VSE will *not* take a checkpoint if E31 user exit is specified for any reason.

4.3.6 Altering, Deleting, and Inserting Records

There are three user exits at which you can manipulate individual records: E15, for records to be sorted; E32, for records to be merged or copied; and E35, for records to be written to the output file.

Subtopics:

4.3.6.1 At Sort Input (E15 User Exit)

Routines at E15 user exit receive control before records are processed. If you have specified the EXIT operand in the INPFIL control statement, routines at E15 user exit must take complete responsibility for all the sort input. See "Control All Input and Output" in topic 4.3.4. The INCLUDE, OMIT, SUM, INREC, and OUTREC functions are performed after E15 user exit.

Altering Records: You can alter the contents of any field in a record and you can change the length of a logical record by adding or deleting fields after the last control field, as long as you comply with the record description supplied on the RECORD control statement.

If you do change the contents or alter the length of any record, you must ensure that the fields in the changed record still match the fields specified in the SORT, INCLUDE, OMIT, SUM, INREC, and OUTREC control statements. If the length of a fixed-length record is changed, the modified length must be specified by the L2 value in the RECORD control statement, unless ADDROUT or ADDROUT=D operand is specified in the OPTION control statement. If the length of a variable-length record is changed such that the maximum, minimum, or modal record length is altered, the respective L2, L4, or L5 value should be specified in the RECORD control statement, unless ADDROUT or ADDROUT=D operand is specified in the OPTION control statement.

Deleting Records: Any record that you do not want in the output file can be deleted. Doing this at input rather than at output (E35 user exit) saves program time. However, the INCLUDE or OMIT function may be used for this purpose instead of a user exit routine.

Inserting Records: Records can be inserted at any time with a routine at E15 user exit.

4.3.6.2 At Merge or Copy Input (E32 User Exit)

E32 user exit functions in two different ways, depending on whether or not you specify EXIT operand in the INPFIL control statement.

When INPFIL EXIT is Not Specified: A routine at E32 user exit can modify the contents of records, including control fields, but it must not be used to alter the length of records.

Input records cannot be inserted or deleted but may be replaced by one of your records.

When INPFIL EXIT is Specified: A routine at E32 user exit has complete responsibility for reading records into the merge or copy, that is, for all operations on the input files: defining them, opening them, and processing their labels.

The contents of the input records can be modified and the record lengths can be altered. Records may also be deleted or inserted.

When the records are ready for merging or copying, your routine passes them, one at a time, to DFSORT/VSE; DFSORT/VSE performs the necessary processing, writes a record to the output file (or passes the record to your E35 user exit routine), and then returns to your E32 user exit routine to obtain the next record to be merged or copied.

4.3.6.3 At Output (E35 User Exit)

Altering Records: You can add, modify, or delete fields anywhere in the records. If the maximum length of the records is changed, the modified length must be specified by the L3 value in the RECORD control statement, unless ADDROUT, ADDROUT=D operand or INREC, or OUTREC control statement is specified.

Deleting Records: Any unwanted records can be deleted.

Inserting Records: Records can be inserted at any time but your user exit routine must insert them in the correct sequence.

4.3.7 Summing Records

You can sum records in the output file by using the E35 user exit. However, you can also use SUM control statement to accomplish this without a user exit. See "SUM Control Statement" in topic 3.18.

4.3.8 Modifying for VSAM Processing

There are three user exits which can be used in conjunction with VSAM files to supply passwords or an exit list. The user exits are E18 for sort input, E38 for merge or copy input, and E39 for output files.

Subtopics:

4.3.8.1 Supplying Password List

Your password routine at E18 user exit is entered only once, and must, therefore, supply all necessary passwords for sort input files. The same applies to E38 user exit, where your routine must supply all necessary passwords for merge or copy input files.

4.3.8.2 Supplying Exit List

Exit list must be constructed using the EXLST VSAM macro, and observe all conventions governing VSAM exit lists. Details are given in *VSE/VSAM Library*.

VSAM exits are not entered for null files.

4.3.9 Closing Files

You can write your own user exit routines to close files and perform any necessary housekeeping. For these purposes you should use E11, E17, E31, and E37 user exits.

4.3.10 Terminating DFSORT/VSE

You can write your own user exit routines to terminate DFSORT/VSE before all records have been processed. For these purpose you should use E15, E32, and E35 user exits.

4.4 Addressing and Residence Modes for User Exit Routines

To allow user exit routines to reside above or below 16 MB virtual, and to use either 24-bit or 31-bit addressing mode, DFSORT/VSE supplies these features:

- To ensure that DFSORT/VSE enters your user exit routine with the correct addressing mode, you must observe these rules:
  - If the user exit routine name is specified in a MODS control statement, the user exit routine is entered with the addressing mode indicated by the linkage editor attributes of the routine (for example, 31-bit addressing mode in effect if AMODE 31 is specified).
  - If the address of the user exit routine is passed to DFSORT/VSE via the parameter list (preloaded user exit routine), the user exit routine is entered with 24-bit addressing mode in effect if bit 0 of the phase 1 or phase 3 branch table addresses in the list is 0 or with 31-bit addressing mode in effect if bit 0 of the phase 1 or phase 3 branch table addresses in the list is 1. See for details Chapter 5, "Invoking DFSORT/VSE from a Program" in topic 5.0.
- User exit routines can return to DFSORT/VSE with either 24-bit or 31-bit addressing mode in effect. The return address that DFSORT/VSE placed in register 14 must be used.

- DFSORT/VSE handles all addresses (that is, the pointer to the parameter list and the addresses in the parameter list) as follows:
  - If the user exit routine is entered with 24-bit addressing mode in effect, DFSORT/VSE passes clean (zeros in the first 8 bits) 24-bit addresses to the user exit routine. Such a user exit must pass 24-bit addresses back to DFSORT/VSE.
  - If the user exit routine is entered with 31-bit addressing mode in effect, DFSORT/VSE passes clean 24-bit addresses to the user exit routine. Such a user exit routine must pass 31-bit addresses or clean (zeros in the first 8 bits) 24-bit addresses back to DFSORT/VSE.

4.5 How User Exit Routines Affect DFSORT/VSE Performance

Before writing user exit routines, consider the following factors:

- | Your routines occupy virtual storage that would otherwise be available
  | to DFSORT/VSE. Because its virtual storage is restricted, DFSORT/VSE
  | might need to perform extra passes to sort the data. This, of course, increases sorting time.

- | Virtual storage that follows your routines may not be available for
  | use by DFSORT/VSE. For this reason, self-relocating and loader-relocatable routines are recommended. These subjects are discussed in *VSE/ESA System Macros User's Guide*.

  When you use relocatable routines, remember to specify their length in the MODS control statement, as described in Chapter 3, "Using
  | DFSORT/VSE Program Control Statements" in topic 3.0. DFSORT/VSE can
  | then use virtual storage efficiently (usually by placing your routines at the highest addresses in your partition program area).

- 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/VSE. You must remember this when designing your routines.

- If you have applications (such as COBOL or PL/I applications) that call DFSORT/VSE and then use DFSORT/VSE user exit routines (E15, E35, or both), be sure to turn DFSORT/VSE's STXIT off to avoid performance problems. To turn DFSORT/VSE's STXIT off, pass OPTION NOSTXIT to DFSORT/VSE via SYSIPT or via a VSE Librarian member. Please refer to appropriate compiler's Programming Guide for information on how to pass the OPTION control statement to DFSORT/VSE. Note that applications such as COBOL or PL/I may generate E15, E35, or both user exit routines automatically.

- Using INCLUDE, OMIT, INREC, OUTREC, and SUM control statements instead of your routines allows DFSORT/VSE to perform more efficiently.

Figure 42. Uses of User Exits

Uses of User Exits	Initial Sorting Phase				Final Merging Phase					
	E11	E15	E17	E18	E31(4)	E32	E35(4)	E37(4)	E38	E39(4)
Take checkpoints					X(1)					
Process labels	X		X		X			X		
Open files	X				X					
Close files	X		X		X			X		

Supply password list for VSAM files				X					X	X
Supply VSAM exit list				X					X	X
Read input to a sort		X								
Count input records		X								
Insert/delete records		X					X			
Lengthen/ shorten records		X(3)					X			
Alter records		X(3)				X(3)	X			
Read input to a merge or copy						X(2)				
Sum records							X			
Replace records in a merge or copy						X				
Write output							X			
Terminate DFSORT/VSE		X				X	X			

Notes:

(1) Only if CKPT operand is specified in the SORT control statement.

(2) Only if EXIT operand is specified in the INPFIL control statement.

(3) If control field lengths are changed, they must match those given in the SORT or MERGE control statement.

(4) Can be used at final merging phase (incore sort).

4.6 Loading and Linking to User Exit Routines

DFSORT/VSE will load your user exit routines. All routines must be treated as an entity and cataloged under a unique name. This is the name you specify in the MODS control statement.

When DFSORT/VSE is invoked with the LOAD macro, you have the option of loading your own user exit routines. If you do so, you must inform DFSORT/VSE of their location in the parameter list and also supply a MODS control statement. The parameter list is explained in detail in Chapter 5, "Invoking DFSORT/VSE from a Program" in topic 5.0.

Subtopics:

4.6.1 Passing Control

You have to provide the entry point of each of your own user exit routines in a branch table at the beginning of the module.

The format of the branch tables is shown in Figure 43. If any user exit is not used, it must still have an entry in the branch table, as in the example in Figure 44, for the initial sorting phase.

Initial Sorting Phase Branch Table:

```
         USING ENTRY1,15
ENTRY1   B     E11          ENTRY POINT FOR E11 USER EXIT ROUTINE
         B     E15          ENTRY POINT FOR E15 USER EXIT ROUTINE
         B     E17          ENTRY POINT FOR E17 USER EXIT ROUTINE
         B     E18          ENTRY POINT FOR E18 USER EXIT ROUTINE
* Programmer's Initial Sorting Phase Processing
* Routines Follow
```

Final Merging Phase Branch Table:

```
         USING ENTRY3,15
ENTRY3   B     E31          ENTRY POINT FOR E31 USER EXIT ROUTINE
         B     E32          ENTRY POINT FOR E32 USER EXIT ROUTINE
         B     E35          ENTRY POINT FOR E35 USER EXIT ROUTINE
         B     E37          ENTRY POINT FOR E37 USER EXIT ROUTINE
         B     E38          ENTRY POINT FOR E38 USER EXIT ROUTINE
         B     E39          ENTRY POINT FOR E39 USER EXIT ROUTINE
* Programmer's Final Merging Phase Processing
* Routines Follow
```

Figure 43. Branch Tables for User Exits

Initial Sorting Phase Branch Table:

```
PH1      B     E11          BRANCH TO ROUTINE CALLED E11
         DC    A(0)         E15 IS NOT USED
         B     E17          BRANCH TO ROUTINE CALLED E17
         B     E18          BRANCH TO ROUTINE CALLED E18
```

Figure 44. Branch Table Example

4.6.2 Use of Registers to Pass Information

DFSORT/VSE uses registers 1, 13, 14, and 15 in the standard way to pass information to your user exit routines.

The registers used by DFSORT/VSE for linkage and communication of parameters observe operating system conventions (see Figure 45).

Figure 45. Register Conventions

Register	Use
1	Contains a pointer to a list of addresses (parameter list), each pointing to an item of information. The contents of the parameter lists are different for each user exit and are described below for each user exit. When your routine needs to pass a return code back to DFSORT/VSE, the same parameter list conventions must be used. The valid return codes for each user exit are also described below.
13	Contains the address of a save area nine doublewords long, in which your routine should save the contents of the registers and must restore the registers before returning control to DFSORT/VSE.
14	Contains the return address. Your routine returns control by branching to this address.
15	Contains the address of branch table. You can use this as a base register at the start of processing.

Note:

If dataspace sorting is in effect, your user exit routines must not change the contents of access registers.

If you pass an invalid return code, an appropriated error message will be issued and DFSORT/VSE will terminate.

Figure 46 shows the general method used by DFSORT/VSE to pass parameters at the different user exits.

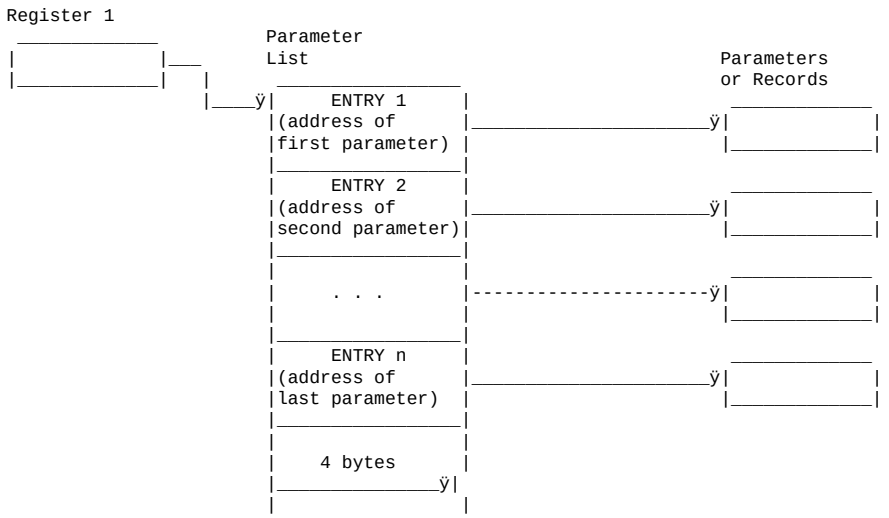


Figure 46. General Method for Passing Parameters

4.7 E11 User Exit

The following notes apply to both disk and tape files with the exception of references to trailer labels. Trailer labels are only relevant to tape files and all references to them should be ignored when disk files are involved.

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Reserved	
2	4	Reserved	
3	8	Address of previous volume unit	Logical unit number of volume just processed (2 bytes)(*)
4	12	Address of next volume unit	Logical unit number of next volume to process (2 bytes)(*)
5	16	Address of block count	Block count (4 bytes)
6	20	Address of SYS number table	SYS number table (see below)

Note:

(*)Only the SYS number is given, not the full CCB format.

Figure 47. E11 User Exit Parameter List

On First Entry: When your E11 user exit routine first receives control, parameter list entries 3 and 4 contain zeros. You must open the first volume of input file and process its header label. You will find its symbolic unit number, in binary, in the second byte of the SYS number table pointed to by parameter list entry 6. Parameter list entry 5 is not used.

On Subsequent Entries: If there is an address in parameter list entry 3, process the trailer label for the volume on the unit pointed to, using the block count indicated in parameter list entry 5. If the contents of parameter list entries 3 and 4 are different (and parameter list entry 3 is not 0), this is the last volume of the file so you must also close the file. Note that you will not get control to close the last input file--this must be done at E17 user exit.

If there is an address in parameter list entry 4, open the volume on the unit pointed to and process its header labels.

SYS Number Table: The SYS number table contains 20 one-byte entries, one for each device in a 4×5 matrix. The first entry is for the output file, the second through tenth entries are for the input files, the eleventh through nineteenth entries are for the work files, and the twentieth entry is for the checkpoint file. Each 1-byte entry contains the symbolic unit number in binary used by DFSORT/VSE for the file in question. If there are fewer than the maximum number of files, the unneeded bytes contain zeros.

Subtopics:

4.7.1 Examples of Label Processing

The examples in Figure 48 in topic 4.7.1.3 show the label processing required of a user routine at E11 user exit for various input configurations. The figure also shows the contents of parameters each time E11 user exit routine receives control and indicates the processing carried out at E17 user exit. Any references to trailer labels only apply to tape files and should be ignored when using disk files.

Subtopics:

4.7.1.1 Example 1--One Multivolume Input File

This example illustrates the nonstandard or user standard label processing required for a single input file that is contained on three volumes:

1. E11 user exit is first activated to open the file and process the header label of the first volume. All label processing parameters contain zeros; only parameter 6 is of interest.

2. E11 user exit is given control the second time to process the trailer label of the first volume (for tape volumes only) and the header label of the second volume (for both tape and direct access volumes).

3. The third and last time E11 user exit receives control, the user routine must again process a trailer label (tape only) and a header label (tape and direct access).

4. A routine at E17 user exit must process the final tape trailer label and close the file.

4.7.1.2 Example 2--Multifile, Multivolume Input

In this example, the input consists of two files. The first is assigned to logical unit SYS002 and extends over two volumes. The second file is on SYS003 and also extends over two volumes.

E11 user exit receives control three times to process the labels of the first file. On the third occasion, it also opens the second file and processes its first header labels.

The fourth time, it handles the second file's labels. A routine at E17 user exit processes the final trailer label (tape only) and closes the second file.

4.7.1.3 Example 3--Three Multivolume Input Files, One With Standard Labels

In this example, the first and third files have nonstandard or user standard labels, while the second file has standard labels (the second file could also be an unlabeled tape file).

The processing required at E11 user exit is exactly the same as described in Example 2, since the system completely handles standard labels.

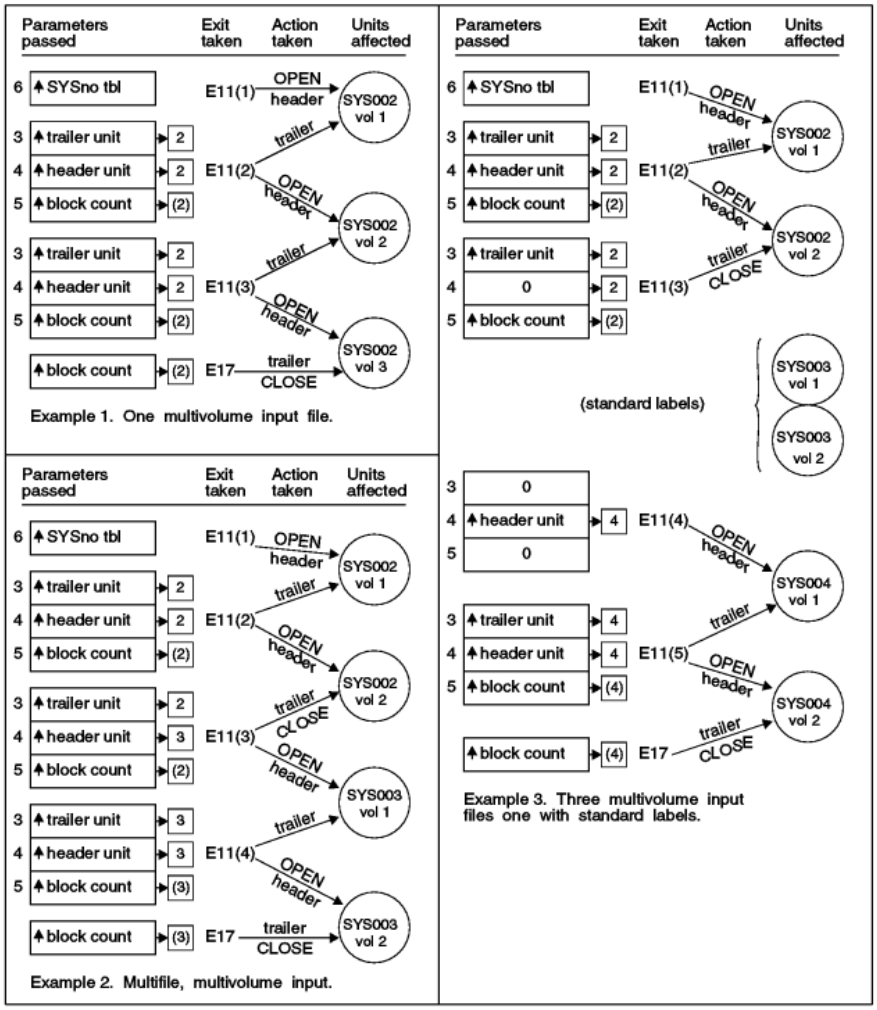


Figure 48. Label Processing at E11 and E17 User Exits

4.8 E15 User Exit

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Address of record (or zeros)(*)	Next record to process
2	4	Address of action word	Action word (1 word)
3	8	Address of record length vector	L1 L2 L3 L4 L5 (5 words)

4	12	Address of record type	Record type code (1 byte)

Note:

(*) Zeros when INPFIL EXIT specified, and at end of file.

Figure 49. E15 User Exit Parameter List

Parameter List Entry 1 points to the record to be processed by your user exit routine.

Parameter List Entry 2 tells DFSORT/VSE, by means of the action word or return code, what to do when control is returned to DFSORT/VSE. You must insert the appropriate code in the rightmost byte of the action word. Valid codes are:

- 00 X'00' Process a record(*)
- 04 X'04' Delete a record(*)
- 08 X'08' Do not return to this user exit
- 12 X'0C' Insert a record
- 16 X'10' Terminate DFSORT/VSE

(*) Invalid when EXIT operand is specified in the INPFIL control statement.

Parameter List Entry 3 gives the address of a 20-byte (5 words) area containing the record length values for L1 through L5 (see the RECORD control statement).

Parameter List Entry 4 gives the address of a byte containing the record type code. Bit 0 is on and bit 1 is off for fixed-length records, and bit 1 is on and bit 0 is off for variable-length records. Other bits may be used by DFSORT/VSE for flags.

Procedure: Normally DFSORT/VSE will read the first input record and then pass control to your user exit routine, with the address of record in parameter list entry 1.

1. If the record is acceptable, your routine should return a code of X'00'.
2. If you want to delete it, your routine should return a code of X'04'.
3. If you want to pass a record to sort which your routine has read in from a different file ("insert"), you put your record's address in parameter list entry 1 and return a code of X'0C'. Next time, parameter list entry 1 will have been restored to its former value.
4. If you want to change the record, your routine must move it to a work area, change it, put its new address in parameter list entry 1, and return a code of X'00'.

This process is repeated for every input record until your routine returns a code of X'08' or X'10'.

Parameter list entries 3 and 4 make it possible for you to write a generalized E15 user exit routine, by giving you all the information you need about record type and length.

If you have specified INPFIL EXIT, only step 3 is applicable: Parameter list entry 1 will always contain zeros which you must replace with the address of the record you have read in. Note that return codes X'00' and X'04' are invalid with INPFIL EXIT specified.

Subtopics:

4.8.1 E15 User Exit Routine Examples

Three examples of coding for routines at E15 user exit are shown below.

Subtopics:

4.8.1.1 Example 1

This example shows how an E15 user exit routine can insert records.

```
EXIT15   CSECT
         PRINT NOGEN
         USING *,15
START    DC    A(0)          EXIT E11 NOT USED
         B     E15           ENTRY FOR E15 USER EXIT
         DC    A(0)          EXIT E17 NOT USED
         DC    A(0)          EXIT E18 NOT USED
E15      SAVE  (14,12)       SAVE REGISTERS
         DROP  15
         USING START,11      SET UP NEW BASE REGISTER
         LR    11,15         REG 15 POINTS TO 1ST BYTE OF RTN
         LA    4,0(,1)       ADDR OF RECORD ADDR PARAMETER
         L     5,4(,1)       ADDR OF ACTION WORD
         LA    6,CARDIN      GET ADDRESS OF DTF
         CLI   FIRSTSW,X'00' TEST IF FIRST TIME
         BNE   GET           NO, BYPASS OPEN
         OPENR (6)           OPEN THE FILE
         MVI   FIRSTSW,X'FF' SET FIRST TIME SWITCH
GET      GET   (6)           READ A RECORD
         LA    7,RCDIN       GET ADDR OF RECORD JUST READ
         ST    7,0(,4)       STORE IN PARAMETER LIST
         LA    7,12          CODE FOR ACTION WORD = INSERT RCD
RETURN   ST    7,0(,5)       STORE IT IN PARAMETER LIST
         RETURN (14,12)      BACK TO SORT
EOF      LA    7,8           SET ACTION CODE TO 8 = END OF FILE
         CLOSER (6)          CLOSE THE FILE
         B     RETURN        BACK TO SORT
*
RCDIN    DC    CL80' '       INPUT AREA
FIRSTSW  DC    X'00'         FIRST TIME SWITCH
*
CARDIN   DTFCD DEVADDR=SYSIPT,IOAREA1=RCDIN,EOFADDR=EOF
IJCFZIZ  CDMOD
         END
```

4.8.1.2 Example 2

This example shows E15 user exit routine which compares two bytes of each input record with a constant. If the compare is not high, the record is printed and deleted from the input.

```
EXIT15   CSECT
         PRINT NOGEN
         USING *,15          REG15 POINTS TO START OF EXIT
START    DC    A(0)          E11 NOT USED
         B     E15           ENTRY FOR E15 USER EXIT
         DC    A(0)          E17 NOT USED
         DC    A(0)          E18 NOT USED
*
         DROP  15            REG15 TO BE USED BY PUT RTN
         USING START,11      REG11 AS BASE REGISTER
E15      SAVE  (14,12)       SAVE REGISTERS
         LR    11,15         SET UP BASE REGISTER
         LA    4,PRINTER     GET ADDR OF PRINTER DTF
         LM    2,3,0(1)      LOAD ADDR OF RECORD & ACTION WORD
         LTR   2,2           TEST IF END OF FILE
         BZ    EOF           YES, BRANCH
*
         CLC   10(2,2),FIRST   CHECK FIRST CONTROL FIELD
         BH    NOACT         HIGH: SET RETURN CODE
*
OPEN     CLI   SW1,X'FF'     TEST IF FIRST TIME
         BE    PUT           IF NOT, BRANCH
         OPENR (4)           OPEN PRINTER FILE
         CNTRL (4),SK,1      SKIP TO CHANNEL 1
         MVI   SW1,X'FF'     BYPASS OPEN & CTL NEXT TIME
*
PUT      MVC   OUTAREA+1(120),0(2)    MOVE RECORD TO OUTAREA, AND
         PUT   (4)           PRINT IT
         LM    7,8,LINECNT   GET CURRENT AND MAX LINECOUNT
         LA    7,1(,7)       INCREMENT CURRENT LINE NUMBER
         ST    7,LINECNT     SAVE IT
         CR    7,8           TEST IF END OF PAGE
         BH    SKIP1         YES, GO SKIP TO CHANNEL 1
DELETE   LA    4,4           LOAD ACTION CODE 4 (DELETE)
         B     RETURN        RETURN TO SORT
*
NOACT    SR    4,4           LOAD ACTION CODE 0 (NO ACTION)
         B     RETURN        GO BACK TO SORT
*
SKIP1    CNTRL (4),SK,1      SKIP TO CHANNEL 1
         XC    LINECNT,LINECNT    CLEAR CURRENT LINE NUMBER
         B     DELETE        GO BACK
*
EOF      CLOSER (4)          CLOSE PRINTER FILE
         LA    4,8           LOAD ACTION CODE 8 (NO RETURN)
RETURN   ST    4,0(,3)       STORE ACTION WORD
         RETURN (14,12)
*
LINECT   DC    F'0',F'50'       CURRENT AND MAX LINECOUNT
OUTAREA  DC    CL121' '         OUTPUT AREA
FIRST    DC    C'22'            CONSTANT TO COMP. WITH CTL FLD
PRINTER  DTFPR DEVADDR=SYSLST,IOAREA1=OUTAREA,CONTROL=YES
IJDFCZZZ PRMOD CONTROL=YES
SW1      DC    X'00'
         END
```

4.8.1.3 Example 3

This example shows how to compile, linkedit, and use an E15 user exit routine to allow DFSORT/VSE to read instream SYSIPT data.

```
// JOB E15CARDS ASSEMBLE
// LIBDEF *,CATALOG=PRD2.END
// OPTION CATAL,LIST,NOXREF
   PHASE E15CARDS,*
// EXEC ASMA90,SIZE=(ASMA90,64K),                                    C
               PARM='EXIT(LIBEXIT(EDECKXIT(ORDER=EA)))'
         TITLE 'E15CARDS  --  GET CARD IMAGES FOR SORT INPUT'
*        PRINT NOGEN
*
*  TO USE:
*
*      // EXEC SORT,SIZE=300K
*         ...
*         MODS PH1=(E15CARDS,L800,E15)
*         INPFIL EXIT
*         ...
*      /*
*      INSTREAM DATA
*      /*
*
EXIT15   CSECT
         PRINT NOGEN
         USING *,15
START    DC    A(0)                  EXIT E11 NOT USED
         B     E15                   ENTRY FOR E15 USER EXIT
         DC    A(0)                  EXIT E17 NOT USED
         DC    A(0)                  EXIT E18 NOT USED
E15      SAVE  (14,12)               SAVE REGISTERS
         DROP  15
         USING START,11              SET UP NEW BASE REGISTER
         LR    11,15                 R11 NOW POINTS TO LOAD ADR.
         LA    4,0(,1)               ADDR OF RECORD ADDR PARAMETER
         L     5,4(,1)               ADDR OF ACTION WORD
         LA    6,CARDIN              GET ADDR OF DTF
         CLI   FIRSTSW,X'00'         TEST IF FIRST TIME
         BNE   GETREC                NO, BYPASS OPEN
         OPENR (6)                   OPEN THE FILE
         MVI   FIRSTSW,X'FF'         SET NOT FIRST TIME SWITCH
GETREC   GET   (6)                   READ A RECORD
         LA    7,RCDIN               GET ADDR OF RECORD JUST READ
         ST    7,0(,4)               STORE IN PARAMETER LIST
         LA    7,12                  SET INSERT RECORD CODE
RETURN   ST    7,0(,5)               STORE IT IN ACTION WORD
         RETURN (14,12)              RETURN BACK TO SORT
*
EOF      CLOSER (6)                  AT EOF, CLOSE THE FILE
         LA    7,8                   SET NO MORE DATA CODE
         B     RETURN                GO TO STORE IT IN ACTION
*                                    WORD AND RETURN BACK TO SORT
RCDIN    DC    CL80' '               INPUT AREA
FIRSTSW  DC    C'00'                 SET INITIAL VALUE FOR FIRST TIME
*
CARDIN   DTFCD DEVADDR=SYSIPT,       LOGICAL UNIT BEING READ          C
               IOAREA1=RCDIN,        I/O AREA (GETS CARD IMAGE)       C
               EOFADDR=EOF           WHERE TO GO ON EOF CONDITION
IJCFZIZ0 CDMOD
*
         END
/*
// IF $RC > 0 THEN
//    GOTO $EOJ
// EXEC LNKEDT
/*
// LIBDEF *,SEARCH=(PRD2.END)
// EXEC SORT,SIZE=300K
    OPTION ROUTE=LST,SORTOUT=LST
    INPFIL EXIT
    MODS   PH1=(E15CARDS,L800,E15)
    RECORD TYPE=F,LENGTH=80
    SORT   FIELDS=(1,3,CH,A)
    END
```

```
/*
 ZZZZ
 AAAA
 BBBB
 CCCC
/*
/&
```

Note: Be sure to specify /* after the END statement.

4.9 E17 User Exit

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Address of block count	Block count for last volume of last input file (4 bytes)

Figure 50. E17 User Exit Parameter List

Procedure: Your routine at E17 user exit receives control only once. It must then process the trailer label of the last volume of input file, using the block count passed as a parameter, and close the last input file.

4.10 E18 User Exit

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Request type indicator	
2	4	Address of password list or exit list	
3	8	Address of action word	Action word (1 word)

Figure 51. E18 User Exit Parameter List

Parameter List Entry 3 tells DFSORT/VSE, by means of the action word or return code, what to do when control is returned to DFSORT/VSE. You must insert the appropriate code in the rightmost byte of the action word. Valid codes are:

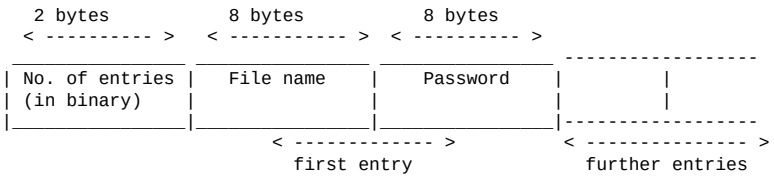
- 00 X'00' No reply
- 04 X'04' Reply provided
- 08 X'08' Do not return to this user exit

Procedure: Your E18 user exit routine is entered twice. The first time, parameter list entry 1 contains C'PWD ', which means that a password list is requested. The second time it contains C'EXL ', which is a request for an exit list.

If you have no reply, return with a code of X'00' in the action word.

Otherwise, put the address of the password list or exit list (whichever has been requested) in parameter list entry 2 and return with a code of X'04'.

Password List: The list must begin with a 2-byte entry count, and continue with a 16-byte entry for each password-protected sort input file:



Exit List: The exit list must be built according to the rules stated in *VSE/ESA System Macros Reference*, or *VSE/VSAM Library*. All routines pointed to by the list must use standard VSAM linkage to return to VSAM. They must not use register 13, which is used by VSAM; instead, they must provide their own save area.

For end-of-file processing, do **not** use the EODAD exit as DFSORT/VSE will not then be able to detect end of file. Instead, supply a LERAD exit and test the FDBK code for X'04', which indicates end of file. Do not change the code, as DFSORT/VSE also uses it.

Note: The same exit list must be valid for all SORTINn files.

4.11 E31 User Exit

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Reserved	
2	4	Address of device list	Device list
3	8	Address of previous volume unit	Logical unit number of volume just processed (2-byte CCB format)
4	12	Address of next volume unit	Logical unit number of next volume to process (2-byte CCB format)

5	16	Address of block count	Block count for trailer device (4 bytes)
6	20	Address of SYS number table	SYS number table

Figure 52. E31 User Exit Parameter List

Procedure: Your E31 user exit routine will receive control for both file handling and checkpointing. If only checkpointing is requested, then the E31 user exit is only entered once. If nonstandard label files are specified in the LABEL option, then the E31 user exit will be entered each time a nonstandard label input or output volume is required.

As with E11 user exit, you use parameter list entries 3 and 4 to determine what action is needed. Either they will both contain all zeros, in which case this is the first entry, or they will both contain addresses. The addresses will be the same, because there are no new files to be opened, only new volumes.

On First Entry: Parameter list entries 3 and 4 both contain zero.

If you want to take a checkpoint, parameter list entry 2 points to a checkpoint device list. The device list begins with a 2-byte count field which contains the number of work file extents that the checkpoint device list will contain. The count field is followed by one 4-byte entry for each work file extent. The 4-byte entries have the form:

2 bytes	2 bytes
unit code	X'0000'

Two 4-byte fields at the end of the device list give the DFSORT/VSE start address and the DFSORT/VSE storage size. The checkpoint device list has the form:

2 bytes	2 bytes	2 bytes		2 bytes	2 bytes	4 bytes	4 bytes
count field	unit code	X'0000'	. . .	unit code	X'0000'	DFSORT/VSE start address	DFSORT/VSE storage size

For more details of checkpointing device list, see *VSE/ESA System Macros Reference* and *VSE/ESA System Macros User's Guide*.

If you are handling files, open all files with nonstandard labels (input, output, or both) and process their header labels. You will find their SYS numbers in the SYS number table pointed to by parameter list entry 6: the first byte gives the number (in binary) of the output file, and the following nine bytes give the numbers of the input files. If you have fewer than nine input files, the unneeded bytes contain zeros. See "E11 User Exit" in topic 4.7 for more details of the SYS number table.

On Subsequent Entries: If parameter list entry 3 contains an address, process the trailer label of the volume pointed to, using the block count indicated in parameter list entry 5. Your routine will not be given control to close the files; this must be done at E37 user exit.

If parameter list entry 4 contains an address, open the volume pointed to and process its header labels.

Figure 53 shows the actions that must be taken by the routines at E31 and E37 user exits when the input to a merge application consists of tape files:

- SYS002, one volume, nonstandard labels
- SYS003, standard labels
- SYS004, two volumes, nonstandard labels

Output consists of one disk file:

- SYS001, three volumes, nonstandard labels
- (user-supplied labels)

Figure 53. Using E31 and E37 User Exits with a Merge Application

Routine given control, parameters passed	**SYS001**			**SYS002**	**SYS004**	
	1	**2**	**3**	**1**	**1**	**2**
E31 (1st time) 3=0,4=0	OPEN header			OPEN header	OPEN header	
E31 (2nd time) 3--SYS001 4--SYS001		OPEN header				
E31 (3rd time) 3--SYS004 4--SYS004					trailer	OPEN header
E31 (4th time) 3--SYS001 4--SYS001			OPEN header			
E37			CLOSE	trailer CLOSE		trailer CLOSE

Subtopics:

- 4.11.1 E31 and E37 User Exit Routines Example

4.11.1 E31 and E37 User Exit Routines Example

This example shows routines which open the output file, write the file label, and finally close the file.

```
LABEXITS CSECT
         PRINT NOGEN
         USING *,15          REG 15 POINTS TO START OF EXIT
START    B     E31           ENTRY POINT FOR EXIT E31
         DC    A(0)          EXIT E32 NOT USED
         DC    A(0)          EXIT E35 NOT USED
         B     E37           ENTRY POINT FOR EXIT E37
         DC    A(0)          EXIT E38 NOT USED
         DC    A(0)          EXIT E39 NOT USED
*
         DROP  15
```

```
         USING START,11      USE REG 11 AS BASE REGISTER
*
***START OF ROUTINE FOR EXIT E31***
*
E31      SAVE  (14,12)       SAVE REGISTERS
         LR    11,15         SET UP BASE REGISTER
         LA    6,SORTOUT     GET ADDRESS OF DTF
         OPENR (6)           OPEN OUTPUT FILE, WRITE LABELS
         RETURN (14,12)      RESTORE REGISTERS, BACK TO SORT
*
***START OF ROUTINE FOR EXIT E37***
*
E37      SAVE  (14,12)       SAVE REGISTERS
         LR    11,15         SET UP BASE REGISTER
         MVI   E37SW,X'FF'   SET E37 SWITCH FOR LABEL RTNE
         LA    6,SORTOUT     GET ADDRESS OF DTF
         CLOSER (6)          CLOSE THE FILE
         RETURN (14,12)
*
LAB      LA     0,UHL1       GET ADDRESS OF FIRST USER HEADER LAB
         CLI    FIRSTSW,X'FF' TEST IF FIRST TIME
         BE     LAB1         NO, BRANCH
         MVI    FIRSTSW,X'FF' SET FIRST TIME SWITCH
         LBRET  2             LBRET 2 = MORE LABELS TO PROCESS
*
LAB1     CLI    E37SW,X'FF'   TEST IF E37
         BE     LAB2          YES, TIME FOR TRAILER LABEL
         LA     0,UHL2        ADDRESS OF 2ND USER HEADER LABEL
         LBRET  1             LBRET 1 = LAST LABEL PROCESSED
*
LAB2     LA     0,UTL1        LOAD ADDRESS OF TRAILER LABEL
         LBRET  1             WRITE TRAILER LABEL - (LAST LABEL0)
UHL1     DC     CL80'UHL1EXAMPLE OF A USER HEADER LABEL'
UHL2     DC     CL80'UHL2ANOTHER EXAMPLE OF A USER HEADER LABEL'
UTL      DC     CL80'UTL1THIS IS AN EXAMPLE OF A USER TRAILER LABEL'
*
E37SW    DC     X'00'        USED IN LABEL RTNE TO IDENTIFY E37
FIRSTSW  DC     X'00'        1ST OR SEQUENCE ENTRY
*
SORTOUT  DTFPH  TYPEFLE=OUTPUT,LABADDR=LAB,DEVADDR=SYS001
*
         END
```

Figure 54. E31 and E37 User Exit Routines Example

4.12 E32 User Exit

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Address of next record (zeros when INPFIL EXIT specified)	Next record to process
2	4	Address of input file number	File number in hexadecimal code (1 word)
3	8	Address of action word	Action word (1 word)
4	12	Address of record length vector	L1 L2 L3 L4 L5 (5 words)
5	16	Address of record type	Record type switch (1 byte)

Figure 55. E32 User Exit Parameter List

The action word is only required when INPFIL EXIT is specified.

Parameter List Entry 3 tells DFSORT/VSE, by means of the action word or return code, what to do when control is returned to DFSORT/VSE. You must insert the appropriate code in the rightmost byte of the action word. Valid codes are:

- 08 X'08' No more records to come from a specified file
- 12 X'0C' Insert a record
- 16 X'10' Terminate DFSORT/VSE

Parameter List Entry 4 gives the address of a 20-byte (5 words) area containing the record length values for L1 through L5 (see the RECORD control statement).

Parameter List Entry 5 gives the address of a byte containing the record type code. Bit 0 is on and bit 1 is off for fixed-length records, and bit 1 is on and bit 0 is off for variable-length records. Other bits may be used by DFSORT/VSE for flags.

Procedure without INPFIL EXIT: When INPFIL EXIT is not specified, only parameter list entries 1, 4, and 5 are passed. DFSORT/VSE reads the first record from the first input file and then passes control to your routine, with the record's address in parameter list entry 1. Your routine can accept the record or replace it with a new one of the same length by changing parameter list entry 1 to point to the new record. Your routine should then return control to DFSORT/VSE. No return codes are passed back. This process is repeated until the input is exhausted.

Procedure with INPFIL EXIT: If you have specified INPFIL EXIT, the procedure is different. When your routine first receives control, parameter list entry 1 contains zeros and parameter list entry 2 contains a pointer to a word containing a hexadecimal code in the rightmost byte which specifies the input file from which the next record should be obtained. The codes are:

- X'00' File 1
- X'04' File 2
- X'08' File 3
- X'0C' File 4
- X'10' File 5
- X'14' File 6
- X'18' File 7
- X'1C' File 8
- X'20' File 9

Your user exit routine should then:

1. Open all input files and do any necessary label processing.
2. Read the first block of records from the first file.
3. Put the address of the first record in parameter list entry 1.
4. Put X'0C' ("insert a record") in the action word pointed to by parameter list entry 3.
5. Return control to DFSORT/VSE.

On each subsequent entry, pass a record to DFSORT/VSE in the same way from the file requested in parameter list entry 2.

When you have no more input on the requested file, close the file (process labels as necessary) and return with zeros in parameter list entry 1 and X'08' in the action word. DFSORT/VSE will then request input from a different file until your routine has returned a code of X'08' for each of the input files. If you need to terminate DFSORT/VSE before end of input (abnormal termination), your routine should return a code of X'10'.

Subtopics:

4.12.1 E32 User Exit Routine Example

Figure 56 in topic 4.12.1 is an example of an E32 user exit routine when the INPFIL EXIT is specified.

```
PH3RTN   CSECT
         PRINT NOGEN
         USING *,15
START    DC    A(0)          E31 NOT USED
         B     E32           ENTRY POINT FOR EXIT E32
         DC    A(0)          E35 NOT USED
         DC    A(0)          E37 NOT USED
         DC    A(0)          E38 NOT USED
         DC    A(0)          E39 NOT USED
         DROP  15
         USING START,12
E32      SAVE  (14,12)       SAVE REGISTERS
         LR    12,15         LOAD BASE REGISTER
         ST    13,SAVE13     SAVE REG 13
         LM    2,3,4(1)      LOAD PARAMETERS
         LR    4,1           SAVE PARAMETER POINTER
*
*        REGISTER 2 NOW POINTS TO FILE NUMBER INDICATOR
*        REGISTER 3 NOW POINTS TO ACTION WORD
*        REGISTER 4 NOW POINTS TO PARAMETER LIST
*
         CLI   FIRST,X'FF'   IS THIS THE FIRST TIME?
         BE    FILENO        NO, BRANCH
         OPEN  MASTER,WEEK   YES, OPEN FILES
         MVI   FIRST,'X'FF'  SET 'FILES OPEN' INDICATOR
*
FILENO   CLI   3(2),X'00'    FILE 1?
         BE    GETMAST       YES, READ FROM MASTER FILE
         CLI   3(2),X'04'    FILE 2?
         BE    GETWEEK       YES, READ FROM WEEKLY UPDATE FILE
*        IF NO BRANCH WAS TAKEN, THIS IS AN ERROR, FORCE DUMP
ERROR    DUMP
*
GETMAST  CLI   MASTOUT,X'FF' FILE ALREADY CLOSED?
         BE    ERROR         YES, ERROR
         GET   MASTER        ELSE, READ A RECORD
         B     INSERT        GO TO SEND IT TO MERGE
*
GETWEEK  CLI   WEEKOUT,X'FF' FILE ALREADY CLOSED?
         BE    ERROR         YES, ERROR
         GET   WEEK          ELSE GET RECORD
*
INSERT   ST    5,0(,4)       STORE ADDRESS OF RECORD
         MVC   3(1,3),ACCEPT SET ACTION WORD
         B     RETURN        RETURN TO MERGE
*
ENDMAST  MVC   3(1,3),END    SET ACTION WORD
         CLOSE MASTER        CLOSE MASTER FILE
         MVI   MASTOUT,X'FF' SET FILE CLOSED INDICATOR
         B     RETURN        RETURN TO MERGE
ENDWEEK  MVC   3(1,3),END    SET ACTION WORD
         CLOSE WEEK          CLOSE WEEKLY UPDATE FILE
         MVI   WEEKOUT,X'FF' SET FILE CLOSED INDICATOR
*
```

```
RETURN   L     13,SAVE13     RESTORE REG 13
         RETURN (14,12)      RESTORE REGISTERS AND RETURN TO MERGE
*
*        RETURN CODES FOR MERGE
ACCEPT   DC    X'0C'         INSERT RECORD (ACCEPT)
END      DC    X'08'         END OF FILE
*
MASTOUT  DC    X'00'         MASTER CLOSED INDICATOR
WEEKOUT  DC    X'00'         WEEK CLOSED INDICATOR
FIRST    DC    X'00'         FIRST TIME INDICATOR
SAVE13   DC    F'0'
INBUFM   DS    100F          INPUT BUFFER FOR MASTER
INBUFW   DS    100F          INPUT BUFFER FOR WEEK
*
MASTER   DTFSD BLKSIZE=400,DEVADDR=SYS020,RECFORM=FIXBLK,             C
               RECSIZE=80,IOREG=(5),ERROPT=SKIP,DEVICE=3380,          C
               IOAREA1=INBUFM,EOFADDR=ENDMAST
WEEK     DTFSD BLKSIZE=400,DEVADDR=SYS021,RECFORM=FIXBLK,             C
               RECSIZE=80,IOREG=(5),ERROPT=SKIP,DEVICE=3380,          C
               IOAREA1=INBUFW,EOFADDR=ENDWEEK
         END
```

Figure 56. E32 User Exit Routine Example

4.13 E35 User Exit

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Address of current record	Current record
2	4	Address of previous record	Previous record now in the output area
3	8	Address of action word	Action word (1 word)
4	12	Address of sequence check word	Sequence check word (1 word)
5	16	Address of record length vector	L1 L2 L3 L4 L5 (5 words)
6	20	Address of record type	Record type code (1 byte)

Figure 57. E35 User Exit Parameter List

Parameter List Entry 1 points to the record currently selected for output. When output is exhausted, it contains all zeros.

Parameter List Entry 2 points to the record most recently moved to the output area. Until the first record has been moved out, it contains zeros.

Parameter List Entry 3 tells DFSORT/VSE, by means of the action word or return code, what to do when control is returned to DFSORT/VSE. You must insert the appropriate return code in the rightmost byte of the action word. Valid codes are:

- 00 X'00' Process a record
- 04 X'04' Delete a record
- 08 X'08' Do not return to this user exit
- 12 X'0C' Insert a record
- 16 X'10' Terminate DFSORT/VSE

Parameter List Entry 4 points to the sequence checkword, which contains zeros; it is for use if you want to insert records which are out of sequence.

Parameter List Entry 5 gives the address of a 20-byte (5 words) area containing the record length values for L1 through L5 (see the RECORD control statement).

Parameter List Entry 6 gives the address of a byte containing the record type code. Bit 0 is on and bit 1 is off for fixed-length records, and bit 1 is on and bit 0 is off for variable-length records. Other bits may be used by DFSORT/VSE for flags.

Procedure: When your E35 user exit routine first receives control, parameter list entry 1 will normally point to the first output record. Parameter list entry 2 will contain zeros.

1. If the current record is acceptable, your user exit routine should return a code of X'00'.
2. If you want to delete it, your user exit routine should return a code of X'04'.
3. If you want to insert a record which you yourself have read in, you check it against the current record (parameter list entry 1). If it collates ahead of the current record, put its address in parameter list entry 1 and return a code of X'0C'. Next time, parameter list entry 1 will have been restored to its former value. You can insert a record which is out of sequence; if you do so, you must inform DFSORT/VSE by putting a nonzero value in the sequence check word pointed to by parameter list entry 4.
4. If you want to change the current record, move it to a work area, change it, put its new address in parameter list entry 1, and return a code of X'00'.
5. If you want to sum records, you can sum records in the output file by changing the record in the output area and then, if you want, by deleting the current record. DFSORT/VSE returns to your routine with the address of a new current record and the same record remains in the output area, so that you can sum further. If you do not delete the current record, that record is added to the output area, and its address replaces the address of the previous record in the output area. DFSORT/VSE returns with the address of a new current record.

This process is repeated for every output record until your routine returns a code of X'08' or X'10'.

If you have specified OUTFIL EXIT, your routine returns a code of X'04' every time until parameter list entry 1 contains zeros; then output is exhausted and, after closing the output file, your routine should return a code of X'08'. A code of X'10' is also valid with OUTFIL EXIT.

Subtopics:

- 4.13.1 E35 User Exit Routine Example

4.13.1 E35 User Exit Routine Example

Figure 58 is an example of an E35 user exit routine that is self-relocating and prints all sorted records on SYSLST using physical IOCS.

```
PH3RTN   CSECT
         PRINT NOGEN
         USING *,15          REG 15 POINTS TO START OF EXIT
EXITS    DC    A(0)          EXIT E31 NOT USED
         DC    A(0)          EXIT E32 NOT USED
         B     EXIT35        EXIT E35 ENTRY POINT
         DC    A(0)          EXIT E37 NOT USED
         DC    A(0)          EXIT E38 NOT USED
         DC    A(0)          EXIT E39 NOT USED
EXIT35   STM   14,12,12(13)  SAVE REGISTERS
         LR    4,1           REG 4 POINTS TO LIST OF ADDRESS
*          CONSTANTS PASSED BY DFSORT/VSE
         SR    5,5           CLEAR REG 5
         C     5,0(,4)       1ST ADDR CONSTANT ZERO?
         BE    EOF           YES, NO MORE RECORDS FROM MERGE
         L     6,0(,4)       LOAD ADDR OF REC LEAVING MERGE
         MVC   OUTAREA,0(6)  MOVE RECORD TO PRINTAREA
         LA    1,PRINTCCB    GET ADDR OF PRINTCCB
         CLI   SW1,X'00'     CHECK IF FIRST TIME THROUGH
         BNE   PRINT         NO, PRINT RECORD
         MVI   SW1,X'FF'     SET FIRST TIME SWITCH
         BAL   3,CHA12       GO SKIP TO CHANNEL 1
PRINT    EXCP  (1)           PRINT A RECORD
         WAIT  (1)           WAIT FOR COMPLETION
         LA    6,CHA12       GET ADDR OF CHANNEL 12 ROUTINE
         BAL   3,CHECK       CHECK IF CHANNEL 12 REACHED
         L     6,8(,4)       LOAD ADDR OF ACTION WORD
         MVI   3(6),X'04'    INSERT RETURN CODE (DELETE)
         B     RETURN        GO TO RETURN TO SORT
EOF      L     6,8(,4)       LOAD ADDRESS OF ACTION WORD
         MVI   3(6),X'08'    SET RETURN CODE (DO NOT RETURN)
RETURN   LM    14,12,12(13)  RESTORE REGISTERS
         BR    14            RETURN TO DFSORT/VSE
CHECK    TM    4(1),X'01'    TEST FOR UNIT EXCEPTION
         BOR   6             YES, GO TO CHANNEL 12 ROUTINE
         BR    3             NO, LINK BACK
CHA12    MVI   PRINTCCW,X'8B' MODIFY PRINTCCW TO SKIP TO CH1
         EXCP  (1)           SKIP TO CHANNEL 1
         WAIT  (1)           WAIT FOR COMPLETION
         MVI   PRINTCCW,X'09' RESTORE OP CODE IN PRINTCCW
         BR    3             BRANCH BACK TO SET RETURN CODE
PRINTCCW CCW   X'09',OUTAREA,X'20',L'OUTAREA
PRINTCCB CCB   SYSLST,PRINTCCW CCB
SW1      DC    X'00'         FIRST TIME SWITCH
OUTAREA  DC    CL100' '      PRINT AREA
         END
```

Figure 58. E35 User Exit Routine Example

4.14 E37 User Exit

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Address of block count list	Block count list (4-byte entries): Output file block count Only for a merge and copy: Input file 1 block count . . . Input file n block count

Entry number	**Offset (dec)**	**Contents of entry**	**Points to**
1	0	Request type indicator	
2	4	Address of password list or exit list	
3	8	Address of action word	Action word (1 word)

Figure 59. E37 User Exit Parameter List

Your routine at E37 user exit receives control only once. It must:

- Process the trailer label of the last volume of output, using the first entry in the block count list passed as a parameter, and close the output file.
- For a merge and copy application, carry out the same processing for the last volume of each input file.

See the example supplied with the E31 user exit.

4.15 E38 User Exit

Entry number	**Offset (dec)**	**Contents of entry**	**Points to**
1	0	Request type indicator	
2	4	Address of password list or exit list	
3	8	Address of action word	Action word (1 word)

Figure 60. E38 User Exit Parameter List

Parameter List Entry 3 tells DFSORT/VSE, by means of the action word or return code, what to do when control is returned to DFSORT/VSE. You must insert the appropriate code in the rightmost byte of the action word. Valid codes are:

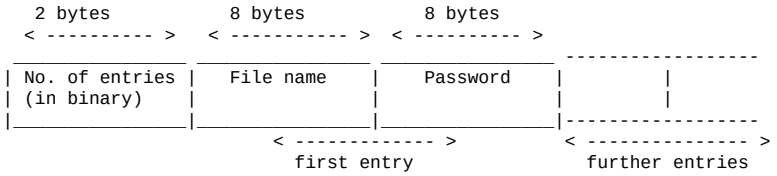
- 00 X'00' No reply
- 04 X'04' Reply provided
- 08 X'08' Do not return to this user exit

Procedure: Your E38 user exit routine is entered twice (for merge and copy applications only). The first time, parameter list entry 1 contains C'PWD '. This means that password list is requested. The second time it contains C'EXL ', which is a request for an exit list.

If you have no reply, return with a code of X'00' in the action word.

Otherwise, put the address of the password list or exit list (whichever has been requested) in parameter list entry 2, and return with a code of X'04'.

Password List: The list must begin with a 2-byte entry count and continue with a 16-byte entry for each password-protected merge or copy input file:



Exit List: The exit list must be built according to the rules in *VSE/ESA System Macros Reference* or *VSE/VSAM Library*. All routines pointed to by the list must use standard VSAM linkage to return to VSAM. They must not use register 13, which is in use by VSAM; instead, they must provide their own save area.

For end-of-file processing, do **not** use the EODAD exit, as DFSORT/VSE will not then be able to detect end of file. Instead, supply a LERAD exit and test the FDBK code for X'04' which indicates end of file. Do not change the code, as DFSORT/VSE also uses it.

Note: The same exit list must be valid for all input files.

4.16 E39 User Exit

Entry number	Offset (dec)	Contents of entry	Points to
1	0	Request type indicator	
2	4	Address of password list or exit list	
3	8	Address of action word	Action word (1 word)

Figure 61. E39 User Exit Parameter List

Parameter List Entry 3 tells DFSORT/VSE, by means of the action word or return code, what to do when control is returned to DFSORT/VSE. You must insert the appropriate code in the rightmost byte of the action word. Valid codes are:

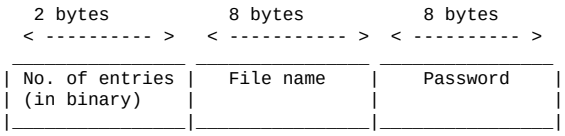
- 00 X'00' No reply
- 04 X'04' Reply provided
- 08 X'08' Do not return to this user exit

Procedure: Your E39 user exit routine is entered twice. The first time, parameter list entry 1 contains C'PWD '. This means that a password list is requested. The second time it contains C'EXL ', which is a request for an exit list.

If you have no reply, return with a code of X'00' in the action word.

Otherwise, put the address of the password list or exit list (whichever has been requested) in parameter list entry 2, and return with a code of X'04'.

Password List: The list is 18 bytes long: an entry count (2 bytes), followed by the name of the output file and its password.



Exit List: The exit list must be built according to the rules in *VSE/ESA System Macros Reference* or *VSE/VSAM Library*. All routines pointed to by the list must use standard VSAM linkage to return to VSAM. They must not use register 13, which is in use by VSAM; instead, they must provide their own save area.

5.0 Chapter 5. Invoking DFSORT/VSE from a Program

Subtopics:

5.1 Introduction

DFSORT/VSE can be invoked dynamically from programs written in Assembler, COBOL, PL/I, and RPG II with the Auto-Report Feature. Specific information on dynamic invocation is covered in the COBOL, PL/I and RPG II with the Auto-Report Feature programming guides. JCL requirements are the same as those for Assembler.

This section explains what you need to know to initiate DFSORT/VSE from within your assembler program using system macros instead of an EXEC job control statement in the input stream.

5.2 What Are System Macros?

System macros are macros provided by IBM for communicating service requests to the control program. You can use these macros only when programming in assembler language. They are processed by the assembler program using macrodefinitions supplied by IBM and placed in the macro library when your control program was installed.

DFSORT/VSE can be initiated from an assembler program by issuing a LOAD followed by a CALL or ATTACH system macro. A partition program area address must be used on the LOAD system macro.

Note: The ATTACH system macro should only be used if you are working in a multiprogramming environment and intend to subtask DFSORT/VSE.

| When your program issues a LOAD system macro, DFSORT/VSE is brought into
| virtual storage. The linkage relationship between your program and DFSORT/VSE depends on the system macro you use to initiate DFSORT/VSE. For a complete description of the system macros and how to use them, refer to *VSE/ESA System Macros Reference*.

5.3 Using System Macros

To initiate DFSORT/VSE processing with the system macros, you must:

- Write the required job control language statements.
- Write DFSORT/VSE program control statements as operands of assembler DC instructions.
- Write a parameter list containing information to be passed to DFSORT/VSE and specify a pointer containing the address of the parameter list.

| When DFSORT/VSE is loaded by another program, it will use all the virtual
| storage from the load point to the upper limit of the partition program
| area (or to the return address of the calling program), unless the virtual
| storage it uses is limited by the STORAGE option or the SIZE operand in the EXEC job control statement. If DFSORT/VSE is subtasked, the STORAGE option must be specified.

Note: DFSORT/VSE is not reusable. It must be loaded each time it is to be used.

5.4 Subtasking

DFSORT/VSE can be subtasked by using the ATTACH system macro. For details, see *VSE/ESA System Macros Reference* and *VSE/ESA System Macros User's Guide*.

Two things must be done if subtasking is to be used:

1. The input, output, and work files must be allocated with unique file names for each task that will or can be run concurrently. This prevents the different tasks from trying to use the same input, output, and work files. The WORKNM and FILNM operands of the OPTION control statement are used to specify the file names.
2. The STORAGE operand of the OPTION control statement must be specified
   | so that DFSORT/VSE knows how much virtual storage it may use. If it is not specified, DFSORT/VSE will try to use the whole partition program area.

If DFSORT/VSE is subtasked, checkpoints cannot be taken by DFSORT/VSE.

When DFSORT/VSE is subtasked, overprinting of messages routed to SYSLST may occur if the main task or other subtasks are using the same SYSLST output. DFSORT/VSE provides a number of options that can be used to control this:

- Specify the ROUTE=xxx option to route DFSORT/VSE messages to the device of your choice (SYSxxx). A system dump, if any, will still appear on SYSLST, and critical messages will also appear on the system console. This parameter can also be set as a default after installation of DFSORT/VSE.
- Route DFSORT/VSE messages to the system console by specifying the ROUTE=LOG option.
- Write only critical messages from DFSORT/VSE by specifying the PRINT=CRITICAL option.
- Write no DFSORT/VSE messages by specifying the PRINT=NONE option.
- Suppress the special formatted dump of DFSORT/VSE areas by specifying the NODUMP option.

If the parameter list pointer is in register 2, DFSORT/VSE will issue a DETACH system macro on completion. If, however, DFSORT/VSE is called from a subtask, register 1 must point to the parameter list, as described above. The calling program must then issue a DETACH system macro for its own subtask.

5.5 Using Job Control Language

Job control statements are required when invoking DFSORT/VSE from another program. The job control statements and their necessary parameters are described in detail in Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0.

5.6 Passing Information with General Registers

The linkage conventions are standard. This means that, when DFSORT/VSE receives control, it expects general registers 1, 2, 13, 14, and 15 to contain the following information:

Register 1
This register must contain the address of a parameter list, the format and contents of which are described below (unless DFSORT/VSE is subtasked).

Register 2
If DFSORT/VSE is subtasked, the address of the parameter list must be in this register instead of in register 1.

Register 13
This register must contain the address of a 9-doubleword save area in which DFSORT/VSE saves the contents of the calling program's registers. The calling program's registers will be restored when DFSORT/VSE completes its processing; they will not be restored, however, when DFSORT/VSE branches to user exit routines.

Register 14
This register must contain the address in the calling program to which DFSORT/VSE will return control upon completion.

Register 15
This register must contain the DFSORT/VSE entry point address.

5.7 Addressing and Residence Modes for an Invoking Program

DFSORT/VSE may be called from a program located below or above 16 MB virtual (any possible calling program's residency mode) and can receive control in either 24-bit or 31-bit addressing mode.

After completion of processing DFSORT/VSE switches to calling program's addressing mode and returns control to the caller.

If DFSORT/VSE is called from a program with AMODE=24, it treats all the addresses (return address, save area address, parameter list address, addresses in parameter list, and so on) as 24-bit addresses. In this case, DFSORT/VSE ignores the high order byte in all parameter list entries. User exit routines must have 24-bit addressing mode and must be located below 16 MB virtual.

DFSORT/VSE passes control to user exit routines in 24-bit addressing mode. DFSORT/VSE provides clean 24-bit addresses (zero in high-order byte) in user exit parameter list entries. This parameter list is located below 16 MB virtual. After DFSORT/VSE receives control from the user exit routine, it treats all returned addresses (address of new record, address of the password list, and so on) as 24-bit addresses.

If DFSORT/VSE is called from a program with AMODE=31, it treats all the addresses (return address, save area address, parameter list address, addresses in parameter list, and so on) as 31-bit addresses. In this case, DFSORT/VSE ignores the high order bit in the parameter list entries (for all entries except for entries that contain any branch table addresses). User exit routines can have 24-bit or 31-bit addressing mode and may be located below or above 16 MB virtual.

DFSORT/VSE determines the user exit addressing mode by checking the high-order bit of the branch table addresses in the DFSORT/VSE parameter list. DFSORT/VSE provides clean 31-bit addresses (zero in the high-order bit) for 31-bit addressing mode user exit routine and clean 24-bit addresses (zero in high-order byte) for 24-bit addressing mode user exit routine in user exit parameter list entries. This parameter list is located below 16 MB virtual. After DFSORT/VSE receives control from the user exit routine, all returned addresses (address of new record, address of the password list, and so on) are treated as 31-bit addresses (for 31-bit addressing mode user exit routine) or as 24-bit addresses (for 24-bit addressing mode user exit routine).

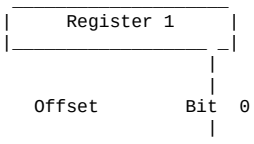
If the user exit routines are loaded by the calling program, DFSORT/VSE treats the high-order bit in the branch table address as follows:

- 0 - 24-bit addressing mode.
- 1 - 31-bit addressing mode.

If the user exit routine is loaded by DFSORT/VSE, it uses AMODE attribute of the routine assigned by the linkage editor.

5.8 Passing Parameters with the DFSORT/VSE Parameter List

The parameter list contains control statement images and other parameter addresses. Figure 62 shows the DFSORT/VSE parameter list. Figure 64 in topic 5.8.4 gives a complete coding example.



(Hex)	(Dec)	Parameter List Contents		Note
0	0	Address of SORT or MERGE statement image		1
4	4	Address of RECORD statement image		1
8	8	Address of INPFIL statement image		1
C	12	Address of OUTFIL statement image		1
10	16	Address of OPTION statement image		1
14	20	Address of MODS statement image		1
18	24	f	Address of branch table for phase 1 preloaded user routines	1
1C	28	Reserved		1
20	32	f	Address of branch table for phase 3 preloaded user routines	1
24	36	Address of return code field		1
28	40	Address of ALTSEQ statement image		2
2C	44	Address of ANALYZE statement image		2
30	48	Address of INCLUDE/OMIT statement image		2
34	52	Address of INREC statement image		2
38	56	Address of OUTREC statement image		2
3C	60	Address of SUM statement image		2
40	64	X'FFFFFFFF'		3

Figure 62. DFSORT/VSE Parameter List

Notes for Figure 62:

1. Required entry. Must appear in the relative position shown. The offset shown is the actual offset of this entry. The first two entries must always be supplied and must contain valid addresses. Any other required entries (from the INPFIL control statement image address to the return code field address) that are not needed can be filled with zeros.

2. Optional entry. Contains an address of the ALTSEQ, ANALYZE, INCLUDE, INREC, OMIT, OUTREC, or SUM control statement image. Can appear anywhere after the required entries. The offset shown is for identification purposes only--the actual offset of this entry can vary. Optional entries can appear in any order. Any optional entry that is not needed can be filled with zeros or omitted.

3 Optional entry. End of the parameter list indicator. Can appear anywhere after the required entries. If the end of the parameter list indicator is supplied, it must be the last entry in the parameter list. The offset shown is for identification purposes only--the actual offset of this entry can vary.

The specifications for each of the parameter list entries follow:

Byte Explanation

0-3
The address of the SORT or MERGE control statement image (required).

4-7
The address of the RECORD control statement image (required).

8-11
The address of the INPFIL control statement image (optional). Zeros if none.

12-15
The address of the OUTFIL control statement image (optional). Zeros if none.

16-19
The address of the OPTION control statement image (optional). Zeros if none.

20-23
The address of the MODS control statement image (optional). Zeros if none.

24-27
The address of the branch table for the Phase 1 preloaded user exit routines (optional). Zeros if none.

f (bit 0) has the following meaning:

- 0 = Enter the user exit routines at the Phase 1 with 24-bit addressing mode in effect (AMODE 24).
- 1 = Enter the user exit routines at the Phase 1 with 31-bit addressing mode in effect (AMODE 31).

28-31
Reserved.

32-35
The address of the branch table for the Phase 3 preloaded user exit routines (optional). Zeros if none.

f (bit 0) has the following meaning:

- 0 = Enter the user exit routines at the Phase 3 with 24-bit addressing mode in effect (AMODE 24).
- 1 = Enter the user exit routines at the Phase 3 with 31-bit addressing mode in effect (AMODE 31).

36-39
The address of the halfword for the return code (optional). Zeros if none.

40-43
The address of the ALTSEQ control statement image or translate table (optional). Zeros if none.

44-47
The address of the ANALYZE control statement image (optional). Zeros if none.

48-51
The address of the INCLUDE or OMIT control statement image (optional). Zeros if none.

52-55
The address of the INREC control statement image (optional). Zeros if none.

56-59
The address of the OUTREC control statement image (optional). Zeros if none.

60-63
The address of the SUM control statement image (optional). Zeros if none.

64-67
The end of the parameter list indicator (4-byte field containing X'FFFFFFFF') can be used after the required entries in the parameter list (first ten entries of the parameter list) to force the completion of parameter list processing (optional).

Subtopics:

5.8.1 Control Statement Images

Figure 63shows examples of control statement images. The images must be coded in the form shown. No extra blanks, continuation characters, or comments are allowed in these images.

```
 _____________________________________________________________________
|                                                                     |
|   SORT     DC    C'SORT FIELDS=(1,24,CH,A,46,4,CH,D),FILES=2,'      |
|            DC    C'WORK=4 '                                         |
|   RECORD   DC    C'RECORD TYPE=F,LENGTH=80 '                        |
|                                                                     |
|_____________________________________________________________________|
Figure 63. Control Statement Images Example
```

The first two lines of Figure 63 show how a continuation line is coded. Note that there is no space between the comma and the apostrophe at the end of the first line in Figure 63.

5.8.2 User Exit Routine Branch Tables

The two branch tables pointed to by the DFSORT/VSE parameter list can contain the addresses for the preloaded user exit routines of each DFSORT/VSE phase. The first branch table specifies the phase 1 user exit routines: E11, E15, E17, and E18. The second branch table specifies the phase 3 user exit routines: E31, E32, E35, E37, E38, and E39.

Each user exit in the branch tables must be accounted for with either a branch instruction to the user exit routine or an address constant of zeros.

The user exit routines can be part of the program that invokes DFSORT/VSE. For more information on preloaded user exit routines, see Chapter 4, "Using Your Own User Exit Routines" in topic 4.0.

5.8.3 Return Code

DFSORT/VSE can pass a return code to the invoking program in the DFSORT/VSE parameter list to indicate whether the sort, merge, or copy
| application was successful or unsuccessful. If DFSORT/VSE completes
| successfully, it passes a return code of 0 or 4. If it is unsuccessful, it passes a return code of 16 or 20. The return code 20 indicates that DFSORT/VSE failed to load the message phase into partition GETVIS area.

5.8.4 Alternative Sequence

In the DFSORT/VSE parameter list, an invoking program can specify an address of the ALTSEQ control statement image or an address to an alternative sequence translate table. For more information on the ALTSEQ control statement, see Chapter 3, "Using DFSORT/VSE Program Control Statements" in topic 3.0.

If you specify an address to an alternative sequence translate table, you place in the ALTSEQ entry of the DFSORT/VSE parameter list an address to a fullword containing the four characters AQTT. In the fullword following these four characters, you place the address of your alternate sequence translate table. Figure 64 shows an example of how to pass an alternate sequence translate table address to DFSORT/VSE.

```
PLIST    DC    A(SORT or MERGE statement image)
          DC    A(RECORD statement image)
          .
          .
          DC    A(Return code field)
          DC    A(AQTT)                ALTSEQ entry
          .
          .
AQTT     DC    C'AQTT'                AQTT constant
          DC    A(AQTABLE)             address of alternate sequence
                                       translate table
          .
          .
AQTABLE  DC    X'0001020304050607'    256-byte
          DC    X'08090A0B0C0D0E0F'    user-built
                                       alternate sequence
          .                            translate table
          .
          DC    X'F8F9FAFBFCFDFEFF'
```

Figure 64. Example of Passing the Alternate Sequence Translate Table Address

---

5.8.5 Sample Coding

Figure 65 in topic 5.8.5 is an example of code that could be used to initiate execution of DFSORT/VSE. It is a sort application example with two preloaded user exit routines active.

```
          LOAD  SORT,LOADLOC                          1
          LR    15,1                                  2
          LA    1,PARAM                               3
          LA    13,SAVAREA                            4
          BALR  14,15                                 5
*
* FOLLOWING STATEMENTS ARE EXECUTED UPON COMPLETION OF SORT
*
          CLC   RETURN(2),=H'0'                       6
```

```
            BNE   SORTERR                                 7
            --
   SORTERR  DS    0H                                     8
            --
   PARAM    DC    A(SORT)                                9
            DC    A(RECORD)                             10
            DC    A(INPFIL)                             11
            DC    A(OUTFIL)                             12
            DC    A(0)                                  13
            DC    A(MODS)                               14
            DC    A(E11)                                15
            DC    A(0)                                  16
            DC    A(E31)                                17
            DC    A(RETURN)                             18
            DC    A(ALTSEQ)                             19
            DC    A(SUM)                                20
            DC    A(OMIT)                               21
            DC    A(0)                                  22
   SORT     DC    C'SORT FIELDS=(10,5,CH,A),WORK=5 '    23
   RECORD   DC    C'RECORD TYPE=F,LENGTH=80 '           24
   INPFIL   DC    C'INPFIL EXIT '                       25
   OUTFIL   DC    C'OUTFIL BLKSIZE=320,OPEN=NORWD '     26
   MODS     DC    C'MODS PH1=(,,E15),PH3=(,,E35) '      27
   SUM      DC    C'SUM FIELDS=(20,4,ZD) '              28
   ALTSEQ   DC    C'ALTSEQ CODE=(5BEA,7BEB,7CEC) '      29
   OMIT     DC    C'OMIT COND=(1,1,CH,EQ,C''*'') '      30
   SAVAREA  DS    9D                                    31
   RETURN   DC    H'0'                                  32
            LTORG                                       33
   *
   * PHASE 1 BRANCH TABLE
   *
            USING E11,15                                34
   E11      DC    A(0)                                  35
   E15      B     INPUT                                 36
   E17      DC    A(0)                                  37
   E18      DC    A(0)                                  38
   *
   * PROGRAMMER'S PHASE 1 PROCESSING ROUTINES FOLLOW
   INPUT    SAVE  (14,12)                               39
            --
            RETURN (14,12)                              40
   *
   * PHASE 3 BRANCH TABLE
   *
            USING E31,15                                41
   E31      DC    A(0)                                  42
   E32      DC    A(0)                                  43
   E35      B     OUTPUT                                44
   E37      DC    A(0)                                  45
   E38      DC    A(0)                                  46
   E39      DC    A(0)                                  47
   *
   * PROGRAMMER'S PHASE 3 PROCESSING ROUTINES FOLLOW
   OUTPUT   SAVE  (14,12)                               48
            --
            RETURN (14,12)                              49
            LTORG                                       50
   LOADLOC  DC    D'0'                                  51
            END
```

Figure 65. Example of Coding to Initiate the Execution of DFSORT/VSE

The numbers in the following list correspond to the rightmost numbers in Figure 65 in topic 5.8.5.

1
The LOAD system macro loads DFSORT/VSE module on the doubleword boundary at LOADLOC address (LOADLOC is defined by instruction 51).

2
The address of DFSORT/VSE entry point is placed in register 15.

3
Register 1 is loaded with the address of the parameter list. This list is defined below by instructions 9-22.

4
This instruction loads register 13 with the user's save area address.

5

This instruction loads register 14 with the user's return address and gives control to DFSORT/VSE. After execution of DFSORT/VSE, control returns to the next instruction.

6

Before returning control to the user, DFSORT/VSE places a return code in the halfword named RETURN. This instruction tests the return code for successful DFSORT/VSE completion.

7

If the return code is not zero, control is passed to the error processing routine SORTERR.

8

Instructions begin here to process a nonzero return code upon completion of DFSORT/VSE.

9-14

The parameter list begins at instruction 9. The first six address constants point to DFSORT/VSE control statement images that are defined below. Note that the OPTION statement image entry in the parameter list contains zeros. In this situation (OPTION control statement not supplied), DFSORT/VSE uses installation defaults.

15-17

These address constants point to the required branch tables. Instruction 16 must always contain zeros.

18

This is the address of a halfword in which DFSORT/VSE is to place a return code. The halfword is defined by instruction 32.

19-22

These four address constants point to DFSORT/VSE control statement images. One (at instruction 22) contains zero, and will be ignored; it could have been omitted. The SUM, ALTSEQ, and OMIT functions are defined in instructions 28-30.

23-30

These instructions define the control statement images. Note that, on the MODS statement, since the user exit routines are preloaded, no entries are required for the phase name and the address or length parameters.

31

This instruction defines an area in which DFSORT/VSE saves the contents of the user registers.

32

This halfword is set aside for the return code from DFSORT/VSE.

33

All literals generated by previous coding are to be collected here.

34

This instruction establishes addressability for the phase 1 branch table and the processing routines that follow the table.

35

This is the first instruction in the branch tables for phase 1, the initial sorting phase. Since no user exit routine is provided for E11 user exit, DFSORT/VSE will never give control to this instruction.

36

This instruction is a branch to the E15 user exit routine. INPUT is the label of the entry point for this routine.

37-38
E17 and E18 user exits are not used.

39
The E15 user exit routine follows the phase 1 branch table, and the first macro will save registers that are used in the routine.

40
This macro will restore registers and will return control to DFSORT/VSE.

41
This instruction establishes addressability for the phase 3 branch table and the processing routines that follow the table.

42-43
These are the first instructions of the phase 3 branch table. Since no user exit routine is provided for E31 and E32 user exits, DFSORT/VSE will never give control to this instruction.

44
This instruction is a branch to the E35 user exit routine. OUTPUT is the label of the entry point for this routine.

45-47
E37, E38 and E39 user exits are not used.

48
The E35 user exit routine follows the phase 3 branch table and the first macro will save registers that are used in the routine.

49
This macro will restore registers and will return control to DFSORT/VSE.

50
All literals generated by previous coding are to be collected here.

51
DFSORT/VSE will be loaded here on a doubleword boundary.

Note: If you use logical IOCS in your program and you assemble the logic modules with your program, DFSORT/VSE can be loaded over these modules. You must, therefore, use the linkage editor to ensure that the logic module CSECT is loaded before your program instead of after it.

For example:

```
PHASE    USERPROG,S
INCLUDE  LOGICMOD,(logmod CSECT name)
INCLUDE  USERPROG,(user CSECT)
```

For more information, see "Linkage Editor" in *VSE/ESA System Control Statements*.

6.0 Chapter 6. Using ICETOOL

Subtopics:

6.1 Overview

This chapter describes ICETOOL, a multi-purpose DFSORT/VSE utility. ICETOOL uses the capabilities of DFSORT/VSE to perform multiple operations on one or more files in a single job step. These operations include the following:

- Creating multiple copies of sorted, edited, or unedited input files
- Creating output files containing subsets of input files based on various criteria for character and numeric field values or the number of times unique values occur
- Creating output files containing different field arrangements of input files
- Printing messages 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
- Printing reports showing the DFSORT/VSE installation defaults in use
- Printing reports 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
- Printing reports 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.

Subtopics:

6.1.1 ICETOOL and DFSORT/VSE Relationship

ICETOOL is a batch front-end utility that uses the capabilities of DFSORT/VSE to perform the operations you request.

ICETOOL is comprised of thirteen 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/VSE control statements (such as SORT and INCLUDE) in the input file on SYSIPT (SYSIPT file) or in the ICETOOL parameter list to take full advantage of DFSORT/VSE's capabilities.

ICETOOL automatically calls DFSORT/VSE with the particular DFSORT/VSE control statements required for each operation.

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/VSE messages produced as a result of an ICETOOL run, they are available if you need them.

ICETOOL can be called directly or from a program. ICETOOL allows operator statements (and comments) to be supplied 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.

6.1.2 ICETOOL JCL Summary

The JCL statements used with ICETOOL are summarized below. See "Job Control Language for ICETOOL" in topic 6.2 for more detailed information. See also "JCL Restrictions" in topic 6.2.1 and "ICETOOL Notes and Restrictions" in topic 6.18.

// **DLBL infile** Defines an input disk file for a COPY, COUNT, DISPLAY, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, or VERIFY operation.

// **TLBL infile** Defines an input tape file for a COPY, COUNT, DISPLAY, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, or VERIFY operation.

// **DLBL outfile** Defines an output disk file for a COPY, SELECT, or SORT operation.

// **TLBL outfile** Defines an output tape file for a COPY, SELECT, or SORT operation.

// **ASSGN SYSnnn,cuu** Defines a printer address (cuu) for routing DFSORT/VSE messages or ICETOOL reports to a specific printer.

// **DLBL sortwk** Defines a work file for DFSORT/VSE for an OCCUR, SELECT, SORT, or UNIQUE operation.

Notes:

1. You must define all input and output files as required for ICETOOL operations.

2. Tape devices cannot be used for DFSORT/VSE work files.

3. Because ICETOOL is a DFSORT/VSE application, it uses JCL statements that are described in Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0.

6.1.3 ICETOOL Operator Summary

ICETOOL has thirteen operators which are used to perform a variety of functions. The functions of these operators are summarized below. See "ICETOOL Statements" in topic 6.3 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" in topic 6.17.1 for details.

COPY Copies one or more input files to one or more output files.

COUNT Prints a message containing the count of records in one or more input files.

DEFAULTS Prints the DFSORT/VSE installation defaults.

DEFINE Specifies input or output file characteristics for COUNT, COPY, DISPLAY, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, and VERIFY operations or where the DFSORT/VSE messages are to be routed for a group of operations.

DISPLAY Prints the values or characters of specified numeric or character fields. 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. 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 one or more input files.

SELECT Selects records from one or more input files for inclusion in an output file 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).

SORT Sorts one or more input files to one or more output files.

STATS Prints messages containing the minimum, maximum, average, and total for specified numeric fields.

UNIQUE Prints a message containing the count of unique values for a specified numeric or character field.

VERIFY Examines specified decimal fields in one or more input files and prints a message identifying each invalid value found for each field.

6.1.4 Complete ICETOOL Examples

The example below shows the JCL and control statements for a simple ICETOOL job.

```
//   // JOB EXAMP JOBA,PROGRAMMER                                          
//   // DLBL IN1,'SORT.SAMPIN',,VSAM,DISP=(OLD,KEEP)                       
//   // DLBL IN2,'SORT.BRANCH',,VSAM,DISP=(OLD,KEEP)                       
//   // DLBL OUT1,'FLY.NEW',,VSAM,DISP=(NEW,KEEP),RECORDS=500,RECSIZE=33   
//   // DLBL OUT2,'FLY.BU1',,VSAM,DISP=(NEW,KEEP),RECORDS=500,RECSIZE=33   
//   // DLBL OUT3,'FLY.BU2',,VSAM,DISP=(NEW,KEEP),RECORDS=500,RECSIZE=33   
//   // DLBL SEL1,'FLY.DUPS',,SD                                           
//   // EXTENT ,339000,,,,1900,500                                         
//   // ASSGN SYS012,SYSLST                                                
//   // EXEC ICETOOL,SIZE=128K                                             
     DEFINE NAME(IN1) TYPE(F) LENGTH(33)                                  
     DEFINE NAME(IN2) TYPE(F) LENGTH(33)                                  
   * Show installation (ILUINST) defaults                                 
     DEFAULTS LIST(LST)                                                   
   * Create three copies of a file                                        
     COPY FROM(IN1) TO(OUT1,OUT2,OUT3)                                    
   * Print a report                                                       
     DISPLAY FROM(IN2) LIST(012) DATE TITLE('Monthly Report') PAGE -      
       HEADER('Location') ON(1,15,CH) -                                   
       HEADER('Revenue') ON(22,6,PD) -                                    
       HEADER('Profit') ON(28,6,PD) -                                     
       TOTAL('Totals') AVERAGE('Averages') BLANK                          
   * Select all records with duplicate (non-unique) keys                 
     SELECT FROM(IN2) TO(SEL1) ON(1,15,CH) ALLDUPS                        
   /*                                                                     
   /&                                                                     
```

Figure 66. Simple ICETOOL Job

6.1.5 Invoking ICETOOL

ICETOOL can be invoked in the following two ways:

- Directly (that is, not from a program)
- From a program using the Parameter List Interface.

With the Parameter List Interface, your program supplies ICETOOL statements in a parameter list. ICETOOL prints messages and also puts information in the parameter list for use by your program.

6.1.6 Putting ICETOOL to Use

By using various combinations of the thirteen 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.

Subtopics:

6.1.6.1 Obtaining Various Statistics

```
 _____________________________________________________________________
|                                                                     |
|        MODE STOP                                                    |
|        DEFINE NAME(DATA1) TYPE(F) LENGTH(33)                        |
|        VERIFY FROM(DATA1) ON(22,7,PD)                               |
|        DISPLAY FROM(DATA1) LIST(012) -                              |
|          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(012)  -                               |
|          TITLE('Employees Receiving Each Salary') DATE TIME -       |
|          HEADER('Salary') HEADER('Employee Count') -                |
|          ON(22,7,PD)      ON(VALCNT)               BLANK            |
|                                                                     |
|_____________________________________________________________________|
Figure 67. Obtaining Various Statistics
```

Assume that you specify JCL statements for the following:

DATA1
A file 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.

SYS012
A printer device.

You can use the ICETOOL operators in Figure 67 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).

DEFINE
Describes the characteristics of the DATA1 file.

VERIFY
Prints error messages identifying any invalid values in the packed decimal salary field.

DISPLAY
Prints a report with each employe's name and salary and the average for all employee salaries to the SYS012 printer.

STATS
Prints messages showing the minimum, maximum, average, and total of the individual salaries.

RANGE
The three RANGE operators print messages 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 to the SYS012 printer.

6.1.6.2 Creating Multiple Versions/Combinations of Files

```
  DEFINE NAME(DATA1) TYPE(F) LENGTH(173)
  DEFINE NAME(MSTR1) TYPE(F) LENGTH(173)
* GROUP 1
  MODE CONTINUE
  COPY FROM(DATA1) TO(DATA2)
  COPY FROM(MSTR1) TO(MSTR2)
  SELECT FROM(DATA1) TO(SMALLDP) ON(30,4,CH) LOWER(10)
  UNIQUE FROM(MSTR1) ON(30,4,CH)
* GROUP 2
  MODE STOP
  COPY FROM(DATA1) TO(TEMP1) USE
  USTART
  . . . DFSORT/VSE control statements to INCLUDE
  . . . employees in department X100 and change the records to match the
  . . . format of MSTR1.
  UEND
  COPY FROM(DATA1) TO(TEMP2) USE
  USTART
  . . . DFSORT/VSE control statements to INCLUDE
  . . . employees in department X200 and change the records to match the
  . . . format of MSTR1.
  UEND
  COPY FROM(DATA1) TO(TEMP3) USE
  USTART
  . . . DFSORT/VSE control statements to INCLUDE
  . . . employees in department X300 and change the records to match the
  . . . format of MSTR1.
  UEND
  SORT FROM(MSTR1,TEMP1,TEMP2,TEMP3) TO(FINALD1,FINALD2) USE
  USTART
  . . . DFSORT/VSE control statement to sort four files
  . . . MSTR1, TEMP1, TEMP2, and TEMP3 by
  . . . department and employee name
  UEND
```

Figure 68. Creating Multiple Versions/Combinations of Files

Assume that you specify JCL statements for the indicated files:

DATA1 A file 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 file 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 SMALLDP Permanent files.

TEMP1, TEMP2, and TEMP3 Temporary files.

FINALD1, FINALD2 Permanent files.

You can use the ICETOOL operators in Figure 68 to do the following:

DEFINE
Defines characteristics of DATA1 and MSTR1 input files.

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 files.

SELECT
Creates a permanent output file containing the name, salary, department, location, and so on, of each employee in departments with less than 10 people.

UNIQUE
Prints a message 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 the output files for the employees in each department containing only name and department. The USE operand indicates that a DFSORT/VSE section is specified. See "DFSORT/VSE Section" in topic 6.3.2 for further details.

SORT
Sorts the three output files created by the COPY operators along with the master name/department file and creates two permanent files containing the resulting sorted records. The USE operand indicates that a DFSORT/VSE section is specified. See "DFSORT/VSE Section" in topic 6.3.2 for further details.

You can combine both of these examples into a single ICETOOL job step.

6.2 Job Control Language for ICETOOL

An overview of the job control language statements for ICETOOL is given below followed by discussions of each ASSGN, DLBL, or TLBL statement and the use of reserved file names and logical unit names.

```
// JOB   EXAMPL ...
// ASSGN SYS012,SYSLST
// ASSGN SYS013,01E
 .
 .
 .
// DLBL infile1
// TLBL infile2
// DLBL outfile1
// TLBL outfile2
// DLBL sortwk1
 .
 .
 .
// EXEC ICETOOL,SIZE=100K
 .
 .
 .
 ICETOOL statements and DFSORT/VSE control statements
 .
 .
 .
/*
/&
```

Figure 69. JCL Statements for ICETOOL

// **ASSGN SYSnnn,SYSLST** Defines a printer for ICETOOL reports or DFSORT/VSE messages.

// **ASSGN SYSnnn,cuu** Defines a printer address (cuu) for routing DFSORT/VSE messages or ICETOOL reports to a specific printer. (VSE/POWER statement LST can be used instead for spooling DFSORT/VSE messages or ICETOOL reports to a specific printer queue.)

Note: A printer with the specified cuu address must have been defined to the VSE/ESA operating system.

// **DLBL infile1** Defines an input disk file for an operation. Refer to Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0 for more details.

// **TLBL infile2** Defines an input tape file for an operation. Refer to Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0 for more details.

// **DLBL outfile1** Defines an output disk file for a COPY, SELECT, or SORT operation. Refer to Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0 for more details.

// **TLBL outfile2** Defines an output tape file for a COPY, SELECT, or SORT operation. Refer to Chapter 2, "Invoking DFSORT/VSE with Job Control Language" in topic 2.0 for more details.

// **DLBL sortwk1** Defines a work file for DFSORT/VSE for an OCCUR, SELECT, SORT, or UNIQUE operation.

The SYSIPT file is used for ICETOOL statements and DFSORT/VSE control statements. SYSLST is used for ICETOOL messages.

Subtopics:

6.2.1 JCL Restrictions

```
You should avoid using file names or logical unit names utilized by
DFSORT/VSE in ICETOOL operands (FROM, TO, LIST, ROUTE).  In general, you
should also avoid supplying DLBL statements with file names utilized by
DFSORT/VSE because doing so can cause unpredictable results.
Specifically, SORTWKn should not be used as a file name in ICETOOL
operators. (ICETOOL uses work files for OCCUR, SELECT, SORT, and UNIQUE
operations, if appropriate.)
```

Note: Tape devices cannot be used for DFSORT/VSE work files.

6.3 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.

Subtopics:

6.3.1 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.

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:

```
 _____________________________________________________________________
|                                                                     |
|   SORT FROM(ININ) -                                                 |
|      TO(OUTPUT1,OUTPUT2,OUTPUT3) -                                  |
|      USE                                                            |
|                                                                     |
|_____________________________________________________________________|


   Any characters specified after the dash are ignored.  Each operand
   must be completely specified on one line.
```

A control 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.

6.3.2 DFSORT/VSE Section

A DFSORT/VSE section is used to supply the DFSORT/VSE control statements for a SORT operation, and can be used to supply DFSORT/VSE control statements for a COPY or COUNT operation. The USE operand must be specified to indicate that a DFSORT/VSE section immediately follows the associated ICETOOL statement in the SYSIPT file or Parameter List.

A DFSORT/VSE section must consist of the following three parts:

- A USTART delimiter statement
- DFSORT/VSE control statements
- A UEND delimiter statement

The USTART delimiter statement indicates the start of the DFSORT/VSE section. It is required for every DFSORT/VSE section.

The UEND delimiter statement indicates the end of the DFSORT/VSE section. It is required for every DFSORT/VSE section.

A DFSORT/VSE section can contain comment and blank statements.

Subtopics:

- 6.3.2.1 DFSORT/VSE Section Restrictions

6.3.2.1 DFSORT/VSE Section Restrictions

1. DFSORT/VSE control statements in the DFSORT/VSE section must conform to the rules detailed in Chapter 3, "Using DFSORT/VSE Program Control Statements" in topic 3.0, except for the following:

   - A label field is not allowed.
   - A remark field must contain only blanks.

2. Since ICETOOL invokes DFSORT/VSE passing necessary parameters for its use, the ROUTE, FILNM, and PRINT options must not be specified in the DFSORT/VSE section.

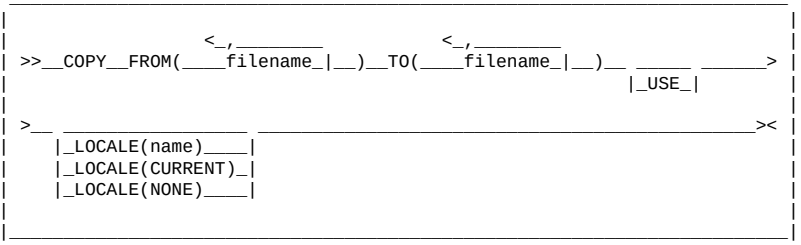
3. ICETOOL ignores the DFSORT/VSE section when SCAN mode is in effect or the USE operand is not specified.

A DFSORT/VSE section might look as shown in Figure 70. Here is an example of a SORT operator with a DFSORT/VSE section.

```
SORT FROM(IN1) TO(OUT1) USE
USTART
 INCLUDE COND=(32,3,CH,EQ,C'L92',OR,
               32,3,CH,EQ,C'J69')
 SORT FIELDS=(5,4,CH,A,22,3,PD,D)
 OPTION DSPSIZE=1
UEND
```

Figure 70. The DFSORT/VSE Section Example

6.4 COPY Operator



Copies one or more input files to one or more output files.

DFSORT/VSE is called to copy the input files to the first output file; the DFSORT/VSE section is used if USE is specified. You can use DFSORT/VSE statements to copy a subset of the input records (INCLUDE or OMIT statement, exit routines), reformat the records for output (OUTREC statement, exit routines), and so on.

If an INCLUDE or OMIT statement is specified in the DFSORT/VSE section, the active locale's collating rules affect INCLUDE and OMIT processing as

explained in the "Cultural Environment Considerations" discussion in "INCLUDE Control Statement" in topic 3.7.

If the first copy is successful, DFSORT/VSE is called as many times as necessary to copy the first output file to the second and subsequent output files (if any are specified). Therefore, for maximum efficiency, use a DASD file as the first in a list of output files on both DASD and tape. If more than one output file is specified, DFSORT/VSE must be able to read the *first* output file after it is written in order to copy it to the other output files.

The operands described below can be specified in any order.

FROM(filename,...)
Specifies the file names of the input files to be read by DFSORT/VSE for this operation. From 1 to 9 file names can be specified. JCL statements must be present and must define input files that conform to the rules for the DFSORT/VSE SORTIN files. A DEFINE operator for the first file name must be previously specified.

Refer to "JCL Restrictions" in topic 6.2.1 for more information regarding the selection of file names.

TO(filename,...)
Specifies the file names of the output files to be written by DFSORT/VSE for this operation. From 1 to 10 file names can be specified. JCL statements must be present and must define output files that conform to the rules for the DFSORT/VSE SORTOUT files.

A file name specified in the FROM operand must not be specified in the TO operand. A DEFINE operator must be supplied for tape output files or VSAM output files.

Refer to "JCL Restrictions" in topic 6.2.1 for more information regarding the selection of file names.

USE
Specifies that the DFSORT/VSE section that appears immediately after this COPY statement is to be used for this operation. For complete details, see "DFSORT/VSE Section" in topic 6.3.2.

LOCALE(name)
Specifies that locale processing is to be used and designates the name of the locale to be made active during DFSORT/VSE 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" in topic 3.13.

LOCALE(CURRENT)
Specifies that locale processing is to be used, and the current locale active when DFSORT/VSE is entered will remain the active locale during DFSORT/VSE 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" in topic 3.13.

LOCALE(NONE)
Specifies that locale processing is not to be used. 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" in topic 3.13.

Subtopics:

6.4.1 COPY Example

Subtopics:

6.4.1.1 Example 1

```
 ____________________________________________________________________
|                                                                    |
|   COPY FROM(IN1,IN2) TO(OUT1,OUT2,OUT3) USE                        |
|   USTART                                                           |
|    INCLUDE COND=(31,4,CH,EQ,C'HERR')                               |
|   UEND                                                             |
|   COPY FROM(VSAMIN) TO(VSAMOUT)                                    |
|   COPY FROM(INPUT) TO(DASD,TAPE1,TAPE2)                            |
|                                                                    |
|____________________________________________________________________|
```

The first COPY operator copies the records included from the IN1 and IN2 files to the OUT1 file and then copies the resulting OUT1 file to the OUT2 and OUT3 files. The DFSORT/VSE INCLUDE control statement is used for the first COPY operation.

The second COPY operator copies the entire VSAMIN file to the VSAMOUT file.

The third COPY operator copies the entire INPUT file to the DASD file and then copies the resulting DASD file to the TAPE1 and TAPE2 files. Since the first TO file is processed three times (written, read, read), placing the DASD file first is more efficient than placing the TAPE1 and TAPE2 files first.

6.5 COUNT Operator

```
 ____________________________________________________________________
|                                                                    |
|                  <_,________                                       |
| >>__COUNT__FROM(____filename_|__)__ _____ __ ___________________ _____>< |
|                                  |_USE_|  |_LOCALE(name)____|      |
|                                           |_LOCALE(CURRENT)_|      |
|                                           |_LOCALE(NONE)____|      |
|                                                                    |
|____________________________________________________________________|
```

Prints a message containing the count of records.

DFSORT/VSE is called to copy the input files to ICETOOL's E35 user exit. The DFSORT/VSE section is used if the USE operand is specified. You can use a DFSORT/VSE INCLUDE or OMIT control statement to count a subset of the input records.

If an INCLUDE or OMIT statement is specified in the DFSORT/VSE section, the active locale's collating rules affect INCLUDE and OMIT processing as explained in the "Cultural Environment Considerations" discussion in "INCLUDE Control Statement" in topic 3.7.

ICETOOL prints a message containing the record count as determined by its E35 user exit.

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(filename,...)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

USE
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4. Should generally be used only for a DFSORT/VSE INCLUDE or OMIT control statement.

LOCALE(name)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

LOCALE(CURRENT)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

LOCALE(NONE)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

Subtopics:

6.5.1 COUNT Example

```
 ____________________________________________________________________
|                                                                    |
|   COUNT FROM(IN1,IN2,IN3)                                          |
|   COUNT FROM(IN4) USE                                              |
|   USTART                                                           |
|    OMIT COND=(15,2,ZD,GT,32,                                       |
|         AND,28,8,PD,LT,20000)                                      |
|   UEND                                                             |
|                                                                    |
|____________________________________________________________________|
```

The first COUNT operator prints a message containing the count of records in the IN1, IN2, and IN3 files.

The second COUNT operator prints a message containing the count of records included from the IN4 file.

6.6 DEFAULTS Operator

```
 ____________________________________________________________________
|                                                                    |
| >>__DEFAULTS__ _LIST(LST)_ ____________________________________><  |
|                |_LIST(xxx)_|                                       |
|                                                                    |
|____________________________________________________________________|
```

Prints the DFSORT/VSE installation defaults.

DFSORT/VSE is shipped with a set of IBM-supplied defaults that can be modified using the ILUINST macro. The DEFAULTS operator provides an easy way to determine the installation defaults selected when DFSORT/VSE was installed. See the *Installation and Tuning* for a complete discussion of the various installation defaults and how they can be modified using the ILUINST macro.

The general format of the output can be illustrated as follows:

```
DFSORT/VSE INSTALLATION (ILUINST) DEFAULTS           - p -

 * ONLY SHOWN IF DIFFERENT FROM THE SPECIFIED INSTALLATION DEFAULT

PARAMETER      INSTALLATION DEFAULT                IBM-SUPPLIED DEFAULT *
----------     --------------------------------    --------------------------------
parameter      value                               IBM_value

  .
  .
  .
```

The value for each parameter is shown as it is set in the ILUINST macro. For any value that is different from the IBM-supplied value, the IBM-supplied value is shown to the right of it in a separate column.

The control character occupies the first byte of each record. The title and headings are always printed; p is the page number. The parameter name column occupies 10 bytes, the installation default column and the IBM-supplied default column each occupy 32 bytes, and 5 blanks appear between columns.

LIST(LST)
specifies that output will be routed to SYSLST.

LIST(xxx)
specifies that output will be routed to the printer associated with logical unit SYSxxx.

Note: ICETOOL uses CTLCHR=ASA and BLKSIZE=121 to define the file for the printer.

Subtopics:

- 6.6.1 DEFAULTS Example

6.6.1 DEFAULTS Example

```
DEFAULTS LIST(011)
```

Prints the DFSORT/VSE installation defaults to the printer associated with logical unit SYS011. The output starts on a new page and looks as follows (the parameters are shown with illustrative values):

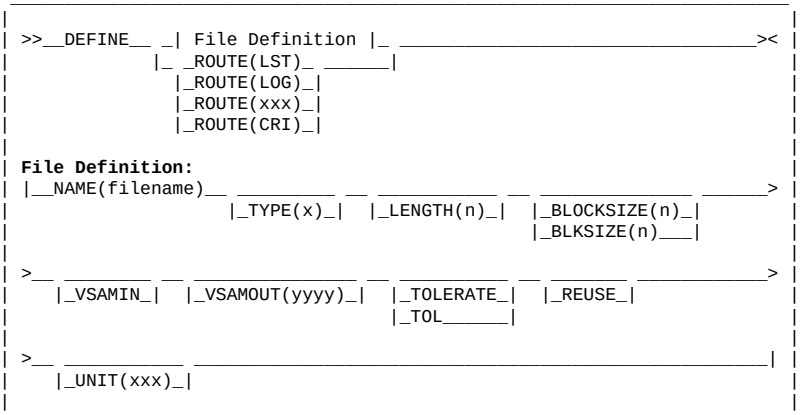
```
DFSORT/VSE INSTALLATION (ILUINST) DEFAULTS           - 1 -

 * ONLY SHOWN IF DIFFERENT FROM THE SPECIFIED INSTALLATION DEFAULT
```

```
|   PARAMETER      INSTALLATION DEFAULT                IBM-SUPPLIED DEFAULT *          |
|   ----------     --------------------------------    -------------------------------- |
|   CHALT          NOCHALT                                                              |
|   DIAG           DIAG                                NODIAG                           |
|   DUMP           NODUMP                                                               |
|   EQUALS         NOEQUALS                                                             |
|   ERASE          NOERASE                                                              |
|   VERIFY         VERIFY                              NOVERIFY                         |
|   ALTSEQ         *** SEE BELOW ***                                                    |
| | DIAGINF        NONE                                                                 |
|   DSPSIZE        0                                                                    |
|   FMS            NO                                                                   |
|   GVSIZE         256K                                0                                |
|   GVSRANY        32K                                                                  |
|   GVSRLOW        32K                                                                  |
|   LOCALE         NONE                                                                 |
| | NRECOUT        RC0                                                                  |
|   PRINT          ALL                                                                  |
|   ROUTE          LST                                                                  |
|   SORTIN         (2,3,4,5,6,7,8,9,10)                                                 |
|   SORTOUT        1                                                                    |
|   STORAGE        0                                                                    |
| | STXIT          MIN                                                                  |
| | VSAMBSP        OPTIMAL                                                              |
|   WRKSEC         YES                                                                  |
|   Y2PAST         80                                                                   |
| | ZDPRINT        NO                                                                   |
|   ILUINST ALTSEQ TABLE:  SAME AS IBM-SUPPLIED ALTSEQ TABLE BELOW                      |
|                                                                                       |
|   IBM-SUPPLIED ALTSEQ TABLE (IN HEXADECIMAL):                                         |
|          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F                             |
|        ------------------------------------------------                               |
|     00 | 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F |                            |
|     10 | 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F |                            |
|     20 | 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F |                            |
|     30 | 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F |                            |
|     40 | 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F |                            |
|     50 | 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F |                            |
|     60 | 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F |                            |
|     70 | 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F |                            |
|     80 | 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F |                            |
|     90 | 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F |                            |
|     A0 | A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF |                            |
|     B0 | B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF |                            |
|     C0 | C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF |                            |
|     D0 | D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF |                            |
|     E0 | E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF |                            |
|     F0 | F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF |                            |
|        ------------------------------------------------                               |
|                                                                                       |
|_______________________________________________________________________________________|
```

The user specified and IBM-supplied ALTSEQ tables are printed at the bottom of the installation defaults list. The title and appropriate heading lines appear at the top of each page.

6.7 DEFINE Operator



|_____|

Specifies input or output file characteristics for COPY, COUNT, DISPLAY, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, and VERIFY operations or where the DFSORT/VSE messages are to be routed for a group of operations. Up to 50 DEFINE operators with unique file names can be specified in one job step. The DEFINE statement replaces the file characteristics specified in the previous DEFINE statement with the same file name.

The DEFINE operator statement supplies DFSORT/VSE with the parameters required for the ICETOOL operation. The relationship between the operands of the DEFINE operator and the operands of the corresponding DFSORT/VSE control statement is shown in Figure 71. The DEFINE operator must precede the associated ICETOOL operator.

The operands described below can be specified in any order.

NAME(filename)
Specifies the file name of the input or output file being defined with a FROM or TO operand in the COPY, COUNT, DISPLAY, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, and VERIFY operators.

TYPE(x)
x must be one of the following:

F
indicates that the records are fixed-length records.

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

This operand must be specified for the input file.

For detailed information see the TYPE operand in the "RECORD Control Statement" in topic 3.16.

LENGTH(n)
Specifies in bytes the input record length. For variable-length records, it specifies the maximum input record length. This operand must be specified for the input file.

For detailed information see the LENGTH operand in the "RECORD Control Statement" in topic 3.16.

BLOCKSIZE(n) or BLKSIZE(n)
Specifies in bytes the maximum input block size for the input file or maximum output block size for the output file.

For detailed information see the BLKSIZE operand on the DFSORT/VSE INPFIL and OUTFIL control statements in "INPFIL Control Statement" in topic 3.8 and "OUTFIL Control Statement" in topic 3.14.

VSAMIN
Specifies that the input file is VSAM. This operand must be used for the input VSAM file.

For detailed information see the VSAM operand in the "INPFIL Control Statement" in topic 3.8.

VSAMOUT(yyyy)
Specifies that the output file is to be stored in a VSAM file. This operand must be used for the output VSAM file. yyyy must be ESDS, KSDS, or RRDS.

For detailed information see the ESDS, KSDS, or RRDS operand in the "OUTFIL Control Statement" in topic 3.14.

TOLERATE or TOL
Specifies that DFSORT/VSE should tolerate a warning code from VSAM when opening a VSAM file.

For detailed information see the TOL operand in "INPFIL Control Statement" in topic 3.8 and "OUTFIL Control Statement" in topic 3.14.

REUSE
Specifies that you want to write over an existing non-empty VSAM file defined with the REUSE attribute.

For detailed information see the REUSE operand in the "OUTFIL Control Statement" in topic 3.14.

UNIT(xxx)
Specifies the value for a logical unit number in the range 001 to 221. The UNIT operand must be used for a tape file.

For detailed information see the SORTIN or SORTOUT operand in the "OPTION Control Statement" in topic 3.13.

ROUTE(LST)
specifies that all DFSORT/VSE messages are to be routed to SYSLST. In addition, critical messages are to be routed to the system console.

ROUTE(LOG)
specifies that all DFSORT/VSE messages are to be routed to the system console.

ROUTE(xxx)
specifies that all DFSORT/VSE messages are to be routed to the SYSxxx device, where xxx can be any valid SYS number between 000 and 221.

ROUTE(CRI)
specifies that only critical DFSORT/VSE error messages are to be routed to the system console.

Note: If the DEFINE operator with the ROUTE operand is not specified, only critical DFSORT/VSE messages are issued to the system console.

Figure 71. The Relationship between the Operands of the DEFINE Operator and DFSORT/VSE Control Statements

Operands of the DEFINE operator	Operands of the DFSORT/VSE control statement			
	INPFIL	OUTFIL	RECORD	OPTION
TYPE	-	-	TYPE	-
LENGTH	-	-	LENGTH	-
BLOCKSIZE	BLKSIZE	BLKSIZE	-	-
VSAMIN	VSAM	-	-	-
VSAMOUT(ESDS)	-	ESDS	-	-
VSAMOUT(KSDS)	-	KSDS	-	-
VSAMOUT(RRDS)	-	RRDS	-	-
TOLERATE	TOL	TOL	-	-
REUSE	-	REUSE	-	-
UNIT	-	-	-	SORTIN/ SORTOUT

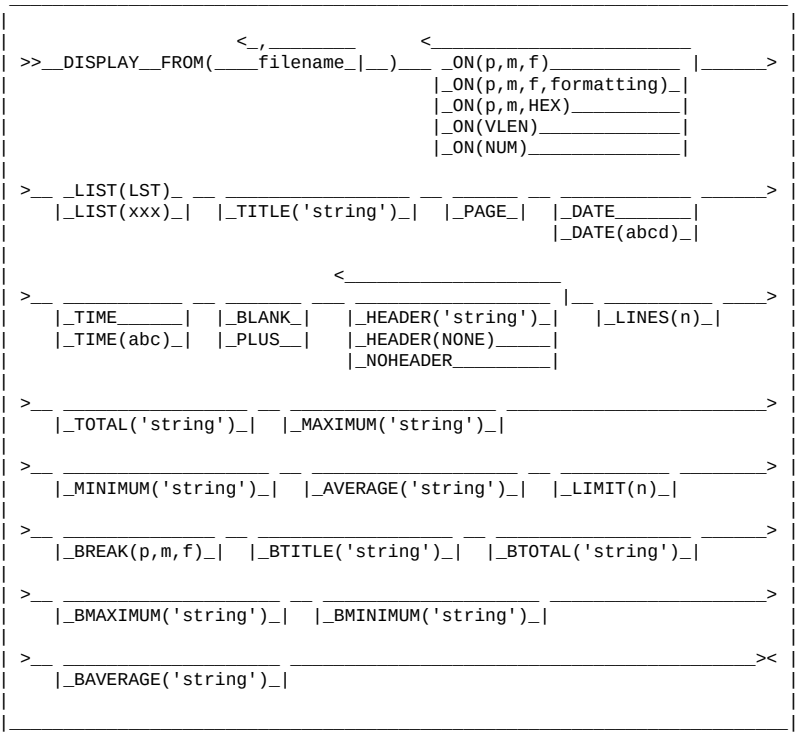
Subtopics:

6.7.1 DEFINE Example

```
 ____________________________________________________________________
|                                                                    |
|    DEFINE ROUTE(012)                                               |
|    DEFINE NAME(IN1) TYPE(F) LENGTH(173)                            |
|       COPY FROM(IN1) ...                                           |
|                                                                    |
|____________________________________________________________________|
```

- The first DEFINE operator defines that DFSORT/VSE messages are to be routed to the SYS012 printer.
- The second DEFINE operator defines characteristics of the IN1 input file for the COPY operation. The IN1 input file contains fixed-length records and the record length is 173 bytes.

6.8 DISPLAY Operator



Prints the values or characters of specified numeric or character fields. Simple, tailored, and sectioned reports can be produced. From 1 to 20 fields can be specified, but the resulting output line length must not exceed 121 bytes. The record number can be printed as a special field.

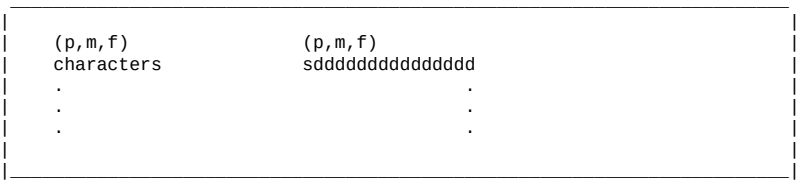
DFSORT/VSE is called to copy the input files to ICETOOL's E35 user exit. ICETOOL uses its E35 user exit to print appropriate titles, headings and data.

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.

Subtopics:

6.8.1 Simple Report

You can produce a simple report by specifying just the required operands. For example, if you specify a FROM operand and ON operands for 10-byte character and 7-byte zoned decimal fields, the output can be represented as follows:



A control character occupies the first byte of each output 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 produce only data records.

TOTAL, MAXIMUM, MINIMUM, and AVERAGE can be used to print statistics for numeric fields after the columns of data.

6.8.2 Tailored Report

You can tailor the output 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, BLANK, TITLE, PAGE, DATE, TIME, HEADER, and AVERAGE operands, and ON operands for 10-byte character and 7-byte zoned decimal fields, the output 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 output record. The title line is printed at the top of each page. 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, signs, division by 1000, 1000000 (1000×1000), 1000000000 (1000×1000×1000), 1024, 1048576 (1024×1024) or 1073741824 (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.

6.8.3 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 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                                            |
|                                                                     |
|_____________________________________________________________________|
```

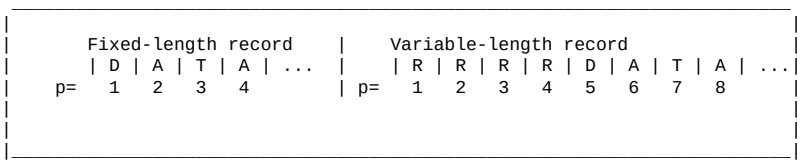
The operands described below can be specified in any order.

FROM(filename,...)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

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):



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.

Figure 72. Field Formats

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

Note: See Appendix B, "Data Format Examples" in topic B.0 for detailed format descriptions.

For a ZD or PD format field:

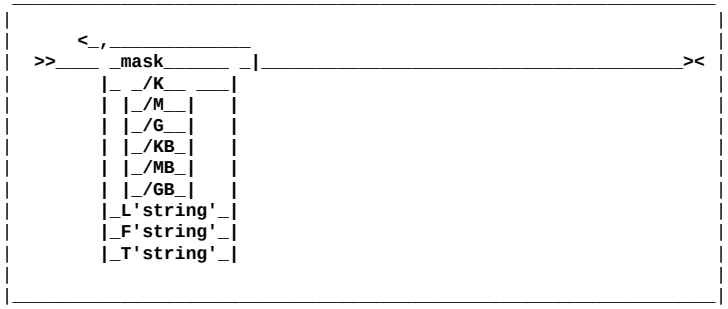
- If a decimal value contains an invalid digit (A-F), ICETOOL identifies the bad value in a message and prints asterisks for that value, and for the total, maximum, minimum and average (if specified) for that field. If the number of bad values reaches the LIMIT for invalid decimal values, ICETOOL terminates 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 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 predefined edit masks are available, encompassing many of the numeric notations throughout the world with respect to separators, decimal point, decimal places, signs, and so on. 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 are shown below.

Figure 73. 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

Figure 74. 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
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,PD,/K,A3) and as (1 234 567) with ON(1,11,PD,/K,E3).

/M

specifies division of the numeric data for this field by 1000000 (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,PD,/M,C4) and as (1.23) with ON(31,10,PD,/M,F4).

/G

specifies division of the numeric data for this field by 1000000000 (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,PD,/MB).

/GB
specifies division of the numeric data for this field by 1073741824 (1024×1024×1024) before formatting. The resulting values are rounded down to 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/VSE' is shown as '**DFSORT/VSE' with ON(1,10,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/VSE' is shown as '**DFSORT/VSE***' with ON(1,10,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 to the printer. 'RECORD

NUMBER' is used for the standard column heading (see HEADER('string'), HEADER(NONE) and NOHEADER for alternative heading options).

LIST(LST)
See the discussion of this operand on the DEFAULTS statement in "DEFAULTS Operator" in topic 6.6.

LIST(xxx)
See the discussion of this operand on the DEFAULTS statement in "DEFAULTS Operator" in topic 6.6.

TITLE('string')
Specifies printing of a title string in the title line. The title line is printed at the top of each page. 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. 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. 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. 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. 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, and seconds.

The title line is printed at the top of each page. 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 report contains only data records.

LINES(n)

Specifies the number of lines per page (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 a 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 MINIMUM line, followed by the 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 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 minimums for each numeric ON field are printed in the format (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 off 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.

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 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 off 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.

6.8.4 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.

Subtopics:

6.8.4.1 Example 1

```
DISPLAY FROM(SOURCE) LIST(LST) ON(NUM) ON(40,12,CH) -
ON(20,8,PD)
```

Prints to SYSLST:

- 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 file
  - The packed decimal values from positions 20-27 of the SOURCE file in the standard format

The 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)
   0000000000000001  SAN JOSE              000000000003745
   0000000000000002  MORGAN HILL           000000000016502
                  .  .                                  .
                  .  .                                  .
                  .  .                                  .
```

The heading line appears at the top of each page.

6.8.4.2 Example 2

```
DISPLAY FROM(IN) LIST(LST) -
      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 to SYSLST:

- 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 file
  - The zoned decimal values from positions 45-54 of the IN file
  - The zoned decimal values from positions 35-44 of the IN file
- 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 output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
National Accounting Report        - 1 -        10/21/95        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.

6.8.4.3 Example 3

```
DISPLAY FROM(DATA) LIST(LST) -
  NOHEADER -
  ON(17,5,PD) ON(1,2,FI)
```

Prints to SYSLST:

- Data lines in the standard format containing:
  - The packed decimal values from positions 17-21 of the DATA file in the standard format
  - The fixed-point values from positions 1-2 of the DATA file in the standard format

The 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
                .                   .
                .                   .
                .                   .
```

6.8.4.4 Example 4

```
COPY FROM(INPUT) TO(TEMP) USE
USTART
INCLUDE COND=(44,8,CH,EQ,C'Regular')
UEND
DISPLAY FROM(TEMP) LIST(015) -
  TITLE('Report on Regular Tools      ') PAGE -
  HEADER(NONE) ON(1,18,CH) -
  HEADER('Item') ON(35,5,CH) -
  HEADER('Percent Change') ON(28,4,PD,B1) -
  LINES(66)
COPY FROM(INPUT) TO(TEMP) USE
USTART
INCLUDE COND=(44,8,CH,EQ,C'Power')
UEND
DISPLAY FROM(TEMP) LIST(016) -
  TITLE('Report on Power Tools        ') PAGE -
  HEADER(NONE) ON(1,18,CH) -
```

```
|       HEADER('Item') ON(35,5,CH) -                                    |
|       HEADER('Percent Change') ON(28,4,PD,B1) -                       |
|       LINES(66)                                                       |
|                                                                       |
|_______________________________________________________________________|
```

This example shows how reports for different subsets of data can be produced.

The first COPY operator copies the records from the INPUT file that contain 'Regular ' in positions 44-51 to the TEMP (temporary) file.

The first DISPLAY operator uses the first subset of records in the TEMP file to print to the SYS015 printer:

- 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 packed decimal 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 file that contain 'Power ' in positions 44-51 to the TEMP (temporary) file.

The second DISPLAY operator uses the second subset of records in the TEMP file to print to the SYS016 printer:

- 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 packed decimal values from positions 28-31 formatted with one decimal place and a period as the decimal point

The SYS015 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 SYS016 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.

6.8.4.5 Example 5

```
       DISPLAY FROM(INV) LIST(017) -
         TITLE('No Frills RDW Report') -
         ON(NUM) -
         ON(VLEN) -
         ON(1,4,HEX) -
         MINIMUM('Smallest') -
         MAXIMUM('Largest')
```

Prints to the SYS017 printer:

- 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 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.

6.8.4.6 Example 6

```
       DISPLAY FROM(INV1,INV2,INV3) LIST(017) -
         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-Length File:') -
         MAXIMUM('Largest Record in Variable-Length File:')
```

Prints to the SYS017 printer:

- 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.

The output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
21.01.97          Fancy RDW Report          01:52:28 pm

Relative Record       RDW (length)  RDW (Hex)
---------------   ----------------  ---------
              1                 75  004B0000
              2                 71  00470000
              .                  .  .
              .                  .  .
              .                  .  .

Smallest Record in Variable-Length File:
                                58

Largest Record in Variable-Length File:
                                78
```

The title line and underlined heading line appear at the top of each page.

6.8.4.7 Example 7

```
    SORT FROM(PARTS) TO(TEMP) USE
    USTART
    SORT FIELDS=(15,6,CH,A)
    UEND
    DISPLAY FROM(TEMP) LIST(012) -
      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(013) -
      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(014) -
      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.

The SORT operator sorts the PARTS file to the TEMP file using the SORT statement in the DFSORT/VSE section.

The first DISPLAY operator uses the sorted records in the TEMP file to print to the SYS012 printer:

- 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 file to print to the SYS013 printer:

- 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 file to print to the SYS014 printer:

- 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 SYS012 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 SYS013 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 SYS014 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.

6.8.4.8 Example 8

```
    SORT FROM(DATA) TO(TEMP) USE
    USTART
    SORT FIELDS=(3,10,A,16,13,A),FORMAT=CH
    UEND
    DISPLAY FROM(TEMP) LIST(LST) -
      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.

The SORT operator sorts the DATA file to the TEMP file using the SORT statement in the DFSORT/VSE section.

The DISPLAY operator uses the sorted records in the TEMP file to print to the SYSLST:

- 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 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 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 output starts on a new page and looks as follows:

```
 _____________________________________________________________________
|                                                                     |
|   01/14/95        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                                     |
|                                                                     |
|_____________________________________________________________________|
```

6.8.4.9 Example 9

```
 _____________________________________________________________________
|                                                                     |
|   MODE CONTINUE                                                     |
|   VERIFY FROM(CHECK) ON(2,3,PD) LIMIT(500)                          |
|   DISPLAY FROM(CHECK) LIST(LST) 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 file. Message ILU618A is printed 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 file. Message ILU618A is printed 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 to the SYSLST:

- 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 ILU618A 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 ILU618A.
  - The hexadecimal representation of the packed decimal value in positions 2-4 (also shown in ILU618A). 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 ILU618A messages for the VERIFY operator are:

```
 _____________________________________________________________________
|                                                                     |
|   ILU618A  INVALID (2,3,PD)       VALUE - RECORD:  000000000000003,    |
|            HEX VALUE  53A54C                                        |
|   ILU618A  INVALID (2,3,PD)       VALUE - RECORD:  000000000000012,    |
|            HEX VALUE  621540                                        |
|   ILU618A  INVALID (2,3,PD)       VALUE - RECORD:  000000000000019,    |
|            HEX VALUE  400F3C                                        |
|                                                                     |
|_____________________________________________________________________|
```

The ILU618A messages for the DISPLAY operator are:

```
 _____________________________________________________________________
|                                                                     |
|   ILU618A  INVALID (2,3,PD)       VALUE - RECORD:  000000000000003,    |
|            HEX VALUE  53A54C                                        |
|   ILU618A  INVALID (2,3,PD)       VALUE - RECORD:  000000000000019,    |
|            HEX VALUE  400F3C                                        |
|                                                                     |
|_____________________________________________________________________|
```

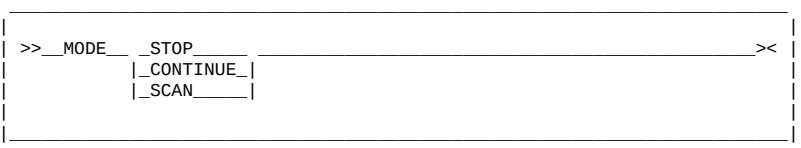
The output looks as follows:

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

The output can be used in conjunction with the ILU618A 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."

6.9 MODE Operator



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/VSE) 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/VSE.

2. Check for errors in ICETOOL statements, but do not call DFSORT/VSE.

The operands described below can be specified in any order.

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/VSE 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/VSE is not called.

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.

Subtopics:

- 6.9.1 MODE Example

6.9.1 MODE Example

```
 _____________________________________________________________________
|                                                                     |
|   MODE SCAN                                                         |
|     RANGE ...                                                       |
|     UNIQUE ...                                                      |
|   MODE STOP                                                         |
|     VERIFY ...                                                      |
|     DISPLAY ...                                                     |
|   MODE CONTINUE                                                     |
|     COPY ...                                                        |
|     SORT ...                                                        |
|     STATS ...                                                       |
|                                                                     |
|_____________________________________________________________________|
```

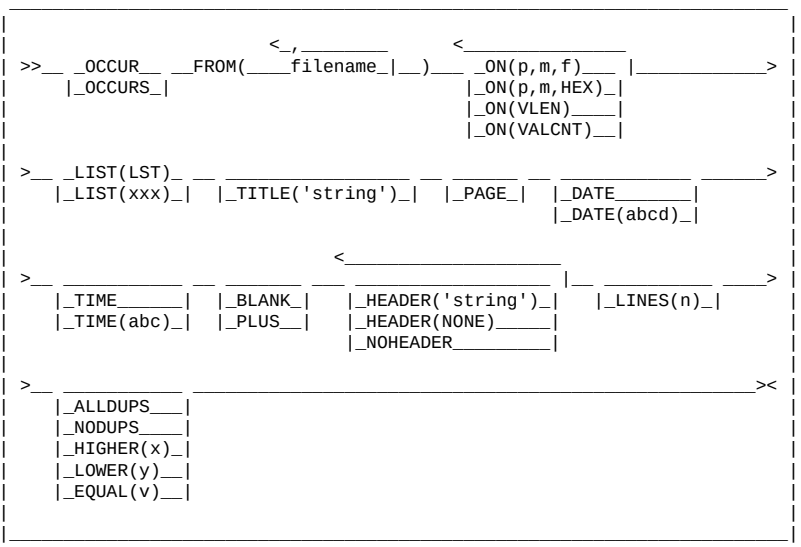
SCAN mode: RANGE and UNIQUE are checked for statement errors, but DFSORT/VSE is not called.

| 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/VSE 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 do not affect the other groups of operators.

6.10 OCCUR Operator



Prints each unique value for specified numeric or character fields and how many times it occurs. 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 output line length must not exceed 121 bytes. 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 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 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.

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/VSE is called to sort the input files to ICETOOL's E35 user exit. ICETOOL uses its E35 exit to print appropriate titles, headings and data.

Note: The OCCUR operation uses a DFSORT/VSE work file. JCL statements must be supplied for a SORTWK work file.

Subtopics:

6.10.1 Simple Report

You can produce a simple report by specifying just the required operands. For example, if you specify FROM and ON operands for 10-byte character and 7-byte zoned decimal fields and the value count, the output can be represented as follows:

```
 _____________________________________________________________________
|                                                                     |
|    (p,m,f)               (p,m,f)            VALUE COUNT             |
|    characters            sddddddddddddddd   ddddddddddddddd         |
|    .                                  .                 .           |
|    .                                  .                 .           |
|    .                                  .                 .           |
|                                                                     |
|_____________________________________________________________________|
```

A control character occupies the first byte of each output 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 output report containing only data records.

6.10.2 Tailored Report

You can tailor the output 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, 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 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 output record. The title line is printed at the top of each page. 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(filename,...)

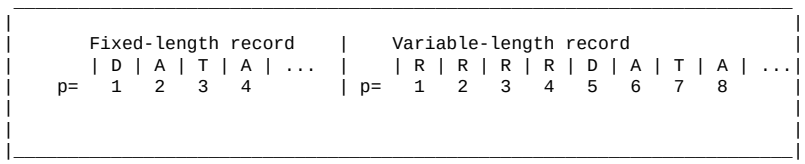
Specifies the file names of the input files to be read by DFSORT/VSE for this operation. From 1 to 9 file names can be specified. JCL statements must be present and must define input files that conform to the rules for DFSORT/VSE SORTIN files. A DEFINE operator for the first file name must be previously specified and a DEFINE operator for each input tape file name must be previously specified (see "DEFINE Operator" in topic 6.7).

Refer to "JCL Restrictions" in topic 6.2.1 for more information regarding the selection of file names.

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):



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
Note: See Appendix B, "Data Format Examples" in topic B.0 for detailed format descriptions.		

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).

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" in topic 6.8.

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(LST)
See the discussion of this operand on the DEFAULTS statement in "DEFAULTS Operator" in topic 6.6.

LIST(xxx)
See the discussion of this operand on the DEFAULTS statement in "DEFAULTS Operator" in topic 6.6.

TITLE('string')
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

PAGE
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

DATE
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

DATE(abcd)
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

TIME
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

TIME(abc)
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

BLANK
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

PLUS
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

For ON(VALCNT), PLUS is treated as BLANK.

HEADER('string')
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

HEADER(NONE)
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

NOHEADER
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

LINES(n)
See the discussion of this operand on the DISPLAY statement in "DISPLAY Operator" in topic 6.8.

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.

6.10.3 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" in topic 6.8 for additional examples of tailoring the report format.

Subtopics:

6.10.3.1 Example 1

```
 _____________________________________________________________________
|                                                                     |
|   OCCUR FROM(SOURCE,UPDATE) LIST(LST) ON(40,6,CH) ON(VALCNT)        |
|                                                                     |
|_____________________________________________________________________|
```

Prints to SYSLST:

- 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 SOURCE and UPDATE files for the unique value
  - The count of occurrences in SOURCE and UPDATE files of the unique value

The 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            0000000000000025
    ABC002            0000000000000011
    .                              .
    .                              .
    .                              .
```

The heading line appears at the top of each page.

6.10.3.2 Example 2

```
    OCCUR FROM(IN) LIST(016) -
          TITLE('   3090 Distribution   ') -
          PAGE -
          HEADER('Data Centers')  ON(VALCNT) -
          HEADER('State') ON(1,16,CH) -
          HEADER('3090s') ON(25,3,PD) -
          BLANK
```

Prints to the SYS016 printer:

- 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 file of the unique value
  - The characters from positions 1-16 of the IN file for the unique value
  - The packed decimal values from positions 25-27 of the IN file for the unique value

The 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.

6.10.3.3 Example 3

```
 _____________________________________________________________________
|                                                                     |
|    OCCURS FROM(FAILURS) LIST(LST) -                                 |
|          DATE TITLE('Possible System Intruders')  PAGE -            |
|          HEADER(' Userid ') HEADER(' Logon Failures ') -            |
|          ON(23,8,CH)      ON(VALCNT) -                              |
|          HIGHER(4) -                                                |
|          BLANK                                                      |
|                                                                     |
|_____________________________________________________________________|
```

Prints to SYSLST:

- 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 FAILURS file
  - The count of occurrences of the characters from positions 23-30 of the FAILURS file

The output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
 _____________________________________________________________________
|                                                                     |
|    09/21/95        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.

6.10.3.4 Example 4

```
OCCUR FROM(VARIN) LIST(016) -
      TITLE('Record lengths that occur only once') -
      TIME(12:) DATE(DMY.) -
      ON(VLEN) NODUPS BLANK
```

Prints to the SYS016 printer:

- 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

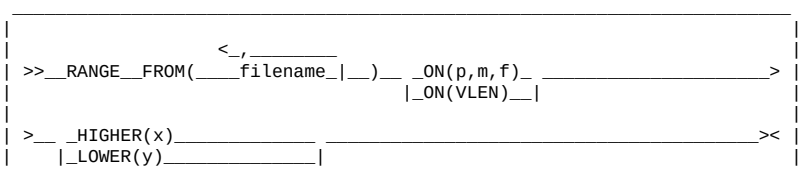
The output starts on a new page and looks as follows (the first 2 records are shown with illustrative values):

```
Record lengths that occur only once        09:52:17 am        21.09.95

RECORD LENGTH
            57
            61
             .
             .
             .
```

The title line and heading line appear at the top of each page.

6.11 RANGE Operator



```
|    |_HIGHER (x)__LOWER (y)_|                                              |
|    |_EQUAL(v)_______________|                                              |
|    |_NOTEQUAL(w)____________|                                              |
|                                                                            |
|____________________________________________________________________________|
```

Prints a message containing the count of values in a specified range for a specific numeric field.

DFSORT/VSE is called to copy the input files 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).

The operands described below can be specified in any order.

FROM(filename,...)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

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
Note: See Appendix B, "Data Format Examples" in topic B.0 for detailed format descriptions.		

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 or PD 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" in topic 6.8.

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.

Subtopics:

6.11.1 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 file 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 file that are lower than 3000.

The third RANGE operator prints a message containing the count of fixed-point values from positions 29001-29004 of the DATA3 file that are higher than -10000 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 file that are equal to -999.

The fifth RANGE operator prints a message containing the count of binary values from position 40 of the DATA3 file 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 the DFSORT/VSE section containing
OMIT COND=(40,1,CH,EQ,C'G').

6.12 SELECT Operator

```
 ______________________________________________________________________
|                                                                      |
|                    <_,________                 <_____________        |
| >>__SELECT__FROM(____filename_|__)__TO(filename)___ _ON(p,m,f)_ |____> |
|                                                  |_ON(VLEN)__|       |
|                                                                      |
| >__ _ALLDUPS___ ____________________________________________________><  |
|     |_NODUPS____|                                                   |
|     |_HIGHER(x)_|                                                   |
|     |_LOWER(y)__|                                                   |
|     |_EQUAL(v)__|                                                   |
|     |_FIRST_____|                                                   |
|     |_LAST______|                                                   |
|                                                                      |
|______________________________________________________________________|
```

Selects records from one or more input files for inclusion in an output file 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.

DFSORT/VSE is called to sort the input files to the output file. ICETOOL uses its E35 exit to determine which records to include in the output file.

ICETOOL requires extra storage for SELECT processing, over and above what is normally needed by ICETOOL and DFSORT/VSE, 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 or low 16 MB virtual). However, for a FROM file 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 megabytes of storage is needed. If ICETOOL cannot get the storage it needs, it issues a message and terminates the SELECT operation.

Note: The SELECT operation uses a DFSORT/VSE work file. JCL statements must be supplied for a SORTWK work file.

The operands described below can be specified in any order.

FROM(filename,...)

See the discussion of this operand on the OCCUR statement in "OCCUR Operator" in topic 6.10.

TO(filename)

Specifies the file name of the output file to be written by DFSORT/VSE for this operation. Output file information in JCL statements must be present and must define an output file that conforms to the rules for DFSORT/VSE's SORTOUT file.

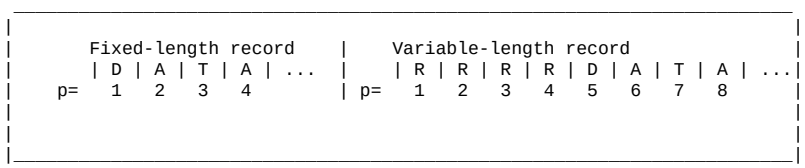
The file name specified in the FROM operand must not be the same as the file name specified in the TO operand.

Refer to "JCL Restrictions" in topic 6.2.1 for more information regarding the selection of file names.

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):



m specifies the length of the field in bytes. A field must not extend beyond position 4088, 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
Note: See Appendix B, "Data Format Examples" in topic B.0 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" in topic 6.8.

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.

Subtopics:

- 6.12.1 SELECT Examples

6.12.1 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.

Subtopics:

6.12.1.1 Example 1

```
SELECT FROM(INPUT) TO(DUPS) ON(11,8,CH) ON(30,44,CH) ALLDUPS
```

Sorts the INPUT file to the DUPS file, 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 file might look as follows (several records are shown for illustrative purposes):

```
USR002    EISSLER     012    DOC.EXAMPLES
DFSRT2    EISSLER     005    DOC.EXAMPLES
DFSRT5    MADRID      020    MYDATA
DFSRT1    MADRID      020    MYDATA
SYS003    MADRID      020    MYDATA
DFSRT2    MADRID      020    SORTST1.TEST
USR003    MADRID      020    SORTST1.TEST
.         .            .     .
.         .            .     .
.         .            .     .
```

6.12.1.2 Example 2

```
SELECT FROM(INPUT) TO(ONLYONE) ON(23,3,ZD) NODUPS
```

Sorts the INPUT file to the ONLYONE file, selecting only the records from INPUT with zoned decimal values in positions 23-25 that occur just once (that is, only records with no duplicate ON field values).

The ONLYONE file might look as follows (several records are shown for illustrative purposes):

```
DFSRT2    EISSLER     005    DOC.EXAMPLES
DFSRT1    PACKER      008    ICETOOL.SMF.RUNS
USR002    EISSLER     012    DOC.EXAMPLES
SYS003    YAEGER      032    ICETOOL.TEST.CASES
DFSRT2    MCNEILL     108    FS.TEST.CASES
.         .           .      .
.         .           .      .
.         .           .      .
```

6.12.1.3 Example 3

```
SELECT FROM(FAILURS) TO(CHECKOU) ON(28,8,CH) ON(1,5,CH) -
  HIGHER(3)
```

Sorts the FAILURS file to the CHECKOU file, selecting only the records from FAILURS 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 CHECKOU file might look as follows (several records are shown for illustrative purposes):

```
09/12/95   08:36:59        A3275647
09/12/95   09:27:32        A3275647
09/12/95   09:03:18        A3275647
09/12/95   08:56:13        A3275647
09/06/95   15:12:01        C3275647
09/06/95   14:57:00        C3275647
09/06/95   15:43:19        C3275647
09/06/95   16:06:39        C3275647
09/06/95   15:22:08        C3275647
.          .               .
.          .               .
.          .               .
```

6.12.1.4 Example 4

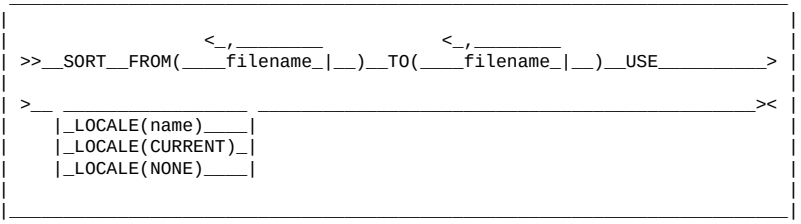
```
SELECT FROM(BOOKS,BKADD) TO(PUBLISH) ON(29,10,CH) FIRST
```

Sorts BOOKS and BKADD files to the PUBLISH file, selecting only the records from BOOKS and BKADD 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 PUBLISH file 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
.                           .          .
.                           .          .
.                           .          .
```

6.13 SORT Operator



Sorts one or more input files to one or more output files.

DFSORT/VSE is called to sort the input files to the first output file using the DFSORT/VSE section. You must supply a DFSORT/VSE SORT control statement to indicate the control fields for the sort. You can use additional DFSORT/VSE statements to sort a subset of the input records (INCLUDE or OMIT statement, exit routines), reformat the records for output (INREC or OUTREC statement, exit routines), and so on.

If an INCLUDE or OMIT statement is specified in the DFSORT/VSE section, the active locale's collating rules affect INCLUDE and OMIT processing as explained in the "Cultural Environment Considerations" discussion in "INCLUDE Control Statement" in topic 3.7.

If the first sort is successful, DFSORT/VSE is called as many times as necessary to copy the first output file to the second and subsequent output files (if any are specified). Therefore, for maximum efficiency, use a DASD file as the first in a list of output files on both DASD and tape. If more than one output file is specified, DFSORT/VSE must be able to read the *first* output file after it is written in order to copy it to the other output files.

Note: The SORT operation uses DFSORT/VSE work files. If more than one work file is needed for a SORT operation, you can specify the WORK operand in the SORT control statement. See "SORT Control Statement" in topic 3.17. JCL statements must be supplied for SORTWK work files.

The operands described below can be specified in any order.

FROM(filename,...)
See the discussion of this operand on the OCCUR statement in "OCCUR Operator" in topic 6.10.

TO(filename,...)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

USE
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

LOCALE(name)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

LOCALE(CURRENT)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

LOCALE(NONE)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

Subtopics:

6.13.1 SORT Examples

Subtopics:

6.13.1.1 Example 1

```
 _____________________________________________________________________
|                                                                     |
|         SORT FROM(IN1,IN2) TO(OUT1,OUT2,OUT3) USE                   |
|         USTART                                                      |
|          SORT FIELDS=(1,15,CH,A)                                    |
|          INCLUDE COND=(45,3,CH,EQ,C'J69')                           |
|         UEND                                                        |
|                                                                     |
|                                                                     |
|_____________________________________________________________________|
```

Sorts the IN1 and IN2 files to the OUT1 file using SORT and INCLUDE statements in the DFSORT/VSE section. Copies the resulting OUT1 file to the OUT2 and OUT3 files.

6.13.1.2 Example 2

```
 _____________________________________________________________________
|                                                                     |
|   SORT FROM(IN1) TO(FRANCE) USE LOCALE(FR_FR)                       |
|         USTART                                                      |
|          SORT FIELDS=(5,20,CH,A,31,15,CH,A,1,4,FI,D,63,10,CH,D)     |
|         UEND                                                        |
|   SORT FROM(IN1) TO(CANADA) USE LOCALE(FR_CA)                       |
|         USTART                                                      |
|          SORT FIELDS=(5,20,CH,A,31,15,CH,A,1,4,FI,D,63,10,CH,D)     |
|         UEND                                                        |
|   SORT FROM(IN1) TO(BELGIUM) USE LOCALE(FR_BE)                      |
|         USTART                                                      |
|          SORT FIELDS=(5,20,CH,A,31,15,CH,A,1,4,FI,D,63,10,CH,D)     |
|         UEND                                                        |
|                                                                     |
|_____________________________________________________________________|
```

This example shows how sorted data for three different countries can be produced.

The first SORT operator sorts all records from the IN1 file to the FRANCE file, using the SORT statement in the DFSORT/VSE section. 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 file to the CANADA file, using the SORT statement in the DFSORT/VSE section. 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 file to the BELGIUM file, using the SORT statement in the DFSORT/VSE section. The character (CH) control fields are sorted according to the collating rules defined in locale FR_BE (French language for Belgium).

6.14 STATS Operator

```
 _____________________________________________________________________
|                                                                     |
|                    <_,________      <_____________                  |
| >>__STATS__FROM(____filename_|__)___ _ON(p,m,f)_ |_________________>< |
|                                    |_ON(VLEN)__|                    |
|                                                                     |
|_____________________________________________________________________|
```

Prints messages containing the minimum, maximum, average, and total for specified numeric fields. From 1 to 10 fields can be specified.

DFSORT/VSE is called to copy the input files 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.

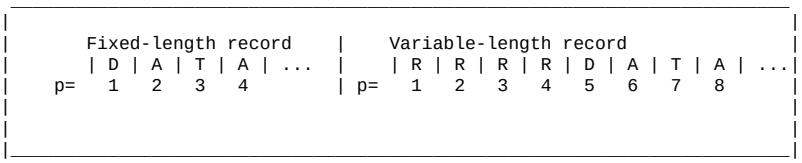
The average (or mean) is calculated by dividing the total by the record count and rounding off to the nearest integer (examples: 23 / 5 = 4, -23 / 5 = - 4).

The operands described below can be specified in any order.

FROM(filename,...)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

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):



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
Note: See Appendix B, "Data Format Examples" in topic B.0 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 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 or PD 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" in topic 6.8.

Subtopics:

- 6.14.1 STATS Example

6.14.1 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 file. 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 file.

6.15 UNIQUE Operator

```
 _____________________________________________________________________
|                                                                     |
|                    <_,________                                      |
| >>__UNIQUE__FROM(____filename_|__)__ _ON(p,m,f)_ ___________________><|
|                                   |_ON(VLEN)__|                     |
|                                                                     |
|_____________________________________________________________________|
```

Prints a message containing the count of unique values for a specified numeric or character field.

DFSORT/VSE is called to sort the input files to ICETOOL's E35 user exit. ICETOOL prints a message containing the unique count as determined by its E35 user exit.

Note: The UNIQUE operation uses a DFSORT/VSE work file. JCL statements must be supplied for a SORTWK work file.

The operands described below can be specified in any order.

FROM(filename,...)
See the discussion of this operand on the OCCUR statement in "OCCUR Operator" in topic 6.10.

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
Note: See Appendix B, "Data Format Examples" in topic B.0 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" in topic 6.8.

Subtopics:

- 6.15.1 UNIQUE Example

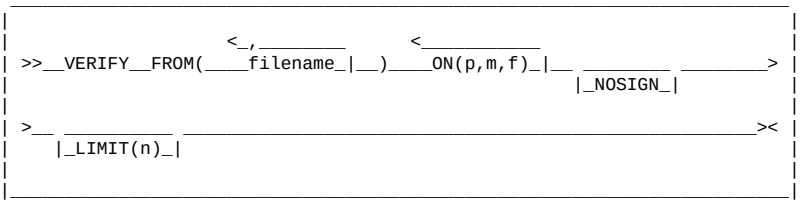
6.15.1 UNIQUE Example

```
UNIQUE FROM(DATAIN)  ON(20,40,CH)
UNIQUE FROM(DATAIN)  ON(5,3,ZD)
```

The first UNIQUE operator prints a message containing the count of unique character data in positions 20-59 of the DATAIN file.

The second UNIQUE operator prints a message containing the count of unique zoned decimal values in positions 5-7 of the DATAIN file.

6.16 VERIFY Operator



Examines particular decimal fields in one or more input files and prints a message identifying each invalid value found for each field. From 1 to 10 fields can be specified.

DFSORT/VSE is called to copy the input files 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).

Notes:

1. Values with invalid decimal digits are also identified for the DISPLAY, OCCUR, RANGE, and STATS operators.

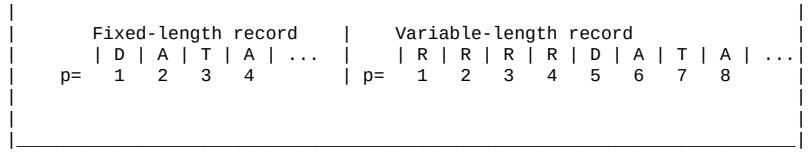
2. The DISPLAY operator can be used to print a report identifying the relative record number, hexadecimal value and associated fields for each invalid (and valid) decimal value, as shown in Example 9 under "DISPLAY Operator" in topic 6.8.

The operands described below can be specified in any order.

FROM(filename,...)
See the discussion of this operand on the COPY statement in "COPY Operator" in topic 6.4.

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):



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 B, "Data Format Examples" in topic B.0 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" in topic 6.8.

Subtopics:

- 6.16.1 VERIFY Example

6.16.1 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 file. 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 file. 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" in topic 6.8.

6.17 Calling ICETOOL from a Program

ICETOOL can be called from assembler by issuing the LOAD system macro followed by the CALL or ATTACH system macro:

- When ICETOOL is called via the CALL macro, register 1 must point to the parameter list, and register 2 must be set to 0.

  When ICETOOL finishes processing, it returns to the calling program with the return code field (in the parameter list) and register 15 set to the highest operation return code encountered. See "ICETOOL Return Codes" in topic 6.19 for an explanation of the ICETOOL return codes.

- When ICETOOL is subtasked using the ATTACH macro, the parameter list pointer must be in register 2.

  When ICETOOL finishes processing, it returns to the calling program with the return code field set to the highest operation return code encountered. See "ICETOOL Return Codes" in topic 6.19 for an explanation of the ICETOOL return codes.

Subtopics:

- 6.17.1 Parameter List Interface

6.17.1 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 and associated DFSORT/VSE statements in a parameter list. ICETOOL prints messages, and puts an operation status indicator and "operation-specific values" in the ICETOOL parameter list for use by your calling program.

Figure 75 shows the format of the parameter list used with the Parameter List Interface. Figure 76 in topic 6.17.1.1 shows the operation-specific values returned to the calling program.

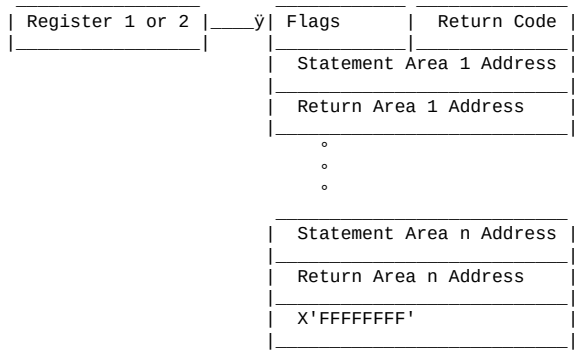


Figure 75. Parameter List for ICETOOL Parameter List Interface

The flags field and return code 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).

Subtopics:

6.17.1.1 Explanation of Fields

Flags (2 bytes)
Reserved. Must be set to zero to ensure future extendibility.

Return Code (2 bytes)
Return code field. Will be set by ICETOOL to the highest operation code encountered.

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 and the DFSORT/VSE section (if required). 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. 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" in topic 6.3. Each statement area must have **one** ICETOOL operator statement. Comment and blank statements before the operator statement are processed. Comment, blank, and additional operator statements after the end of the first ICETOOL operator statement are ignored. If the ICETOOL statement contains the USE operand, this area must contain the DFSORT/VSE section with 80-character DFSORT/VSE control statement images in the format described in "DFSORT/VSE Section" in topic 6.3.2.

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-byte packed 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.

The required return area length and the operation-specific values returned for each operator are shown in Figure 76. If the return area length is less than the length required, ICETOOL issues a message and terminates the operation.

Figure 76. 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
DEFINE	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

6.17.1.2 ICETOOL Parameter List Interface Example

The example in Figure 77 shows a portion of an assembler language program that uses the Parameter List Interface. Figure 78 shows the JCL you might use to run the program in Figure 77.

```
DEPTVIEW CSECT
         ...
* SET UP PARAMETER LIST, LOAD AND CALL ICETOOL
         LA    R1,LOADTL        LOAD ICETOOL ENTRY POINT ADDRESS
         LOAD  ICETOOL,(1)      LOAD ICETOOL PROGRAM
         LTR   R15,R15          IF LOAD WAS NOT SUCCESSFUL,
         BNZ   CKLOAD            AN ERROR MESSAGE WILL BE ISSUED
         LR    R15,R1           LOAD ICETOOL ENTRY POINT ADDRESS
         SR    R2,R2            ZERO R2 TO INDICATE CALL INVOKED
         LA    R1,PARLST        LOAD ADDRESS OF PARAMETER LIST
         BALR  R14,R15          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    RT4AVG1,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    RT4AVG2,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),RT5RCDS+4   COPY NUMBER OF DEPARTMENTS
         SP    WORK+2(4),RT5RNG+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),RT5RCDS+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 ICETOOL OUTPUT
         ...
* LOAD OPERATIONS FAILED
CKLOAD   EQU   *
* ISSUE MESSAGE: LOAD OPERATION FAILED - CHECK ICETOOL OUTPUT
         ...
* PARAMETER LIST
PARLST   DC    A(0)               FLAGS/RETURN CODE
         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    A(ST4A)            STATEMENT AREA 4 ADDRESS
         DC    A(RT4A)            RETURN AREA 4 ADDRESS
         DC    A(ST5A)            STATEMENT AREA 5 ADDRESS
         DC    A(RT5A)            RETURN AREA 5 ADDRESS
         DC    F'-1'              END OF PARAMETER LIST


* OPERATOR STATEMENT AREAS
*   DEFINE OPERATION
ST1A     DC    AL2(ST1E-ST1)      LENGTH OF STATEMENT AREA 1
ST1      DC    CL80'* DEFINE COPY ATTRIBUTES FOR IN FILE'
         DC    CL80'DEFINE NAME(IN) TYPE(F) LENGTH(53)'
ST1E     EQU   *
*   COPY OPERATION
ST2A     DC    AL2(ST2E-ST2)      LENGTH OF STATEMENT AREA 2
ST2      DC    CL80'* CREATE TWO COPIES OF THE DENVER SITE'
         DC    CL80'* DEPARTMENT RECORDS'
         DC    CL80'COPY FROM(IN) TO(OUT1,OUT2) USE'
         DC    CL80'USTART'
         DC    CL80'* SELECT ONLY THE DENVER SITE DEPARTMENT RECORDS '
         DC    CL80' INCLUDE COND=(1,12,CH,EQ,C''DENVER'') '
         DC    CL80'UEND'
ST2E     EQU   *
*   DEFINE OPERATION
ST3A     DC    AL2(ST3E-ST3)      LENGTH OF STATEMENT AREA 3
ST3      DC    CL80'* DEFINE STATS AND RANGE ATTRIBUTES FOR OUT1 FILE'
         DC    CL80'DEFINE NAME(OUT1) TYPE(F) LENGTH(53)'
ST3E     EQU   *
*   STATS OPERATION
ST4A     DC    AL2(ST4E-ST4)      LENGTH OF STATEMENT AREA 4
ST4      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)'
ST4E     EQU   *
*   RANGE OPERATION
ST5A     DC    AL2(ST5E-ST5)      LENGTH OF STATEMENT AREA 5
ST5      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)'
ST5E     EQU   *
* RETURN AREAS
*   DEFINE OPERATION
RT1A     DC    AL2(RT1E-RT1STAT)  LENGTH OF RETURN AREA 1
RT1STAT  DS    C                  OPERATION STATUS
RT1E     EQU   *
*   COPY OPERATION
RT2A     DC    AL2(RT2E-RT2STAT)  LENGTH OF RETURN AREA 2
RT2STAT  DS    C                  OPERATION STATUS
RT2E     EQU   *
*   DEFINE OPERATION
RT3A     DC    AL2(RT3E-RT3STAT)  LENGTH OF RETURN AREA 3
RT3STAT  DS    C                  OPERATION STATUS
RT3E     EQU   *
*   STATS OPERATION
RT4A     DC    AL2(RT4E-RT4STAT)  LENGTH OF RETURN AREA 4
RT4STAT  DS    C                  OPERATION STATUS
RT4RCDS  DS    PL8                COUNT OF RECORDS PROCESSED
RT4MIN1  DS    PL8                FIELD 1 - MINIMUM VALUE
RT4MAX1  DS    PL8                FIELD 1 - MAXIMUM VALUE
RT4AVG1  DS    PL8                FIELD 1 - AVERAGE VALUE
RT4TOT1  DS    PL8                FIELD 1 - TOTAL VALUE
RT4MIN2  DS    PL8                FIELD 2 - MINIMUM VALUE
RT4MAX2  DS    PL8                FIELD 2 - MAXIMUM VALUE
RT4AVG2  DS    PL8                FIELD 2 - AVERAGE VALUE
RT4TOT2  DS    PL8                FIELD 2 - TOTAL VALUE
RT4E     EQU   *


*   RANGE OPERATION
RT5A     DC    AL2(RT5E-RT5STAT)  LENGTH OF RETURN AREA 5
RT5STAT  DS    C                  OPERATION STATUS
RT5RCDS  DS    PL8                COUNT OF RECORDS PROCESSED
RT5RNG   DS    PL8                COUNT OF VALUES IN RANGE
RT5E     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
         ...
LOADTL   EQU   *                  ICETOOL ENTRY POINT
         END   DEPTVIEW
```

Figure 77. ICETOOL Parameter List Interface Example

```
// JOB EXAMP JOBA,PROGRAMMER
// LIBDEF PHASE,SEARCH=...  Sublibrary containing DEPTVIEW
// DLBL IN,'DEPT.BRANCH',,VSAM,DISP=(OLD,KEEP)
// DLBL OUT1,'DEPT.BACKUP1',,VSAM,RECORDS=200,RECSIZE=53
// DLBL OUT2,'DEPT.BACKUP1',,VSAM,RECORDS=200,RECSIZE=53
// EXEC DEPTVIEW,SIZE=128K
/*
/&
```

Figure 78. JCL for Parameter List Interface Program Example

6.18 ICETOOL Notes and Restrictions

- Since ICETOOL calls DFSORT/VSE, the installation options used for DFSORT/VSE are those specified for ILUINST. The DFSORT/VSE installation options only apply to DFSORT/VSE, not to ICETOOL. For example, ILUINST option ROUTE=LOG causes DFSORT/VSE, but not ICETOOL, to write messages to the operator console.

- When ICETOOL calls DFSORT/VSE, it passes control statements appropriate to the specific operations being performed. You should not override the DFSORT/VSE control statements passed by ICETOOL unless you understand the ramifications of doing so.

- 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/VSE issues error message and terminates.

- Each ICETOOL operation results in a set of ICETOOL messages to SYSLST output and a corresponding set of DFSORT/VSE messages to the logical or physical printer specified previously with DEFINE ROUTE(xxx) statement. If your installation supports FEF logical printer for your partition, you can route DFSORT/VSE messages to separate printer output DFSMSG1 as shown in the Figure 79:

```
* $$ JOB JNM=EXAMP
* $$ LST CLASS=E
// JOB EXAMP JOBA,PROGRAMMER
// ASSGN SYS012,FEF
* $$ LST JNM=DFSMSG1,CLASS=K,LST=SYS012
. . .
// EXEC ICETOOL,SIZE=100K
. . .
DEFINE ROUTE(012)
. . .
* report 1
DISPLAY  . . .
. . .
* report 2
DISPLAY  . . .
. . .
/*
/&
* $$ EOJ
```

Figure 79. Routing DFSORT/VSE Messages for ICETOOL Operation Reports to Separate Printer Output

For VSE/POWER operator details see *VSE/POWER Library*.

6.19 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 of one or more operations, ICETOOL passes back a return code of 12, 16, 20, or 24 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/VSE passed back a return code of 4 for one or more operations
| because NRECOUT=RC4 was in effect and DFSORT/VSE did not write any
| records to the output file.

12

Unsuccessful completion. ICETOOL detected one or more errors that prevented it from completing successfully. Messages for these errors were printed on SYSLST.

16

Unsuccessful completion. DFSORT/VSE detected one or more errors that prevented ICETOOL from completing successfully. Critical error messages for these errors were routed to the system console. If ROUTE=xxx operand was specified, all DFSORT/VSE messages were routed to SYSxxx.

20

Unsuccessful completion. ICETOOL failed to obtain 12 KB from the 24-bit GETVIS area.

24

Unsuccessful completion:. ICETOOL failed to load the phase required for ICETOOL operations. The message for this error was printed on SYSLST.

7.0 Chapter 7. Improving Efficiency

Subtopics:

7.1 Introduction

DFSORT/VSE 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/VSE performance in several ways:

- Place DFSORT/VSE modules into the SVA
- Design your applications to maximize performance:
  - Directly invoke DFSORT/VSE processing
  - Plan ahead when designing new applications
  - Specify input/output files characteristics accurately
  - Specify devices that improve elapsed time
  - Use options that improve performance
  - Avoid options that degrade performance
- | Use virtual storage efficiently
- Use work space efficiently
- Use getvis sorting
- Use dataspace sorting.

The DIAG option can be used for tuning purposes.

Installation and Tuning provides additional information related to many of the topics covered in this chapter.

7.2 Placing DFSORT/VSE Modules into the SVA

When eligible DFSORT/VSE modules are placed into the SVA, they will not be loaded into your partition program area. DFSORT/VSE should be executed | from the SVA to get maximum virtual storage utilization and performance.

7.3 Designing Your Applications to Improve Performance

Even though DFSORT/VSE automatically optimizes performance when your application is run, you can still improve efficiency by using specifications and options that permit DFSORT/VSE to make the best possible use of available resources.

Subtopics:

7.3.1 Directly Invoke DFSORT/VSE Processing

You can improve performance by invoking DFSORT/VSE directly instead of from COBOL, PL/I, RPG II with the Auto-Report feature, or Assembler. Generally, COBOL is used for convenience. However, the trade-off can be degraded performance. You can improve efficiency by taking advantage of the way DFSORT/VSE 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 or OMIT, INREC or OUTREC, and SUM. You can eliminate records from input files, reformat records to eliminate unwanted fields, and sum numeric records without requiring routines from other programs.

7.3.2 Plan Ahead when Designing New Applications

You should consider several factors when designing new applications. Whenever possible, you should:

- Use either EBCDIC character or binary control fields
- 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, hex 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.

Subtopics:

7.3.2.1 Efficient Blocking

Performance of DFSORT/VSE can be significantly improved if you block your input and output records. For large files, files on tape, or files that you sort often, you might want to choose a larger block size. In general, the larger your input and output block sizes, the better DFSORT/VSE performs.

With FBA devices, a large CI (control interval) value gives improved performance over a small one.

7.3.3 Specify Input/Output Files Characteristics Accurately

DFSORT/VSE uses all available information to perform operations efficiently. When you supply incorrect information or do not supply information such as minimum or modal record length, DFSORT/VSE makes assumptions which, if incorrect, can lead to inefficiency or termination.

Subtopics:

7.3.3.1 Variable-Length Records

When the input file consists of variable-length records, you can aid performance by careful specification of L5 in the LENGTH operand of the RECORD control statement. The L5 parameter value gives the record length that occurs most frequently in the input file (modal length). If you do not specify a value for L5, it is assumed to be equal to the average of the maximum and minimum (L2 and L4) record lengths (L2 and L4 values in the LENGTH operand of the RECORD control statement) in the input file. For optimum performance, both L4 and L5 should be specified. Specifying an inaccurate modal length can cause degraded performance.

7.3.3.2 Direct Access Storage Devices

System performance is improved if intermediate storage is allocated on cylinders boundaries. The number of tracks per cylinder for direct access devices is shown in Figure 80.

Figure 80. Number of Tracks per Cylinder for Direct-Access Devices

Device	Tracks per Cylinder
9345	15
3390	15
3380	15
3375	12

7.3.4 Specify Devices That Improve Elapsed Time

To get the best elapsed time improvement when using DASD with DFSORT/VSE, you should use 3390s for SORTINn, SORTWKn, and SORTOUT. A mixture of 3380s and 3390s for these files might not result in the same elapsed time improvement you would get with all 3390s; this is indirectly affected by DFSORT/VSE processing techniques, but is primarily due to the improved performance characteristics of the 3390 in relation to the 3380.

The exact elapsed time improvement you see when using 3390s depends on which files (SORTINn, SORTWKn, SORTOUT) reside on 3390s. We recommend that if you cannot use all 3390s, you use 3390s for SORTINn and SORTOUT files in preference to SORTWKn files.

7.3.5 Use Options That Improve Performance

To obtain optimum performance, you can fine-tune the options specified during installation and at run time. Several options that can improve performance are described below.

Subtopics:

7.3.5.1 GVSIZE

Performance can be improved for sort applications by using getvis sorting.

The GVSIZE operand of the OPTION control statement defines the maximum amount of GETVIS area that will be used for getvis sorting.

7.3.5.2 DSPSIZE

Performance can be improved for sort applications by using dataspace sorting.

The DSPSIZE operand of the OPTION control statement defines the maximum amount of data space that will be used for dataspace sorting.

Note: DFSORT/VSE assumes the following order of data space, partition GETVIS area, and partition program area usage to sort records:

- Data space if requested (DSPSIZE option is in effect) and available
- Partition GETVIS area if requested (GVSIZE option is in effect) and available
- Partition program area

7.3.5.3 INCLUDE or OMIT, STOPAFT, and SKIPREC

You can use the INCLUDE or OMIT control statement and the STOPAFT and SKIPREC options to reduce the amount of records being processed, which can reduce processor and data transfer time.

- The INCLUDE and OMIT control statements allow you to specify that only records whose fields meet certain criteria are to be accepted for sorting, merging, or copying.
- 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 files and sort or copy only the remaining records.

7.3.5.4 INREC and OUTREC

You can use the INREC or OUTREC control statement (or both) to reduce the size of the input records before they are sorted (and increase their size after they

are sorted, if necessary). This will usually result in a more efficient sort by saving data transfer time.

The INREC control statement always directs DFSORT/VSE to reformat the input records before sorting. This can provide a more efficient sort if you reduce the size of the records (or a less efficient sort if you increase the size of the records). When the INREC or OUTREC control statement is used with the INCLUDE or OMIT control statement and the STOPAFT and SKIPREC options, only the records that remain in the output file are reformatted after the process of elimination initiated by the INCLUDE or OMIT control statement and the STOPAFT and SKIPREC options.

The OUTREC control statement allows DFSORT/VSE to decide whether the input records should be reformatted before or after sorting. This decision is based on whether the INREC control statement was specified, the position of the summary fields in the reformatted records, and the size of the reformatted records compared to the input records. Since the INREC control statement cannot reformat after sorting, the OUTREC control statement can be used to lengthen the record after sorting, aligning the data fields and inserting blanks and strings to separate fields to make the output more legible.

You must also be aware of the change in record size and layout of the resulting reformatted input/output records, and of which DFSORT/VSE functions must refer to the layout of the reformatted input records, rather than the layout of the original input records.

Three types of fields can be removed by the INREC or OUTREC control statement:

- Padding fields (blanks or binary zeros) can be removed before sorting by using the INREC control statement, and then reinserted after sorting by using the OUTREC control statement.
- Fields that are not needed in the output records can be removed before sorting by using the INREC control statement or after sorting by using the OUTREC control statement. Use both INREC and OUTREC control statements if padding, repeated fields, or strings must be inserted in the output records. That is, use the INREC control statement to remove unnecessary fields before sorting, and the OUTREC control statement to repeat fields and insert padding and strings after sorting.
- If input files consist of variable-length records that are being sorted and only the fixed part of the records is needed, the variable part can be eliminated before sorting. This can result in a significantly more efficient sort.

The object of using the INREC or OUTREC control statement is to reduce the amount of data to be processed. The INREC control statement is best used to shorten the record (by eliminating fields), and the OUTREC control statement, to lengthen the record (by repeating or realigning fields, padding with blanks and inserting strings).

Using the INREC or OUTREC control statement can degrade performance if the number of bytes eliminated from the input records before sorting is less than the total length of the control fields for the sort application.

7.3.5.5 LOCALE

You can use the LOCALE option to sort and merge character data based on collating rules in an active locale; this enables you to obtain results with DFSORT/VSE that were previously possible only through preprocessing or postprocessing (or both) of your data. By eliminating such costly processing, you can save time and processing resources.

7.3.5.6 SUM

You can improve performance by using a SUM control statement to add the contents of fields. The SUM control statement adds the contents of specified summary fields in records with identical control fields (duplicate keys). The result is placed in one record while the other record is deleted, thus reducing the number of records to be output by DFSORT/VSE. The SUM control statement is processed after SKIPREC option and after INCLUDE or OMIT, or INREC control statement.

Eliminating Records with Duplicate Keys: You can eliminate records with duplicate keys without the summing by specifying

```
SUM FIELDS=NONE
```

when using the SUM control statement.

7.3.5.7 ICETOOL

ICETOOL is a multi-purpose utility that allows you to use DFSORT/VSE's highly efficient I/O and processing to perform multiple operations on one or more files in a single job step. ICETOOL's thirteen operators allow you to perform sort, copy, statistical, and report operations quickly and efficiently.

7.3.5.8 ADDROUT or ADDROUT=D

You can use the ADDROUT or ADDROUT=D operand of the OPTION control statement to reduce the size of the input records before they are sorted, because the final sort output is the direct access address of the input record (followed, in the case of ADDROUT=D, by the control fields of the input record). This will usually result in a more efficient sort application by saving data transfer time.

Use of the ADDROUT or ADDROUT=D operand of the OPTION control statement can degrade performance if the total length of the control fields is as large as the total length of the input record; in this case, the output record would be larger than the input record. When using ADDROUT=D, try to remove as many control fields as possible.

7.3.5.9 NOCHAIN

If the input file is on tape, is declared as fixed format, and contains many blocks which are shorter than the specified maximum block size, you will incur the overhead of chaining without its benefits because fixed-format tape input command chains cannot read past a short input block. In this case (and only in this case), specify the NOCHAIN operand of the INPFIL control statement to prevent degraded performance. However, if possible, it is better to alter the BLKSIZE operand of the INPFIL control statement to be accurate and try to avoid short blocks in the input file.

7.3.6 Avoid Options That Degrade Performance

Certain options can adversely affect performance, and should be used only when necessary.

Subtopics:

7.3.6.1 BYPASS

The BYPASS operand of the INPFIL control statement precludes the use of command chaining when reading input files; this can degrade performance.

7.3.6.2 CKPT

| If DFSORT/VSE is program invoked or if user exit routines are in use, all
| virtual storage allocated to the partition in which DFSORT/VSE is running
| is checkpointed. Otherwise, only the virtual storage being used by DFSORT/VSE is checkpointed. Checkpointing the whole partition can take
| longer than checkpointing the virtual storage used by DFSORT/VSE; DFSORT/VSE performance can, therefore, be degraded.

7.3.6.3 ERASE

The ERASE operand of the OPTION control statement involves additional writes on the work files which can degrade performance.

7.3.6.4 EQUALS

The EQUALS operand of the SORT control statement causes an additional field to be added to each record, which increases the time needed for comparison of records and for data transfer. DFSORT/VSE performance can, therefore, be degraded.

7.3.6.5 LOCALE

The LOCALE option may increase the time required to run an application.

7.3.6.6 VERIFY

The VERIFY operand of the OPTION control statement will degrade performance because each output file block will be checked to ensure it was written correctly. It also precludes the use of command chaining when writing output files which can degrade performance.

7.3.6.7 WORK=0

DFSORT/VSE does not use command chaining for input or output files when work files are not provided (WORK=0 operand of the SORT control statement), unless getvis or dataspace sorting is used.

7.3.6.8 User Exit Routines

When user exit routines are included in an application, the time required to run the application usually increases.

The run time required by most user exit routines is generally small, but the routines at E15, E32, and E35 user exits are entered for each record of the files. For large input files, the total run time of these routines can be relatively large.

| User exit routines also occupy virtual storage that could otherwise be
| used by DFSORT/VSE to improve its performance. Depriving DFSORT/VSE of
| this virtual storage particularly degrades performance when DFSORT/VSE is running in a small partition or when the size of the input files is very large.

| 7.4 Using Virtual Storage Efficiently

| In general, making more virtual storage available to DFSORT/VSE (up to a
| certain limit), will improve performance. The default amount of vitrual storage that will be made available to DFSORT/VSE is defined when it is installed.

| DFSORT/VSE generally uses all the virtual storage available to it.
| Figure 81 shows how various parameters affect DFSORT/VSE use of virtual
| storage.

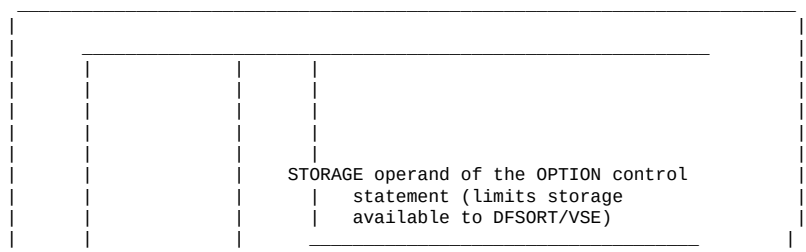
| The amount of virtual storage of the partition program area actually used by DFSORT/VSE is the smaller of:

SIZE
: from the EXEC or SIZE job control statement

and

STORAGE
: from the OPTION control statement, or the installed default.

The use of the STORAGE operand is a good way to avoid overcommitment. If neither STORAGE nor SIZE is specified, DFSORT/VSE has access to the entire partition program area. A storage allocation map is shown in Figure 81.



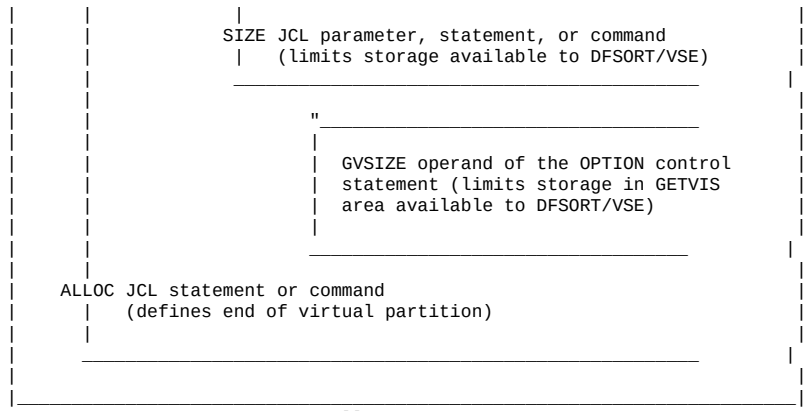


Figure 81. DFSORT/VSE Storage Allocation Map

| DFSORT/VSE needs a minimum of 32 KB of virtual storage of the partition program area.

The minimum requirement for a given application can be more than 32 KB.

| For improved performance, it is recommended that you use significantly
| more than the minimum amount of virtual storage of the partition program area even when getvis or dataspace sorting is in effect.

| You will generally need more virtual storage if you use:

- Large input files
- Spanned records
- ALTSEQ control statement or CHALT operand of the OPTION control statement
- INCLUDE, OMIT, SUM, OUTREC, or INREC control statements (although the
  | INREC control statement can also reduce virtual storage requirements
  | by shortening record sizes)
- Very large blocks or logical records

In general, when you vary the amount of storage available to DFSORT/VSE, several things occur:

1. | If you increase the amount of virtual storage:
   - SIOs are reduced.
   - For larger input files, processor time generally decreases; that
     | is, overhead in managing the extra virtual storage is offset by DFSORT/VSE having to make fewer passes over the data.
   - For larger input files, elapsed time generally decreases.

- For a very heavily loaded system, elapsed time might increase because overall system paging activity increased.

- For very small input files, processor time might remain stable or
  | increase because of the overhead in managing the extra virtual
  | storage. For larger input files, processor time will usually
  | decrease because the overhead in managing the extra virtual
  | storage would be less than the benefit gained by DFSORT/VSE making fewer passes over the data.

2. | If you decrease the amount of virtual storage:

   - | SIOs are increased.

   - Elapsed time increases for most DFSORT/VSE applications.

   - Processor time decreases for very small input files, but increases for larger input files.

7.5 Using Work Space Efficiently

Performance is improved 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 DFSORT/VSE.

DFSORT/VSE performance is improved if you use devices and channels efficiently.

You can get improved performance using direct access intermediate storage devices when you:

- Use two or more work files.
- Select the device with the fastest data transfer rate (for example, use 3390 devices attached to 3990-3 Storage Control).
- Allocate space on cylinder boundary for CKD devices.
- Assign one work file per actuator.
- Use devices that are of the same type.
- Use multiple channel paths to devices.
- Make all work files the same size, or as nearly the same size as possible.
- Make sure the SORTWKn files do not share devices or channels with the SORTINn and SORTOUT files.
- Specify enough space for each work file to make sure that automatic secondary allocation is not needed.

Elapsed time is decreased when DFSORT/VSE can both read input while writing to SORTWKn and write output while reading from SORTWKn. If, for example, you have two channels, the best allocation of them is to have SORTINn and SORTOUT on one and SORTWKn on the other.

Work space requirements can be estimated by using the guidelines found in Appendix A, "Estimating Storage Requirements" in topic A.0.

7.6 Using Getvis Sorting

Getvis sorting uses the GETVIS area to improve performance of DFSORT/VSE applications.

Getvis sorting allows DFSORT/VSE to sort large pieces of data without work files. It helps to reduce CPU time and elapsed time.

The amount of the available GETVIS area used for getvis sorting can be controlled with the GVSIZE operand of the OPTION control statement.

Therefore, getvis sorting with the GVSIZE=MAX option should be used for achieving the best performance of DFSORT/VSE sort applications as follows:

- If incore getvis sorting is possible (that is all records of input files can be held in the available GETVIS area) then the WORK=0 option of the SORT control statement can also be specified to eliminate work buffer allocation. Thus, the available GETVIS area will be used to
  | hold all records of input files and the available virtual storage (partition program area) will be used for input and output buffer allocation.

- If incore getvis sorting is not possible (that is all records of input files cannot be held in the available GETVIS area) or it is unknown whether it is possible, then WORK=n option of the SORT control statement must be specified for work space usage. Thus, the available
  | GETVIS area will be used to hold records of input files and the
  | available virtual storage (partition program area) will be used for input, output, and work buffer allocation.

7.7 Using Dataspace Sorting

Dataspace sorting uses data space available with VSE/ESA (ESA supervisor mode) to improve performance of DFSORT/VSE applications.

Dataspace sorting allows DFSORT/VSE to sort large pieces of data without work files. It helps to reduce CPU time and elapsed time.

The amount of data space used for dataspace sorting can be controlled with the DSPSIZE operand of the OPTION control statement.

Notes:

1. For incore dataspace sorting, WORK=0 can be used to eliminate work buffer allocation.

2. Dataspace sorting with the DSPSIZE=MAX option should be used for achieving the best performance because of the dynamic control of storage with dataspace sorting.

3. For non-incore dataspace sorting, WORK=n can be used to to specify work space usage with dataspace sorting.

4. When dataspace sorting is in effect, input file records are held in data space and the partition program area holds the input, output, and for a non-incore sort, work buffers.

7.8 Using the DIAG Option

The DIAG operand of the OPTION control statement can be used for tuning purposes, to investigate how well DFSORT/VSE is performing in its current environment, and to discover whether and how improvements could be made.

| If you specify the DIAG operand, you receive extra messages about
| DFSORT/VSE virtual and intermediate storage use, optimization parameters, data handling, and time required.

Installation and Tuning provides additional information related to tuning.

8.0 Chapter 8. Examples of DFSORT/VSE Applications

Subtopics:

8.1 Summary of Examples

The table below summarizes the examples provided in this chapter.

No.	Application	Input	Output	Functions/Options
1	Sort	Tape	Tape	PRINT, LABEL, ROUTE, CLOSE, E31, E37
2	Sort	Tape	Tape	PRINT, LABEL, CLOSE
3	Sort	Disk	Tape	PRINT, LABEL, ADDROUT, STORAGE, OPEN, CLOSE, E31, E37
4	Sort	Disk	Disk	VSAM, ESDS, ROUTE, ADDROUT, DUMP, FILNM
5	Sort	Tape	Tape	LABEL, BYPASS, INCLUDE, OUTREC
6	Sort	Tape	Tape	LABEL, BYPASS, SUM, ALTSEQ
7	Sort	Disk	Disk	SAM ESDS
8	Sort	Disk	Disk	VSAM, work file secondary allocation
9	Sort	Disk	Disk	VSAM, work files, LOCALE, GVSIZE, DIAG, INCLUDE
10	Sort	Disk	Print/Pu	cOutput to SYSLST and SYSPCH
| 11	| Sort	| Disk	| Disk	| Year 2000 features with COBOL
12	Merge	Tape/Dis	Tape	LABEL
13	Merge	Disk	Disk	EQUALS/NOEQUALS
14	Copy	Tape	Tape	PRINT, LABEL, ROUTE, CLOSE
15	Copy	Disk	Disk	Multiple input
16	ICETOOL	Disk	Disk	STATS, SORT, RANGE, DISPLAY, OCCUR, SELECT

8.2 Sort Examples

This section includes eleven sort examples.

Example 1. Sort with Tape Input and Output

```
// JOB SAMPLE01                                                    01
// ASSGN   SYS001,X'283'      SORT OUTPUT                          02
// ASSGN   SYS002,X'282'      SORT INPUT                           03
// ASSGN   SYS003,X'162'      SORT WORK 1                          04
// ASSGN   SYS004,X'162'      SORT WORK 2                          05
// ASSGN   SYS005,X'163'      SORT WORK 3                          06
// ASSGN   SYS006,X'163'      SORT WORK 4                          07
// TLBL    SORTOUT                                                 08
// DLBL    SORTWK1,,0                                              09
// EXTENT  SYS003,,1,0,760,38                                      10
// DLBL    SORTWK2,,0                                              11
// EXTENT  SYS004,,1,0,380,38                                      12
// DLBL    SORTWK3,,0                                              13
// EXTENT  SYS005,,1,0,380,38                                      14
// DLBL    SORTWK4,,0                                              15
// EXTENT  SYS006,,1,0,760,38                                      16
// EXEC    SORT,SIZE=48K                                           17
  OPTION PRINT=ALL,LABEL=(N,U),ROUTE=LST                          18
  SORT FIELDS=(5,4,CH,D,20,12,BI,A,50,3,CH,A),                    19
    WORK=4,FILES=1                                                20
  RECORD TYPE=V,LENGTH=(158,,,54,100)                             21
  INPFIL BLKSIZE=1544,CLOSE=RWD                                   22
  OUTFIL BLKSIZE=1544,CLOSE=RWD                                   23
  MODS PH3=(SAEXIT,L2000,E31,E37)                                 24
/*                                                                25
/&                                                                26
```

Line Explanation

01
JOB statement. Indicates the beginning of control information for the SAMPLE01 job.

02-07
ASSGN statements. Assign devices to be used as output, input, and work units. The output and input units, SYS001 and SYS002, respectively, are assigned to tape units at addresses 283 and 282. Four work units, SYS003 through SYS006, are assigned to disk.

08
TLBL statement. Defines a tape file for output.

09-16
DLBL and EXTENT statements. Define four extents on the two disk work volumes; 152 tracks are allocated in all.

17
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 48 KB of virtual storage directly available

for its use.

18
OPTION control statement. Specifies that all DFSORT/VSE messages are to be written. The output file has nonstandard labels. The input file is unlabeled. All DFSORT/VSE messages are to be routed to SYSLST.

19-20
SORT control statement. Specifies that a sort based on three control fields will be executed. There are four work files available and the input file is contained on one file.

21
RECORD control statement. Specifies that variable-length records are to be sorted. The maximum input record length is 158 bytes, as indicated by the L1 value of the LENGTH operand. By default, the values for L2 and L3 are the same as L1. The L4 value indicates that the minimum input record length is 54, and the most common (modal) record length is 100 bytes, as indicated by the L5 parameter.

22
INPFIL control statement. Specifies that the input block size is 1544 bytes, and DFSORT/VSE is to rewind the input volume at end of file.

23
OUTFIL control statement. Specifies that the output block size is 1544 bytes, and DFSORT/VSE is to rewind the output volume at end of application.

24
MODS control statement. Specifies that DFSORT/VSE is to load the SAEXIT user routine for execution during phase 3. This routine has a length of 2000 bytes and is relocatable. The 2000 bytes are to be included within the virtual storage size that DFSORT/VSE is to operate in. The user exits to be activated during phase 3 are E31 and E37, which are used for label processing of the output file.

25-26
End-of-data file and end-of-job statements.

Example 2. Sort with Tape Input and Output and a multiextent work file

```
// JOB SAMPLE02                                                  01
// ASSGN   SYS001,X'284'      SORT OUTPUT                        02
// ASSGN   SYS002,X'282'      SORT INPUT                         03
// ASSGN   SYS003,X'191'      SORT WORK                          04
// ASSGN   SYS004,X'192'      SORT WORK                          05
// DLBL    SORTWK1,,1,DA                                         06
// EXTENT  SYS003,191191,1,0,760,38                              07
// EXTENT  SYS003,191191,1,1,380,38                              08
// EXTENT  SYS004,192192,1,2,380,38                              09
// EXTENT  SYS004,192192,1,3,760,38                              10
// EXEC    SORT,SIZE=48K                                         11
  OPTION PRINT=ALL,LABEL=(U,U)                                  12
  SORT FIELDS=(5,4,CH,D,20,12,BI,A,50,3,CH,A),                  13
    FILES=1                                                     14
  RECORD TYPE=V,LENGTH=(158,,,54,100)                           15
  INPFIL BLKSIZE=1544,CLOSE=RWD                                 16
  OUTFIL BLKSIZE=1544,CLOSE=RWD                                 17
/*                                                              18
/&                                                              19
```

This example is the same as Example 1, but illustrates the use of a multiextent work file.

Line Explanation

01
JOB statement. Indicates the beginning of control information for the SAMPLE02 job.

02-05
ASSGN statements. Assign devices to be used as output, input, and work units. The output and input units, SYS001 and SYS002, respectively, are assigned to tape units at addresses 284 and 282. Two work units, SYS003 and SYS004, are assigned to disk.

06
DLBL statement. Defines the SORTWK1 work file. Code DA is specified, indicating that there is one multiextent work file, and a retention period of one day is requested.

07-10
EXTENT statements. Define the four work extents.

11
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 48 KB of virtual storage directly available for its use.

12
OPTION control statement. Specifies that all DFSORT/VSE messages are to be written. Both the output file and input file are unlabeled.

13-14
SORT control statement. Specifies that a sort based on three control fields will be executed. The input file is contained on one file. Since WORK is not specified, the default, WORK=DA, applies.

15
RECORD control statement. Specifies that variable-length records are to be sorted. The maximum input record length is 158 bytes, as indicated by the L1 value of the LENGTH operand. By default, the values for L2 and L3 are the same as L1. The L4 value indicates that the minimum record length is 54 bytes. The L5 value indicates that the most common (modal) record length is 100 bytes.

16
INPFIL control statement. Specifies that the input block size is 1544 bytes, and DFSORT/VSE is to rewind the input volume at end of file.

17
OUTFIL control statement. Specifies that the output block size is 1544 bytes, and DFSORT/VSE is to rewind the output volume at end of application.

18-19
End-of-data file and end-of-job statements.

Example 3. Sort with Input on Disk, Tape Output, and ADDROUT

```
|                                                                    |
|   // JOB SAMPLE03                                            01    |
|   // ASSGN   SYS001,X'284'      SORT OUTPUT                  02    |
|   // ASSGN   SYS002,X'190'      SORT INPUT                   03    |
|   // ASSGN   SYS003,X'191'      SORT WORK 1                  04    |
```

```
|   // ASSGN   SYS004,X'192'      SORT WORK 2                      05    |
|   // ASSGN   SYS005,X'192'      SORT WORK 3                      06    |
|   // ASSGN   SYS006,X'191'      SORT WORK 4                      07    |
|   // ASSGN   SYS007,X'191'      SORT WORK 5                      08    |
|   // TLBL    SORTOUT                                             09    |
|   // DLBL    SORTIN1                                             10    |
|   // EXTENT  SYS002,190190,,,1900,500                            11    |
|   // DLBL    SORTWK1                                             12    |
|   // EXTENT  SYS003,191191,,,3800,10                             13    |
|   // DLBL    SORTWK2                                             14    |
|   // EXTENT  SYS004,192192,,,3838,12                             15    |
|   // DLBL    SORTWK3                                             16    |
|   // EXTENT  SYS005,192192,,,5700,700                            17    |
|   // DLBL    SORTWK4                                             18    |
|   // EXTENT  SYS006,191191,,,380,99                              19    |
|   // DLBL    SORTWK5                                             20    |
|   // EXTENT  SYS007,191191,,,760,99                              21    |
|   // EXEC    SORT                                                22    |
|      OPTION PRINT=ALL,LABEL=(N),ADDROUT,STORAGE=64K              23    |
|      SORT FORMAT=FL,FIELDS=(86,19,A,106,8,D,415,08,D,391,33,A),  24    |
|        WORK=5                                                    25    |
|      RECORD TYPE=V,LENGTH=(1641,,,547,900)                       26    |
|      OUTFIL BLKSIZE=80,OPEN=NORWD,CLOSE=RWD                      27    |
|      MODS PH3=(PHASE3A,L19000,E31,E37)                           28    |
|   /*                                                             29    |
|   /&                                                             30    |
|                                                                        |
|________________________________________________________________________|
```

Line Explanation

01

JOB statement. Indicates the beginning of control information for the SAMPLE03 job.

02-08

ASSGN statements. Assign devices to be used as output, input, and work units. The output unit, SYS001, is assigned to a tape unit at address 284. The input unit, SYS002, is assigned to a disk unit at address 190. The work units, SYS003 through SYS007, are assigned to disk units at addresses 191 and 192.

09

TLBL statement. Defines the tape label information for the output file.

10-21

DLBL and EXTENT statements. Define six extents: one extent containing 500 tracks for input, and five extents containing a total of 920 tracks on three volumes for work space.

22

EXEC statement. Calls DFSORT/VSE directly by its name, SORT.

23

OPTION control statement. Specifies that all DFSORT/VSE messages are to be written. The output file will have nonstandard labels. The input file has standard labels (by default). Each output record will contain the direct access address of its corresponding sorted input record (a 10-byte disk address). 65536 bytes of virtual storage (partition program area) is reserved for use by DFSORT/VSE, if available.

24-25

SORT control statement. Specifies that a sort based on four control fields, all containing floating point data, will be executed. There are five work files available and the input file is contained on one file (by default).

26

RECORD control statement. Specifies that variable-length records are to be sorted. The maximum input record length is 1641 bytes, as indicated by the L1 value of the LENGTH operand. Since OPTION ADDROUT was specified, the default for L2 is 78 (10+the sum of the control field lengths for SAM input files), and the default for L3 is 10 (for SAM input files). The L4 value indicates that the minimum input record length is 547 bytes, and the most common (modal) record length is 900 bytes, as indicated by the L5 parameter.

27

OUTFIL control statement. Specifies that the output block size is 80 bytes. DFSORT/VSE is not to rewind the output volume before opening the volume; it will rewind the output volume at end of application. As there is no INPFIL statement, the input block size is 1645 bytes by default.

28

MODS control statement. Specifies that DFSORT/VSE is to load the PHASE3A user routine for execution during phase 3. This routine has a length of 19000 bytes and is relocatable. The 19000 bytes are to be included within the storage size that DFSORT/VSE is to operate in. The user exits to be activated are E31 and E37. They are needed because the output file will have nonstandard labels (see statement 23).

29-30

End-of-data file and end-of-job statements.

Example 4. Sort with VSAM Input and Output and ADDROUT

```
// JOB SAMPLE04                                                  01
// ASSGN   SYS001,X'160'  SORT OUTPUT                           02
// ASSGN   SYS003,X'163'  SORT WORK                             03
// ASSGN   SYS006,X'160'  SORT INPUT                            04
// DLBL    INPUT,'NAME.DEFINED.BY.AMS',,VSAM                    05
// EXTENT  SYS006,DISK01                                        06
// DLBL    SORTWK1,,0                                           07
// EXTENT  SYS003,,,,150,6                                      08
// DLBL    SORTOUT,'ALSO.DEFINED.BY.AMS',0,VSAM                 09
// EXTENT  SYS001,DISK01                                        10
// EXEC    SORT,SIZE=48K                                        11
   OPTION ROUTE=LST,DUMP,ADDROUT,FILNM=(,INPUT)                 12
   SORT FIELDS=(1,56,BI,A),WORK=1                               13
   RECORD TYPE=F,LENGTH=(80,,5)                                 14
   INPFIL VSAM                                                  15
   OUTFIL ESDS                                                  16
/*                                                              17
/&                                                              18
```

Line Explanation

01

JOB statement. Indicates the beginning of control information for the SAMPLE04 job.

02-04

ASSGN statements. Assign devices for input, output, and work files. All files are on disk devices.

05-06

DLBL and EXTENT statements. Define a VSAM file as input. This file was defined by IDCAMS and previously loaded by another program.

07-08

DLBL and EXTENT statements. Define a work file on a disk; only 6 tracks are required.

09-10

DLBL and EXTENT statements. Define a VSAM file as output. The file must have been created previously.

11

EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 48 KB of virtual storage directly available for its use. There must be sufficient storage left in the partition for VSAM use.

12

OPTION control statement. Specifies that all DFSORT/VSE messages are to be routed to SYSLST. A dump of virtual storage is produced if a critical message is issued. Output records are to consist of VSAM disk addresses only. The input file name is INPUT instead of the default of SORTIN1.

13

SORT control statement. Specifies that a sort based on one control field will be executed. There is one work file available.

14

RECORD control statement. Specifies that fixed-length records are to be sorted. The input record length is 80 bytes as indicated by the L1 value of the LENGTH operand. Since OPTION ADDROUT was specified, the default for L2 is 61 (5+the sum of the control field lengths for VSAM files). The L3 value indicates that each output record is 5 bytes long and contains the direct access address of its corresponding sorted input record. L3 could have been defaulted and DFSORT/VSE would have calculated the correct L3 value.

15

INPFIL control statement. Specifies that the input file is a VSAM file.

16

OUTFIL control statement. Specifies that the output file is an entry-sequenced VSAM file.

17-18

End-of-data file and end-of-job statements.

Example 5. Sort with Tape Input and Output, INCLUDE and OUTREC

```
// JOB SAMPLE05                                                  01
// ASSGN   SYS001,X'284'      SORT OUTPUT -- TAPE                 02
// ASSGN   SYS002,X'282'      SORT INPUT -- TAPE                  03
// ASSGN   SYS003,X'161'      SORT WORK -- DISK                   04
// ASSGN   SYS004,X'162'      SORT WORK -- DISK                   05
// DLBL    SORTWK1,,1,DA                                         06
// EXTENT  SYS003,111111,1,0,760,38                              07
// EXTENT  SYS003,111111,1,1,380,38                              08
// EXTENT  SYS004,222222,1,2,380,38                              09
// EXTENT  SYS004,222222,1,3,760,38                              10
// EXEC    SORT,SIZE=48K                                         11
   OPTION LABEL=(U,U)                                           12
   SORT FIELDS=(1,4,A,6,12,A),FORMAT=CH                         13
   RECORD TYPE=F,LENGTH=80                                      14
   INPFIL BLKSIZE=800,BYPASS                                    15
   OUTFIL BLKSIZE=800                                           16
   INCLUDE COND=(6,1,GE,C'M'),FORMAT=CH                         17
   OUTREC FIELDS=(1,4,6,16)                                     18
/*                                                              19
/&                                                              20
```

Line Explanation

01
JOB statement. Indicates the beginning of control information for the SAMPLE05 job.

02-05
ASSGN statements. Assign devices to be used as output, input, and work units. The output and input units, SYS001 and SYS002, respectively, are assigned to tape units at addresses 284 and 282. Four work units, SYS003 and SYS004, are assigned to disk.

06-10
DLBL and EXTENT statements. Define four extents on the two disk work volumes; 152 tracks are allocated in all.

11
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 48 KB of virtual storage directly available for its use.

12
OPTION control statement. Specifies that the output and input tapes are both unlabeled.

13
SORT control statement. Specifies that a sort based on two control fields will be executed.

14
RECORD control statement. Specifies that fixed-length records are to be sorted. The input record length is 80 bytes as indicated by the L1 value of the LENGTH operand. All other length specifications are defaulted.

15
INPFIL control statement. Specifies that the input block size is 800 bytes, and that input blocks causing I/O errors are to be bypassed.

16
OUTFIL control statement. Specifies that the output block size is 800 bytes.

17
INCLUDE control statement. Specifies that the sixth byte of every record is to be examined. Records whose sixth byte contains a character collating greater than or equal to M are to be included in the sort; all others are to be discarded.

18
OUTREC control statement. Specifies that the output records are to consist of two fields taken from the input records. The first field is 4 bytes long and begins at byte 1 of the input record. The second field begins at byte 6 of the input record and is 16 bytes long. The effective output record length is thus 20 bytes.

19-20
End-of-data file and end-of-job statements.

Example 6. Sort with Tape Input and Output, ALTSEQ and SUM

```
// JOB SAMPLE06                                                  01
// ASSGN  SYS001,X'284'  SORT OUTPUT -- TAPE                     02
// ASSGN  SYS002,X'282'  SORT INPUT -- TAPE                      03
```

```
|   // ASSGN  SYS003,X'161'  SORT WORK -- DISK                    04   |
|   // ASSGN  SYS004,X'162'  SORT WORK -- DISK                    05   |
|   // DLBL   SORTWK1,,,DA                                      06    |
|   // EXTENT SYS003,111111,1,0,760,38                          07    |
|   // EXTENT SYS003,111111,1,1,380,38                          08    |
|   // EXTENT SYS004,222222,1,2,380,38                          09    |
|   // EXTENT SYS004,222222,1,3,760,38                          10    |
|   // EXEC   SORT,SIZE=48K                                     11    |
|      OPTION LABEL=(U,U)                                       12    |
|      SORT   FIELDS=(6,12,AQ,A)                                13    |
|      RECORD TYPE=F,LENGTH=80                                  14    |
|      INPFIL BLKSIZE=800,BYPASS                                15    |
|      OUTFIL BLKSIZE=800                                       16    |
|      SUM  FIELDS=(51,6,ZD)                                    17    |
|      ALTSEQ CODE=(5BEA,7BEB,7CEC)                             18    |
|   /*                                                          19    |
|   /&                                                          20    |
|                                                                     |
|_____________________________________________________________________|
```

Line Explanation

01

JOB statement. Indicates the beginning of control information for the SAMPLE06 job.

02-05

ASSGN statements. Assign devices to be used as output, input, and work units. The output and input units, SYS001 and SYS002, respectively, are assigned to tape units at addresses 284 and 282. Four work units, SYS003 and SYS004, are assigned to disk.

06-10

DLBL and EXTENT statements. Define four extents on the two disk work volumes; 152 tracks are allocated in all.

11

EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 48 KB of virtual storage directly available for its use.

12

OPTION control statement. Specifies that the output and input tapes are both unlabeled.

13

SORT control statement. Specifies that a sort based on one control field will be executed. An alternative collating sequence (specified in the ALTSEQ statement, line 18) is to be used.

14

RECORD control statement. Specifies that fixed-length records are to be sorted. The input record length is 80 bytes as indicated by the L1 value of the LENGTH operand. All other length specifications are defaulted.

15

INPFIL control statement. Specifies that the input block size is 800 bytes, and that input blocks causing I/O errors are to be bypassed.

16

OUTFIL control statement. Specifies that the output block size is 800 bytes.

17

SUM control statement. Specifies that if two records are found with equal sort control fields, the contents of their summary field in bytes 51 through 56 are to be added. The sum is placed in one of the records and the other record is deleted.

18

ALTSEQ control statement. Specifies that X'5B' is to collate as X'EA', X'7B' is to collate as X'EB', and X'7C' is to collate as X'EC'--in other words, national characters are to collate at the end of the alphabet.

19-20

End-of-data file and end-of-job statements.

Example 7. Sort with SAM ESDS Input, Output, and Work Files

```
// JOB SAMPLE07                                                  01
*  VSAM-MANAGED SAM EXAMPLE                                    02
// DLBL   SORTIN1,'INPUT.FILE',,VSAM                           03
// DLBL   SORTOUT,'IMPLICIT.DEFINE',10,VSAM,RECORDS=1000,      04
               RECSIZE=500,DISP=(,KEEP)                       05
// EXTENT ,DISK01                                              06
// DLBL   SORTWK1,,,VSAM,DISP=(,DELETE)                        07
// EXEC   SORT,SIZE=40K                                        08
   SORT   FIELDS=(10,20,CH,A)                                  09
   RECORD TYPE=V,LENGTH=(500,,,100,400)                        10
   INPFIL BLKSIZE=4000                                         11
   OUTFIL BLKSIZE=4000                                         12
/*                                                             13
/&                                                             14
```

Line Explanation

01

JOB statement. Indicates the beginning of control information for the SAMPLE07 job.

02

Comment statement.

03

DLBL statement. Defines sort input as file 'INPUT.FILE' managed by VSAM. This file was defined by IDCAMS and previously loaded by another program.

04-05

DLBL statement. Defines sort output as managed by VSAM. This file will be implicitly defined to be able to contain a thousand 500-byte records. It has a retention period of 10 days.

06

EXTENT statement. Defines to VSAM that the output file must be defined on a disk with serial number DISK01. Can be omitted if there is a default model cataloged which indicates the required disk.

07

DLBL statement. Defines a sort work file as managed by VSAM. This file is assumed to have been previously defined by IDCAMS with the NOALLOCATE(NAL) and REUSE attributes. Its space will be allocated when DFSORT/VSE opens it and deallocated when DFSORT/VSE closes it.

08

EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 40 KB of virtual storage directly available for its use.

09

SORT control statement. Specifies that a sort based on one control field will be executed. By default, one work file and one input file are assumed.

10

RECORD control statement. Specifies that variable-length format records are to be sorted. The maximum input record length is 500 bytes as indicated by the L1 value of the LENGTH operand. By default, the values for L2 and L3 are the same as L1. The L4 value indicates that the minimum input record length is 1000, and the most common (modal) record length is 400 bytes, as indicated by the L5 parameter.

11

INPFIL control statement. Specifies that the input block size is 4000 bytes. VSAM is not specified, so DFSORT/VSE will use SAM access.

12

OUTFIL control statement. Specifies that the output block size is 4000 bytes. ESDS is not specified, so DFSORT/VSE will use SAM access.

13-14

End-of-data file and end-of-job statements.

Note: All VSAM definitions are assumed to be in the master catalog.

Example 8. Sort with VSAM Input and Output Files and SAM ESDS Work File

```
// JOB SAMPLE08                                                    01
*  DYNAMIC SECONDARY ALLOCATION OF WORK FILE                       02
// DLBL     SORTIN1,'SORTINP',,VSAM                                03
// DLBL     SORTOUT,'SORTOTP',,VSAM                                04
// DLBL     SORTWK1,'%DOS.WORKFILE.SYS001.SORT',,VSAM,             05
          DISP=(NEW,DELETE),RECSIZE=80,RECORDS=(1000,500)         06
// EXTENT   ,SYSWK5                                                07
// EXEC     SORT,SIZE=64K                                          08
   SORT FIELDS=(12,4,A),FORMAT=BI,WORK=1                          09
   RECORD TYPE=F,LENGTH=80                                        10
   INPFIL TOL,VSAM                                                11
   OUTFIL TOL,ESDS,REUSE                                          12
   OPTION PRINT=ALL,ROUTE=LST                                     13
/*                                                                14
/&                                                                15
```

Line Explanation

01

JOB statement. Indicates the beginning of control information for the SAMPLE08 job.

02

Comment statement.

03
DLBL statement. Defines sort input as file SORTINP managed by VSAM. This file was defined by IDCAMS and previously loaded by another program.

04
DLBL statement. Defines sort output as file SORTOTP managed by VSAM. This file was defined by IDCAMS as ESDS reusable file.

05-06
DLBL statement. Defines sort work file '%DOS.WORKFILE.SYS001.SORT' as a SAM ESDS file. This file will be allocated to contain up to one thousand 80-byte records in its first extent, which will be allocated as a single extent. When the first extent is full, the second extent will be allocated to contain an additional five hundred 80-byte records.

07
EXTENT statement. Defines to VSAM that the work file must be defined on a disk with serial number SYSWK5.

08
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 64 KB of virtual storage directly available for its use.

09
SORT control statement. Specifies that a sort based on one control field will be executed. The format of the control field is binary and one work file is used.

10
RECORD control statement. Specifies that fixed-length records are to be sorted. The input record length is 80 bytes as indicated by the L1 value of the LENGTH operand. All other length specifications are defaulted.

11
INPFIL control statement. Specifies that the input file is a VSAM file and that DFSORT/VSE should tolerate a warning code from VSAM when opening the VSAM input file.

12
OUTFIL control statement. Specifies that the output file is a reusable entry sequence VSAM file and that DFSORT/VSE should tolerate a warning code from VSAM when opening the VSAM output file.

13
OPTION control statement. Specifies that all DFSORT/VSE messages are to be written. All DFSORT/VSE messages are to be routed to SYSLST, and critical messages to the system console.

14-15
End-of-data file and end-of-job statements.

Note: All VSAM definitions are assumed to be in the master catalog.

Example 9. Sort with VSAM Input, Disk Output, Work Files, and LOCALE

```
// JOB SAMPLE09                                                  01
*   GETVIS SORTING WITH INCLUDE AND LOCALE PROCESSING           02
// LIBDEF *,SEARCH=PRD2.LEVSE                                    03
// DLBL SORTIN1,'INPUT.FRENCH.CANADA',,VSAM                      04
```

```
|   // ASSGN SYS001,X'E4F'                                          05    |
|   // DLBL SORTOUT,'OUTPUT.FRENCH.CANADA',1                        06    |
|   // EXTENT SYS001,,,,12000,100                                   07    |
|   // ASSGN SYS004,X'E4F'                                          08    |
|   // DLBL SORTWK1,'WRK1FIL',0                                     09    |
|   // EXTENT SYS004,,,,12100,100                                   10    |
|   // EXEC SORT,SIZE=200K                                          11    |
|     OPTION LOCALE=FR_CA,GVSIZE=MAX,DIAG                           12    |
|     SORT FIELDS=(1,20,CH,A,25,1,BI,D,30,10,CH,A)                  13    |
|     RECORD TYPE=F,LENGTH=60                                       14    |
|     INPFIL TOL,VSAM                                               15    |
|     INCLUDE COND=(40,6,CH,EQ,50,6,CH)                            16    |
|   /*                                                              17    |
|   /&                                                              18    |
|                                                                         |
|_________________________________________________________________________|
```

LINE EXPLANATION

01

JOB statement. Indicates the beginning of control information for the SAMPLE09 job.

02

Comment statement.

03

LIBDEF statement. Defines the sublibrary that contains LE/VSE dynamically loadable routines and locales. The information you specify on this line depends on where LE/VSE is installed at your site.

04

DLBL statement. Defines the sort input as file INPUT.FRENCH.CANADA managed by VSAM. This file was defined by IDCAMS and previously loaded by another program.

05-07

ASSGN, DLBL, and EXTENT statements. Define the sort output as file OUTPUT.FRENCH.CANADA on the disk unit at address X'E4F'. A one day retention period is requested.

08-10

ASSGN, DLBL, and EXTENT statements. Define the sort work file as file WRK1FIL on the disk unit at address X'E4F'.

11

EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 200 KB of virtual storage directly available for its use. The rest of the partition space is available as GETVIS area.

12

OPTION control statement. Specifies that the installation default for LOCALE is to be overridden by FR_CA. The locale for the French language and the cultural conventions of Canada will be active. The GVSIZE option specifies that DFSORT/VSE dynamically determines the maximum amount of the partition GETVIS area to be used for GETVIS sorting. Special diagnostic messages are to be produced.

13

SORT control statement. Specifies that a sort based on three control fields will be executed. The character (CH) control fields will be sorted according to the collating rules defined in locale FR_CA.

14

RECORD control statement. Specifies that fixed length records are to be sorted. The input record length is 60 bytes as indicated by the L1 value of the LENGTH operand. All other length specifications are defaulted.

15

INPFIL control statement. Specifies that the input file is a VSAM file and that DFSORT/VSE should tolerate a warning code from VSAM when opening the VSAM input file.

16

INCLUDE control statement. 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 file. The character (CH) compare fields will be compared according to the collating rules defined in locale FR_CA.

17-18

End-of-data file and end-of-job statements.

Note: All VSAM definitions are assumed to be in the master catalog

Example 10. Sort with Output to Printers and Punch Devices

```
// JOB SAMPLE10                                                  01
// ASSGN SYS030,X'E4F'   SORTWK1                                 02
// ASSGN SYS001,SYS030   SORTIN1                                 03
// ASSGN SYS002,SYS030   SORTIN2                                 04
// DLBL SORTIN1,'SORTIN1',0                                      05
// EXTENT SYS001,,,,11100,100                                    06
// DLBL SORTIN2,'SORTIN2',0                                      07
// EXTENT SYS002,,,,11200,100                                    08
// DLBL SORTWK1,,0                                               09
// EXTENT SYS030,,,,11000,100                                    10
*                                                                11
*  Sort records from two input files and output into SYSLST     12
*                                                                13
// EXEC SORT,SIZE=256K                                           14
 OPTION SORTOUT=LST                                             15
 SORT FIELDS=(5,4,A,9,4,A),FORMAT=CH,FILES=2,WORK=1             16
 SUM FIELDS=(17,4,ZD)                                           17
 RECORD TYPE=F,LENGTH=80                                        18
 INPFIL BLKSIZE=3600                                            19
/*                                                              20
// DLBL SORTIN1,'SORTIN1',0                                     21
// EXTENT SYS001,,,,11100,100                                   22
// DLBL SORTIN2,'SORTIN2',0                                     23
// EXTENT SYS002,,,,11200,100                                   24
// DLBL SORTWK1,,0                                              25
// EXTENT SYS030,,,,11000,100                                   26
*                                                               27
*  Sort records from two input files and output into SYSPCH     28
*                                                               29
// EXEC SORT,SIZE=256K                                          30
 OPTION SORTOUT=PCH                                             31
 SORT FIELDS=(5,4,A,9,4,A),FORMAT=CH,FILES=2,WORK=1             32
 SUM FIELDS=(17,4,ZD)                                           33
 RECORD TYPE=F,LENGTH=80                                        34
 INPFIL BLKSIZE=3600                                            35
/*                                                              36
/&                                                              37
```

Line Explanation

01

JOB statement. Indicates the beginning of control information for the SAMPLE10 job.

02-04
ASSGN statements. Assign devices to be used as work and input units. The work unit, SYS030, is assigned to disk. The input units, SYS001 and SYS002, are assigned to the same unit as SYS030.

05-08
DLBL and EXTENT statements. Define two sort input files, SORTIN1 and SORTIN2.

09-10
DLBL and EXTENT statements. Define the sort work file, SORTWK1.

11-13
Comment statements.

14
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 256 KB of virtual storage directly available for its use.

15
OPTION control statement. Specifies that the output file is to be written to SYSLST.

16
SORT control statement. Specifies that a sort based on two control fields is to be executed. The format of both control fields is character. Two input files are to be sorted and one work file will be used.

17
SUM control statement. Specifies that if two records are found with equal sort control fields, the contents of their summary field in bytes 17 through 20 are to be added. The sum is placed in one of the records and the other record is deleted.

18
RECORD control statement. Specifies that fixed-length records are to be sorted. The input record length is 80 bytes as indicated by the L1 value of the LENGTH operand. All other length specifications are defaulted.

19
INPFIL control statement. Specifies that the input block size is 3600 bytes.

20
Comment statement.

21-24
DLBL and EXTENT statements. Define two sort input files, SORTIN1 and SORTIN2.

25-26
DLBL and EXTENT statements. Define the sort work file, SORTWK1.

27-29
Comment statements.

30
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 256 KB of virtual storage directly available for its use.

31
OPTION control statement. Specifies that the output is to be written to SYSPCH.

32

SORT control statement. Specifies that a sort based on two control fields is to be executed. The format of both control fields is character. Two input files are sorted and one work file is used.

33

SUM control statement. Specifies that if two records are found with equal sort control fields, the contents of their summary field in bytes 17 through 20 are to be added. The sum is placed in one of the records and the other record is deleted.

34

RECORD control statement. Specifies that fixed-length records are to be sorted. The input record length is 80 bytes as indicated by the L1 value of the LENGTH operand. All other length specifications are defaulted.

35

INPFIL control statement. Specifies that the input block size is 3600 bytes.

36-37

End-of-data file and end-of-job statements.

Example 11. COBOL, DFSORT/VSE and Two-digit Year Date Example

```
// JOB SAMPLE11                                                  01
// LIBDEF *,SEARCH=(PRD2.SCEEBASE)                               02
// OPTION LINK,NODECK,LIST                                       03
// EXEC   IGYCRCTL,SIZE=IGYCRCTL,PARM='OBJECT'                   04
      *------------------------------------------------------------
      *   USING DFSORT/VSE'S YEAR 2000 FEATURES WITH COBOL         |
      *                                                            |
      *   THE COBOL PROGRAM CALLS DFSORT/VSE TO SORT RECORDS IN    |
      *   ASCENDING ORDER. A SIX-BYTE DATE FIELD IN THE INPUT RECORD |
      *   (BYTES 7-12) IS USED AS A KEY.                           |
      *                                                            |
      *   THE PROGRAM USES DFSORT/VSE'S SORT, OPTION AND OUTREC    |
      *   CONTROL STATEMENTS FROM SYSIPT TO SORT WITH A 1961-2060  |
      *   FIXED CENTURY WINDOW.  IT TRANSFORMS THE TWO-BYTE YEAR   |
      *   FIELD INTO A FOUR-BYTE YEAR FIELD IN THE OUTPUT RECORD.  |
      *------------------------------------------------------------
       ID DIVISION.                                              05
        PROGRAM-ID. CASEY2K.                                     06
       ENVIRONMENT DIVISION.                                     07
        INPUT-OUTPUT SECTION.                                    08
          FILE-CONTROL.                                          09
             SELECT SORTIN1    ASSIGN TO SORTIN1.                10
             SELECT SORTOUT    ASSIGN TO SORTOUT.                11
             SELECT SORT-FILE  ASSIGN TO SORTFIL.                12
       DATA DIVISION.                                            13
         FILE SECTION.                                           14
          FD SORTIN1 RECORD CONTAINS 160 CHARACTERS              15
             LABEL RECORD STANDARD BLOCK 27840                   16
             DATA RECORDS ARE SORTIN1-RECORD.                    17
          01 SORTIN1-RECORD.                                     18
            05      FILLER        PIC X(6).                      19
            05      SORTIN1-YY    PIC X(2).                      20
            05      SORTIN1-MM    PIC X(2).                      21
            05      SORTIN1-DD    PIC X(2).                      22
            05      FILLER        PIC X(148).                    23
          FD SORTOUT RECORD CONTAINS 162 CHARACTERS              24
             LABEL RECORD STANDARD BLOCK 28188                   25
             DATA RECORDS ARE SORTOUT-RECORD.                    26
          01 SORTOUT-RECORD.                                     27
            05 FILLER             PIC X(6).                     28
            05 FOUR-BYTE-YEAR     PIC X(4).                     29
            05 TWO-BYTE-MONTH     PIC X(2).                     30
            05 TWO-BYTE-DAY       PIC X(2).                     31
```

```
| |              05 FILLER            PIC X(148).                        |  32
| |            SD SORT-FILE RECORD CONTAINS 160 CHARACTERS                 |  33
| |               DATA RECORD SORT-RECORD.                                 |  34
| |            01 SORT-RECORD.                                             |  35
| |              05       FILLER      PIC X(6).                          |  36
| |              05       SORT-KEY    PIC X(6).                          |  37
| |              05       FILLER      PIC X(148).                        |  38
| |           WORKING-STORAGE SECTION.                                     |  39
| |          PROCEDURE DIVISION.                                           |  40
| |            MASTER SECTION.                                             |  41
| |         *--------------------------------------------------------------|--
| |         *     CALL DFSORT/VSE TO SORT THE RECORDS IN ASCENDING ORDER   |
| |         *     USING DFSORT/VSE'S SORT, OPTION, AND OUTREC CONTROL      |
| |         *     STATEMENTS FROM SYSIPT.                                  |
| |         *--------------------------------------------------------------|--
| |               MOVE "SYSIPT" TO SORT-CONTROL.                           |  42
| |               SORT SORT-FILE                                           |  43
| |                 ON ASCENDING KEY SORT-KEY                              |  44
| |                 USING SORTIN1                                          |  45
| |                 GIVING SORTOUT.                                        |  46
| |                 IF SORT-RETURN > 0                                     |  47
| |                   DISPLAY "SORT FAILED".                               |  48
| |               STOP RUN.                                                |  49
| |    /*                                                                  |  50
| |    // EXEC LNKEDT                                                      |  51
| |    // DLBL SORTIN1,'SORTIN1',0,VSAM,DISP=(OLD,KEEP),CAT=VSESPUC        |  52
| |    // DLBL SORTOUT,'SORTOUT',0,VSAM,RECSIZE=4096,RECORDS=(50,100),     | *
| |                  DISP=(NEW,KEEP),CAT=VSESPUC                           |  53
| |    // LIBDEF PHASE,SEARCH=(PRD2.SCEEBASE,SORT.SM3R4,SDL)               |  54
| |    // EXEC ,SIZE=180K                                                  |  55
| |      SORT FIELDS=(7,2,Y2C,A,9,4,CH,A)                                  |  56
| |      OPTION Y2PAST=1961                                                |  57
| |      OUTREC FIELDS=(1,6,7,2,Y2C,9,152)                                 |  58
| |    /*                                                                  |  59
| |    /&                                                                  |  60
  |                                                                        |
  |________________________________________________________________________|
```

| This example shows how you can use DFSORT/VSE's Year 2000 features when
| using the COBOL SORT verb. When you code a SORT verb in your COBOL
| program, COBOL automatically generates DFSORT/VSE control statements. If
| you have a two-digit year date you would like DFSORT/VSE to process, you
| need to pass the following information to DFSORT/VSE through SYSIPT:

1. | A new SORT statement, which identifies two-digit year fields using
   | DFSORT/VSE's Year 2000 formats (Y2x)

2. | The Y2PAST operand on the OPTION control statement

| If you also want to transform the two-digit year field from the input
| record to a four-digit year field in the output record, you can pass an
| OUTREC statement through SYSIPT as well.

| This example shows how DFSORT/VSE can process two-digit years from COBOL
| using DFSORT/VSE control statements passed through SYSIPT.

| **Line Explanation**

| 01
JOB statement. Indicates the beginning of control information for
| the SAMPLE16 job.

| 02
LIBDEF statement. Defines the sublibrary that contains the COBOL
| dynamically loadable routines. The information you specify on this
| line depends on where COBOL is installed at your site.

| 03
OPTION statement. Specifies the job control options.

| 04
EXEC statement. Calls COBOL by its phase name (IGYCRCTL) and
| reserves space for the specified phase in the partition program
| area. The PARM 'OBJECT' will be passed to the COBOL compiler at
| execution time.

| 05-50
COBOL program statements.

| Note that line 42 specifies that the DFSORT/VSE control statements
| in SORT-CONTROL override the COBOL generated DFSORT/VSE control
| statements.

| 51
EXEC statement. Calls the linkage editor to linkedit the COBOL
| program.

| 52
DLBL statement. Defines sort input as file SORTIN1 managed by VSAM
| and cataloged in VSESPUC.

| 53
DLBL statement. Defines sort output as SORTOUT managed by VSAM.
| This file will be allocated to contain fifty 4096-byte records in
| its first extent, which will be allocated as a single extent. When
| the first extent is full, the second extent will be allocated to
| contain an additional one hundred 4096-byte records. It is also
| cataloged in VSESPUC.

| 54
LIBDEF statement. Defines the sublibrary that contains COBOL
| run-time and DFSORT/VSE dynamically loadable routines. The
| information you specify on this line depends on where COBOL and
| DFSORT/VSE are installed at your site.

| 55
EXEC statement. Calls the linkedited COBOL program and reserves
| 180K bytes for it in the partition program area.

| 56
DFSORT/VSE SORT control statement in SYSIPT. A two-digit character
| year field is in bytes 7 and 8. A two-digit character month field
| is in bytes 9 and 10. A two-digit character day field is in bytes
| 11 and 12.

| 57
DFSORT/VSE OPTION control statement in SYSIPT. Specifies a fixed
| century window from 1961 to 2060.

| 58
DFSORT/VSE OUTREC control statement in SYSIPT. Specifies the
| output record is to contain 5 fields from the input record: 1-6
| (FILLER), 7-8 (SORTIN1-YY), 9-10 (SORTIN1-MM), 11-12 (SORTIN1-DD),
| and 13-160 (FILLER). Note that the two-digit year in columns 7-8
| is transformed to a four-digit year in columns 7-10.

| 59-60
End-of-data file and end-of-job statements.

| **Notes:**

| 1. The SYSIPT Year 2000 support requires that LE/VSE 1.4 PTF UQ05847 be
| applied.

| 2. Control statements that you can include at run time are as follows and
| if specified, must be in the order listed below:

a. | SORT or MERGE (used to replace the SORT or MERGE statement
| generated by the compiler)

b. | SMS=nnnnn were nnnnn is the length, in bytes, of the most frequent
| record size. (Ignored if the SD is **not** variable.)

c. | OPTION (except FILNM=)

d. | Other DFSORT/VSE control statements (ALTSEQ, ANALYZE, INCLUDE,
| INREC, OMIT, OUTREC, or SUM, in any order)

e. | If you have multiple sorts in a single COBOL program you can set
| SORT-CONTROL to SYSIPT and use the following JCL setup:

```
|               // EXEC ...
|                  SORT FIELDS= ... override statements for first sort
|               /*
|                  SORT FIELDS= ... override statements for second sort
|               /*
|                  SORT FIELDS= ... override statements for third sort
|               /*
```

8.3 Merge Examples

This section includes two merge examples.

Example 12. Merge with Four Files (Tape/Disk) Input and Tape Output

```
// JOB SAMPLE12                                                  01
// ASSGN   SYS001,X'282'      MERGE OUTPUT                       02
// ASSGN   SYS002,X'284'      MERGE INPUT                        03
// ASSGN   SYS003,X'191'      MERGE INPUT                        04
// ASSGN   SYS004,X'283'      MERGE INPUT                        05
// ASSGN   SYS005,X'192'      MERGE INPUT                        06
// DLBL    SORTIN2                                               07
// EXTENT  SYS003,191191,1,0,20,70                               08
// DLBL    SORTIN4                                               09
// EXTENT  SYS005,192192,1,0,20,70                               10
// EXEC    SORT,SIZE=64K                                         11
  OPTION LABEL=(U,U,S,U,S)                                      12
  MERGE FIELDS=(21,4,ZD,D,9,8,PD,A,30,4,BI,A,40,4,CH,D,         13
     35,4,CH,A,70,4,FL,A,90,14,CH,D,79,5,CH,A,86,2,             14
     BI,A,5,4,PD,D,25,3,CH,D,130,4,BI,A),FILES=4                15
  RECORD TYPE=V,LENGTH=(154,,154)                               16
  INPFIL BLKSIZE=1544                                           17
  OUTFIL BLKSIZE=1544                                           18
/*                                                              19
/&                                                              20
```

Line Explanation

01
JOB statement. Indicates the beginning of control information for the SAMPLE11 job.

02
ASSGN statement. Assigns the output unit, SYS001, to the tape unit at address 282.

03-06
ASSGN statements. Assign tape units SYS002 and SYS004 and disk units SYS003 and SYS005 as input units.

07-10
DLBL and EXTENT statements. Define two extents on the two disk input volumes. A total of 140 tracks is allocated.

11
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 64 KB of virtual storage directly available for its use.

12
OPTION control statement. Specifies that the output tape SYS001 and input tape units SYS002 and SYS004 are unlabeled. Input disk units SYS003 and SYS005 use standard labels.

13-15
MERGE control statement. Specifies that a merge based on 12 control fields is to be executed. Four input files are to be merged.

16
RECORD control statement. Specifies that variable-length records are to be merged. The maximum input record length is 154 bytes as indicated by the L1 value of the LENGTH operand. By default, the value for L2 is the same as L1. The maximum record length after an E35 user exit routine is 154, as indicated by the L3 parameter.

17
INPFIL control statement. Specifies that the input block size is 1544 bytes.

18
OUTFIL control statement. Specifies that the output block size is 1544 bytes.

19-20
End-of-data file and end-of-job statements.

Example 13. Merge with EQUALS

```
|                                                                          |
|   // JOB SAMPLE13                                                 01   |
|   // ASSGN SYS011,X'E4F'   MERGE OUTPUT                           02   |
|   // ASSGN SYS001,SYS011   MERGE INPUT                            03   |
|   // ASSGN SYS002,SYS011   MERGE INPUT                            04   |
|   // ASSGN SYS003,SYS011   MERGE INPUT                            05   |
|   // ASSGN SYS004,SYS011   MERGE INPUT                            06   |
|   // DLBL SORTIN1,'SORTIN1',0                                     07   |
|   // EXTENT SYS001,,,,11100,50                                    08   |
|   // DLBL SORTIN2,'SORTIN2',0                                     09   |
|   // EXTENT SYS002,,,,11150,50                                    10   |
|   // DLBL SORTIN3,'SORTIN3',0                                     11   |
|   // EXTENT SYS003,,,,11200,50                                    12   |
|   // DLBL SORTIN4,'SORTIN4',0                                     13   |
|   // EXTENT SYS004,,,,11250,50                                    14   |
|   // DLBL SORTOUT,'SORTOUT',0                                     15   |
|   // EXTENT SYS011,,,,11300,200                                   16   |
|   *                                                               17   |
|   // EXEC SORT,SIZE=100K                                          18   |
|    OPTION PRINT=ALL,ROUTE=LST,DIAG,DUMP,                          19   |
|      FILNM=(SORTOUT),EQUALS                                       20   |
|    MERGE FIELDS=(5,4,A),FORMAT=BI,FILES=4,NOEQUALS                21   |
|    RECORD TYPE=F,LENGTH=80                                        22   |
|    INPFIL BLKSIZE=3600                                            23   |
|    OUTFIL BLKSIZE=3600                                            24   |
|   /*                                                              25   |
|   /&                                                              26   |
|                                                                          |
|__________________________________________________________________________|
```

Line Explanation

01
JOB statement. Indicates the beginning of control information for the SAMPLE12 job.

02-06
ASSGN statements. Assign devices to be used as output and input units. The output unit, SYS011, is assigned to disk. The input units, SYS001-SYS004,

are assigned to the same unit as SYS011.

07-14
DLBL and EXTENT statements. Define four merge input files, SORTIN1, SORTIN2, SORTIN3 and SORTIN4.

15-16
DLBL and EXTENT statements. Define the merge output file, SORTOUT.

17
Comment statement.

18
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 100 KB of virtual storage directly available for its use.

19-20
OPTION control statement. Specifies that all DFSORT/VSE messages are to be written. All DFSORT/VSE messages are to be routed to SYSLST. Diagnostic messages are produced. A dump of virtual storage is produced if a critical message is issued. The output file name is SORTOUT. This parameter could be omitted, as SORTOUT is the default. The EQUALS option specifies that the original sequence of records must be preserved from input to output.

21
MERGE control statement. Specifies that a merge based on one control field is to be executed. The format of the control field is binary. Four input files are to be merged. The EQUALS option specification in the OPTION control statement overrides the NOEQUALS option specification in the MERGE control statement.

22
RECORD control statement. Specifies that fixed-length records are to be merged. The input record length is 80 bytes as indicated by the L1 value of the LENGTH operand. All other length specifications are defaulted.

23
INPFIL control statement. Specifies that the input block size is 3600 bytes.

24
OUTFIL control statement. Specifies that the output block size is 3600 bytes.

25-26
End-of-data file and end-of-job statements.

8.4 Copy Examples

This section includes two copy examples.

Example 14. Copy with Tape Input and Output

```
|                                                                    |
|   // JOB SAMPLE14                                            01    |
|   // ASSGN   SYS001,X'283'      COPY OUTPUT                  02    |
|   // ASSGN   SYS002,X'282'      COPY INPUT                   03    |
|   // TLBL    SORTOUT                                         04    |
```

```
|   // EXEC    SORT,SIZE=48K                                    05    |
|      OPTION PRINT=ALL,LABEL=(N,U),ROUTE=LST                   06    |
|      SORT FIELDS=COPY                                        07    |
|      RECORD TYPE=V,LENGTH=(158,,,54,100)                     08    |
|      INPFIL BLKSIZE=1544,CLOSE=RWD                           09    |
|      OUTFIL BLKSIZE=1544,CLOSE=RWD                           10    |
|   /*                                                         11    |
|   /&                                                         12    |
|                                                                    |
|____________________________________________________________________|
```

Line Explanation

01
JOB statement. Indicates the beginning of control information for the SAMPLE13 job.

02-03
ASSGN statements. Assign devices to be used as output and input units. The output and input units, SYS001 and SYS002 are assigned to tape units at addresses 283 and 282, respectively.

04
TLBL statement. Defines a tape file for output.

05
EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 48 KB of virtual storage directly available for its use.

06
OPTION control statement. Specifies that all DFSORT/VSE messages are to be written. The output file has nonstandard labels. The input file is unlabeled. All DFSORT/VSE messages are to be routed to SYSLST.

07
SORT control statement. Specifies a copy application.

08
RECORD control statement. Specifies that variable-length records are to be copied. The maximum input record length is 158 bytes, as indicated by the L1 value of the LENGTH operand. By default, the values for L2 and L3 are the same as L1. The L4 value indicates that the minimum input record length is 54, and the most common (modal) record length is 100 bytes, as indicated by the L5 parameter.

09
INPFIL control statement. Specifies that the input block size is 1544 bytes, and DFSORT/VSE is to rewind the input volume at end of file.

10
OUTFIL control statement. Specifies that the output block size is 1544 bytes, and DFSORT/VSE is to rewind the output volumes at end of application.

11-12
End-of-data file and end-of-job statements.

Example 15. Copy with Multiple Input

```
// JOB SAMPLE15                                                  01
// ASSGN SYS011,X'E4F'   COPY OUTPUT                             02
// ASSGN SYS001,SYS011   COPY INPUT                              03
// ASSGN SYS002,SYS011   COPY INPUT                              04
// ASSGN SYS003,SYS011   COPY INPUT                              05
// ASSGN SYS004,SYS011   COPY INPUT                              06
// DLBL SORTIN1,'SORTIN1',0                                      07
// EXTENT SYS001,,,,11100,50                                     08
// DLBL SORTIN2,'SORTIN2',0                                      09
// EXTENT SYS002,,,,11150,50                                     10
// DLBL SORTIN3,'SORTIN3',0                                      11
// EXTENT SYS003,,,,11200,50                                     12
// DLBL SORTIN4,'SORTIN4',0                                      13
// EXTENT SYS004,,,,11250,50                                     14
// DLBL SORTOUT,'SORTOUT',0                                      15
// EXTENT SYS011,,,,11300,200                                    16
*                                                                17
* Copy records from four input files into one output file       18
*                                                                19
// EXEC SORT,SIZE=100K                                           20
 OPTION PRINT=ALL,ROUTE=LST,DIAG,DUMP,                          21
   FILNM=(SORTOUT)                                              22
 SORT FIELDS=COPY,FILES=4                                       23
 RECORD TYPE=F,LENGTH=80                                        24
 INPFIL BLKSIZE=3600                                            25
 OUTFIL BLKSIZE=3600                                            26
/*                                                              27
/&                                                              28
```

Line Explanation

01

JOB statement. Indicates the beginning of control information for the SAMPLE14 job.

02-06

ASSGN statements. Assign devices to be used as output and input units. The output unit, SYS011, is assigned to disk. The input units, SYS001-SYS004, are assigned to the same unit as SYS011.

07-14

DLBL and EXTENT statements. Define four copy input files, SORTIN1, SORTIN2, SORTIN3, and SORTIN4.

15-16

DLBL and EXTENT statements. Define the copy output file, SORTOUT.

17-19

Comment statements.

20

EXEC statement. Calls DFSORT/VSE directly by its name, SORT, and specifies that DFSORT/VSE is to have 100 KB of virtual storage directly available for its use.

21-22

OPTION control statement. Specifies that all DFSORT/VSE messages are to be written. All DFSORT/VSE messages are to be routed to SYSLST. Diagnostic messages are produced. A dump of virtual storage is produced if a critical message is issued. The output file name is SORTOUT. This parameter could be omitted, as SORTOUT is the default.

23

SORT control statement. Specifies a copy application. Four input files are to be copied to the output file.

24
RECORD control statement. Specifies that fixed-length records are to be copied. The input record length is 80 bytes as indicated by the L1 value of the LENGTH operand. All other length specifications are defaulted.

25
INPFIL control statement. Specifies that the input block size is 3600 bytes.

26
OUTFIL control statement. Specifies that the output block size is 3600 bytes.

27-29
End-of-data file and end-of-job statements.

8.5 ICETOOL Example

This section includes one ICETOOL example.

Example 16. ICETOOL with Various Operators

```
|                                                                      |
|    //  JOB  SAMPLE16                                                 |  01
|    //  DLBL IN1,'FLY.INPUT1',,VSAM,DISP=(OLD,KEEP)                  |  02
|    //  DLBL OUTJ69,'%%OUTJ69D',,VSAM,RECORDS=2000,RECSIZE=63        |  03
|    //  DLBL OUTJ82,'%%OUTJ69D',,VSAM,RECORDS=2000,RECSIZE=63        |  04
|    //  DLBL DEPTSD,'FLY.OUTPUT1',,VSAM,RECORDS=2500,RECSIZE=63      |  05
|    //  ASSGN SYS012,181                                             |  06
|    //  TLBL DEPTST,'FLY.BACKUP1',,111111,,1                        |  07
|    //  DLBL IN2,'FLY.INPUT2',,SD                                   |  08
|    //  EXTENT ,339000,,,6500,10                                    |  09
|    //  DLBL OUT4,'FLY.OUTPUT2',,SD                                 |  10
|    //  EXTENT ,339000,,,6600,10                                    |  11
|    //  DLBL SORTWK1,'SAMP.WORK',,SD                                |  12
|    //  EXTENT ,339000,,,6000,10                                    |  13
|    //  EXEC ICETOOL,SIZE=100K                                      |  14
|    * Define input files characteristics                            |  15
|      DEFINE NAME(IN1) TYPE(F)  LENGTH(53)                          |  16
|      DEFINE NAME(IN2) TYPE(V)  LENGTH(153)                         |  17
|      DEFINE NAME(OUTJ69) TYPE(F) LENGTH(63)                        |  18
|      DEFINE NAME(OUT4) TYPE(V)  LENGTH(153)                        |  19
|      DEFINE NAME(DEPTST) UNIT(012)                                 |  20
|    * Print report showing departments with less than 5 employees   |  21
|      OCCUR  FROM(IN1) LIST(LST) LOWER(5) BLANK -                   |  22
|        TITLE('Small Departments') PAGE -                           |  23
|        HEADER('Department') HEADER('Employees') -                  |  24
|        ON(45,3,CH)   ON(VALCNT)                                    |  25
|    * Copy and reformat selected records                            |  26
|       COPY USE FROM(IN1) TO(OUTJ69)                                |  27
|     USTART                                                         |  28
|    * Select J69 employees and reformat fields                      |  29
|       INCLUDE COND=(45,3,CH,EQ,C'J69')                             |  30
|       OUTREC FIELDS=(21,10,1X,1,15,45,3,34X)                       |  31
|     UEND                                                           |  32
|       COPY USE FROM(IN1) TO(OUTJ82)                                |  33
|     USTART                                                         |  34
|    * Select J82 employees and reformat fields                      |  35
|      INCLUDE COND=(45,3,CH,EQ,C'J82')                              |  36
|      OUTREC FIELDS=(21,10,1X,1,15,45,3,34X)                        |  37
|     UEND                                                           |  38
|    * Sort/save the resulting two files                             |  39
|       SORT FROM(OUTJ69,OUTJ82) TO(DEPTSD,DEPTST) USE               |  40
|     USTART                                                         |  41
|    * Sort by last name, first name                                 |  42
|      SORT FIELDS=(12,15,CH,A,1,10,CH,A)                            |  43
|     UEND                                                           |  44
|    * Do following operators even if a previous operator failed,    |  45
|    * but stop processing if a subsequent operator fails.           |  46
```

```
|     MODE STOP                                                          |  47
|    * Verify decimal fields                                             |  48
|       VERIFY FROM(IN2) ON(22,6,PD) ON(30,3,ZD)                         |  49
|    * Print statistics for record length and numeric fields             |  50
|       STATS FROM(IN2) ON(VLEN) ON(22,6,PD) ON(30,3,ZD)                 |  51
|    * Sort and produce total for each unique key                        |  52
|       SORT FROM(IN2) TO(OUT4) USE                                      |  53
|     USTART                                                             |  54
|    * Sort and produce totals in one record for each unique key         |  55
|      SORT FIELDS=(5,10,CH,A)                                           |  56
|      SUM FIELDS=(22,6,PD,30,3,ZD)                                      |  57
|     UEND                                                               |  58
|    * Print report containing:                                          |  59
|    *   - key and total for each unique key                            |  60
|    *   - lowest and highest of the totals                             |  61
|      DISPLAY FROM(OUT4) LIST(LST) -                                    |  62
|        TITLE('Unique key totals report') DATE TIME -                   |  63
|        ON(5,10,CH) ON(22,6,PD) ON(30,3,ZD) -                           |  64
|        MINIMUM('Lowest') MAXIMUM('Highest') PLUS                       |  65
|    /*                                                                  |  66
|    /&                                                                  |  67
|                                                                        |
|________________________________________________________________________|
```

This example shows how ICETOOL can be used to perform multiple operations in a single step.

Line Explanation

01
JOB statement. Indicates the beginning of control information for the SAMPLE15 job.

02-13
DLBL, EXTENT, TLBL, and ASSGN statements. Define the files used for the ICETOOL operations described below.

14
EXEC statement. Calls ICETOOL directly by its name, ICETOOL, and specifies that ICETOOL and DFSORT/VSE are to have 100 KB of virtual storage directly available for their use. Calls ICETOOL specifying the recommended SIZE of 100 KB.

15
Comment statement.

16-19
DEFINE operators for the input files. Specify the record type and the record length of the IN1, IN2, OUTJ69, and OUT4 files for the ICETOOL operations described below.

20
DEFINE operator for the tape file. Specifies the logical unit name for the tape.

21
Comment statement.

22-25
OCCUR operator. Specifies that a report will be printed to SYSLST detailing each value for the specified field in the IN1 file and the number of times that value occurs.

26
Comment statement.

27
COPY operator. Specifies that records from the IN1 file are copied to the OUTJ69 file using the DFSORT/VSE control statements in the following

DFSORT/VSE section. As a result, OUTJ69 file will contain a reformatted subset of the records from FLY.INPUT1 (those records containing 'J69' in the positions 45-47).

28-32
DFSORT/VSE section.

33
COPY operator. Specifies that records from the IN1 file are copied to the OUTJ82 file using the DFSORT/VSE control statements in the following DFSORT/VSE section. As a result, OUTJ82 file will contain a reformatted subset of the records from FLY.INPUT1 (those records containing 'J82' in the positions 45-47).

34-38
DFSORT/VSE section.

39
Comment statement.

40
SORT operator. Specifies that records from the OUTJ69 and OUTJ82 files are sorted to the DEPTSD file using the DFSORT/VSE control statements in the following DFSORT/VSE section. As a result, FLY.OUTPUT1 will contain the sorted, combined records from OUTJ69 and OUTJ82 files. FLY.OUTPUT1 is also copied to the DEPTST tape file as backup copy.

41-44
DFSORT/VSE section.

45-46
Comment statements.

47
MODE operator. Specifies that 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.

48
Comment statement.

49
VERIFY operator. Specifies invalid values, if any, in the specified decimal fields of the IN2 file. Used to stop subsequent operations if any invalid value is found in FLY.INPUT2.

50
Comment statement.

51
STATS operator. Specifies that the minimum, maximum, average, and total for the specified fields of the IN2 file will be printed.

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.

52
Comment statement.

53

SORT operator. Specifies that records from the IN2 file are sorted and summarized to the OUT4 file using the DFSORT/VSE control statements in the following DFSORT/VSE section. As a result, FLY.OUTPUT2 will contain one record from FLY.INPUT2 for each unique sort field with totals for the sum fields.

54-58
DFSORT/VSE section.

59-61
Comment statements.

62-65
DISPLAY operator. Specifies that a report will be printed to SYSLST detailing each sort and sum value for the OUT4 file resulting from the previous operation, and the lowest and highest value for each sum field will be specified.

66-67
End-of-data file and end-of-job statements.

A.0 Appendix A. Estimating Storage Requirements

Subtopics:

| A.1 Using Virtual Storage

| A merge or copy application only needs virtual storage. A sort application usually also needs work space on direct access devices.

| If there is plenty of virtual storage for data, a sort application may be
| able to run without work space. There is a relation between the two
| requirements: the less virtual storage available, the more work space you will need.

| You can find out in advance how much virtual storage a given application will need by submitting the complete control statement set with the addition of an ANALYZE control statement. DFSORT/VSE will then analyze all the control statements, make the usual optimization calculations, will issue all the usual messages (including those specifying how many records
| can be sorted with the given configuration; how much more virtual storage
| should be allocated, if the allocation was insufficient), and then terminate without sorting, merging, or copying.

Subtopics:

| A.1.1 Minimum Virtual Storage

| DFSORT/VSE needs a minimum of 32 KB of virtual storage of the partition program area.

The minimum requirement for a given application can be more than 32 KB. The major factors affecting the requirement are:

- Location of DFSORT/VSE modules
- Maximum block size of input and output files
- Size of the user exit routines
- Use of additional functions

A.1.2 Location of DFSORT/VSE Modules

If eligible DFSORT/VSE modules were not put in the SVA, they will have to
| be loaded into your partition program area instead. They need about 20 KB
| of virtual storage.

A.1.3 Maximum Block Size of Input and Output Files

| Extra virtual storage is required to process files with larger block sizes.

A.1.4 Size of Routines at User Exits

Unless the user exit routines are preloaded and DFSORT/VSE is program invoked, the size of any routines to be used at user exits must be taken into account in the storage allocated for DFSORT/VSE.

A.1.5 Use of Additional Functions

The extra partition program area storage required by some of the additional functions (for example, INCLUDE, OMIT, and SUM) is usually so small that it can for all practical purposes be ignored.

| In any case, the required extra partition program area storage is never
| more than 4 KB per additional function except for INCLUDE or OMIT. The
| INCLUDE and OMIT control statements are mutually exclusive and may require
| up to 32 KB of partition program area storage, depending on the complexity
| of the logical expression.

DFSORT/VSE requires additional partition GETVIS area for locale processing. The amount of storage needed can vary significantly according to the needs of LE/VSE, the locale used, and so on. 2 MB of the partition GETVIS area should be enough in most cases. The storage needed by LE/VSE will be reduced if LE/VSE phases CEEBINIT, CEEPLPKA, and CEEV003 were placed into the SVA. See *LE/VSE Library* for more information.

If you are planning to use locale processing with getvis sorting, then increase the amount of the partition GETVIS area by the GVSIZE value.

A.2 Using Work Space

Subtopics:

A.2.1 Introduction

| When a sort application cannot be performed entirely in virtual storage,
| DFSORT/VSE must use work space. The amount of work space required depends on the:

- Size of your input files
- | Amount of virtual storage available to DFSORT/VSE
- Type of devices you use

A.2.2 Work File Devices

Work storage must be on disk devices only (tape devices cannot be used for work storage). Real or virtual disks can be used. The type of device selected for work files can have a significant effect on performance.

A.2.3 Number of Devices for Work Files

Although one work file is sufficient, using two or more work files on separate devices usually reduces the elapsed time of the application significantly. In general, using more than three work files does not reduce elapsed time any further, and is only necessary if the work files are small or the input file size is large. Regardless, no more than 9 work files can be specified.

A.2.4 Allocation of Work Files

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 files.

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:

- Work files should be the same size as the input files plus 25%.
- In the worst possible case, you might need to add 80% instead 25%; this would be if all the following conditions were met:

  1. Control fields were neither binary nor character format.
  2. The sum of the control field lengths was close to the maximum.
  3. Input records were variable-length.
  4. Input records were not much longer than total control field lengths.

- For CKD work files, you must allocate a minimum of 4 tracks. For FBA work files, you must allocate a minimum of 64 blocks.

| These guidelines assume that a reasonable amount of virtual storage is
| available to DFSORT/VSE. Limiting the available amount of virtual storage
| can increase the amount of needed work space.

DFSORT/VSE can often run with less than the amount of work space indicated by the above guidelines.

A.3 DASD Capacity Considerations

DFSORT/VSE allows the work files to be allocated on two different device types, where the device types are considered to be the same if their track capacity and number of tracks per cylinder are the same. The types of devices available for intermediate storage are:

- IBM 9345 disk
- IBM 3390 disk
- IBM 3380 disk
- IBM 3375 disk
- IBM FBA devices
- IBM virtual (FBAV) disks
- IBM RAMAC Array DASD

The number of tracks per cylinder and maximum bytes used per block for CKD direct access devices is shown in Figure 82.

Figure 82. Number of Tracks per Cylinder for Direct Access Devices

Device	Tracks per Cylinder	Maximum Bytes used per Track
9345	15	46456
3390	15	56664
3380	15	47476
3375	12	35616

Subtopics:

- A.3.1 Exceeding DASD Work Space Capacity

A.3.1 Exceeding DASD Work Space Capacity

| If the allocated work space is exhausted, DFSORT/VSE will try to allocate
| a secondary extent dynamically for SAM ESDS and SD work files. This reduces the probability of exceeding work space capacity.

If the DASD work space is not sufficient to perform the sort application, DFSORT/VSE issues a message and terminates.

B.0 Appendix B. Data Format Examples

The format descriptions refer to the assembled data formats as used with IBM System/370 and 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').

ZSI	(zoned decimal, with sign ignored). The ZSI format can be represented as follows: zd...sd z 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. **Note:** ZSI can be used to make positive ZD fields printable or readable. For example, ZSI can be used to make the ZD field Z'mm' (hexadecimal FmCm) printable.
PD	(packed decimal, signed). Each digit of the decimal number is converted into its 4-bit binary equivalent. The sign indicator is put into the rightmost four bits of the number. Example: -247 becomes 2 4 7 - Decimal 24 7D Hexadecimal 00100100 01111101 Binary The number +247 becomes 247C in hexadecimal. **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 or PZ	(packed decimal, with sign and first digit ignored). The PD0 and PZ formats 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. **Note:** PD0 and PZ can be used for parts of PD fields. For example, in the PD field P'mmddyy' (hexadecimal 0mmddyyC), PD0 (or PZ) can be used separately for 0mmd (mm), mddy(dd), and dyyC (yy).
PSI	(packed decimal, with sign ignored). The PSI format can be represented as follows: dd....ds d is hexadecimal 0-9 and represents a decimal digit. s is hexadecimal 0-F and is ignored. **Note:** PSI can be used to make positive PD fields printable or readable. For example, PSI can be used to make the PD field P'ddd' (hexadecimal dddC) printable.
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 ISCII/ASCII, unsigned). This is similar to format CH but the characters are represented with ISCII/ASCII code.

	Example: AB7 becomes 41 42 37 Hexadecimal 01000001 01000010 00110111 Binary (ISCII/ASCII code)
CSL	(signed number, leading separate sign). This format refers to decimal data as punched into cards, and then assembled into EBCDIC code. Example: +247 punched in a card becomes + 2 4 7 Punched numeric data 4E F2 F4 F7 Hexadecimal 01001110 11110010 11110100 11110111 Binary EBCDIC code -247 becomes - 2 4 7 Punched numeric data 60 F2 F4 F7 Hexadecimal 01100000 11110010 11110100 11110111 Binary EBCDIC code
CST	(signed numeric, trailing separate sign). This has the same representation as the CSL format, except that the sign indicator is punched after the number. Example: 247+ punched on the card becomes F2 F4 F7 4E Hexadecimal
CLO	(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 **Note:** The overpunch sign bit is always 'C' for positive and 'D' for negative.
CTO	(signed numeric, trailing overpunch sign). This format has the same representation as for the CLO format, except that the sign indicator is overpunched on the last decimal digit of the number. Example: +247 with + overpunched on 7 becomes F2 F4 C7 hexadecimal
ASL	(signed numeric, ISCII/ASCII, leading separate sign). Similar to the CSL format but with decimal data assembled into ISCII/ASCII code. Example: +247 punched into card becomes + 2 4 7 Punched numeric data 2B 32 34 37 Hexadecimal 0101011 00110010 00110100 00110111 Binary ISCII/ASCII code Similarly -247 becomes 2D 32 34 37 hexadecimal
AST	(signed numeric, ISCII/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
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 be represented as follows: yy y is hexadecimal 0-9 and represents a year digit.

	Thus, 96 would be represented as hexadecimal 96 (decimal 96).
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).
Y2S	(two-digit, two-byte character or zoned decimal year data with special indicators). 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). The special indicators can be represented as follows: qxzx qx is hexadecimal 00, 40 (20, if DATA=A is specified in the INPFIL control statement) or FF. zx is hexadecimal 00-FF (although typically 00, 40 or 20 and FF). Thus, special indicators might be hexadecimal 0000, 0005, 4040, FFFF, FF85 and so on.

A detailed description of CH, ZD, PD, FI, BI, and FL data formats are found in the *Assembler Reference*.

C.0 Appendix C. EBCDIC and ISCII/ASCII Collating Sequences

Subtopics:

- C.1 EBCDIC
- C.2 ISCII/ASCII

C.1 EBCDIC

Figure 83 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.

Collating Sequence	Bit Configuration	Symbol	Meaning
0	00000000		
.			
.			
74	01001010	¢	Cent sign
75	01001011	.	Period, decimal point
76	01001100	<	Less than sign
77	01001101	(	Left parenthesis
78	01001110	+	Plus sign
79	01001111	|	Vertical bar, Logical OR
80	01010000	&	Ampersand
.			
.			
90	01011010	!	Exclamation point
91	01011011	$	Dollar sign
92	01011100	*	Asterisk
93	01011101	)	Right parenthesis
94	01011110	;	Semicolon
95	01011111	¬	Logical not
96	01100000	-	Minus, hyphen
97	01100001	/	Slash
107	01101011	,	Comma
108	01101100	%	Percent sign
109	01101101	_	Underscore
110	01101110	>	Greater than sign
111	01101111	?	Question mark
.			
.			
122	01111010	:	Colon
123	01111011	#	Number sign
124	01111100	@	Commercial At
125	01111101	'	Apostrophe, prime
126	01111110	=	Equal sign
127	01111111	"	Quotation marks
.			
.			
129	10000001	a	
130	10000010	b	
131	10000011	c	
132	10000100	d	
133	10000101	e	
.			
.			
134	10000110	f	
135	10000111	g	
136	10001000	h	
137	10001001	i	
.			
.			
145	10010001	j	
146	10010010	k	
147	10010011	l	
148	10010100	m	
149	10010101	n	
150	10010110	o	
151	10010111	p	
152	10011000	q	
153	10011001	r	
.			
.			
162	10100010	s	
163	10100011	t	
164	10100100	u	
165	10100101	v	
166	10100110	w	
167	10100111	x	
168	10101000	y	
169	10101001	z	
193	11000001	A	
194	11000010	B	
195	11000011	C	
196	11000100	D	
197	11000101	E	
198	11000110	F	
199	11000111	G	

200	11001000	H
201	11001001	I
.		
.		
209	11010001	J
210	11010010	K
211	11010011	L
212	11010100	M
213	11010101	N
214	11010110	O
215	11010111	P
216	11011000	Q
217	11011001	R
.		
.		
226	11100010	S
227	11100011	T
228	11100100	U
229	11100101	V
230	11100010	W
231	11100111	X
232	11101000	Y
233	11101001	Z
.		
.		
240	11110000	0
241	11110001	1
242	11110010	2
243	11110011	3
244	11110100	4
245	11110101	5
.		
.		
246	11110110	6
247	11110111	7
248	11111000	8
249	11111001	9
.		
.		
255	11111111	

Figure 83. EBCDIC Collating Sequence

C.2 ISCII/ASCII

Figure 84 shows the collating sequence for ISCII/ASCII, character, and unsigned decimal data. The collating sequence ranges from low (00000000) to high (01111111). Bit configurations that do not correspond to symbols are not shown.

Packed decimal, zoned decimal, fixed-point normalized floating-point data, and the signed numeric data formats are collated algebraically; that is, each quantity is interpreted as having a sign.

Collating Sequence	Bit Configuration	Symbol	Meaning
0	00000000		Null
32	00100000	SP	Space
33	00100001	!	Exclamation point
34	00100010	"	Quotation mark
35	00100011	#	Number sign
36	00100100	$	Dollar sign
37	00100101	%	Percent
38	00100110	&	Ampersand
39	00100111	'	Apostrophe, prime
40	00101000	(	Opening parenthesis
41	00101001	)	Closing parenthesis
42	00101010	*	Asterisk

43	00101011	+	Plus
44	00101100	,	Comma
45	00101101	-	Hyphen, minus
46	00101110	.	Period, decimal point
47	00101111	/	Slant
48	00110000	0	
49	00110001	1	
50	00110010	2	
51	00110011	3	
52	00110100	4	
53	00110101	5	
54	00110110	6	
55	00110111	7	
56	00111000	8	
57	00111001	9	
58	00111010	:	Colon
59	00111011	;	Semicolon
60	00111100	<	Less than
61	00111101	=	Equals
62	00111110	>	Greater than
63	00111111	?	Question mark
64	01000000	@	Commercial At
65	01000001	A	
66	01000010	B	
67	01000011	C	
68	01000100	D	
69	01000101	E	
70	01000110	F	
71	01000111	G	
72	01001000	H	
73	01001001	I	
74	01001010	J	
75	01001011	K	
76	01001100	L	
77	01001101	M	
78	01001110	N	
79	01001111	O	
80	01010000	P	
81	01010001	Q	
82	01010010	R	
83	01010011	S	
84	01010100	T	
85	01010101	U	
86	01010110	V	
87	01010111	W	
88	01011000	X	
89	01011001	Y	
90	01011010	Z	
91	01011011	[	Opening bracket
92	01011100	/	Reverse slash
93	01011101	]	Closing bracket
94	01011110	^	Circumflex, Logical NOT
95	01011111	_	Underscore
96	01100000	`	Grave Accent
97	01100001	a	
98	01100010	b	
99	01100011	c	
100	01100100	d	
101	01100101	e	
102	01100110	f	
103	01100111	g	
104	01101000	h	
105	01101001	i	
106	01101010	j	
107	01101011	k	
108	01101100	l	
109	01101101	m	
110	01101110	n	
111	01101111	o	
112	01110000	p	
113	01110001	q	
114	01110010	r	
115	01110011	s	
116	01110100	t	
117	01110101	u	
118	01110110	v	
119	01110111	w	
120	01111000	x	
121	01111001	y	
122	01111010	z	
123	01111011	{	Opening Brace
124	01111100	|	Vertical Line
125	01111101	}	Closing Brace
126	01111110	~	Tilde

Figure 84. ISCII/ASCII Collating Sequence

D.0 Appendix D. DFSORT/VSE Abend Processing

Subtopics:

D.1 Introduction to DFSORT/VSE Abend Processing

This appendix explains how DFSORT/VSE 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/VSE are system abend dumps that can be processed by standard dump analysis programs (for example, Info/Analysis program). A dump will be generated for DFSORT/VSE detected errors if the DUMP option is in effect.

| If the STXIT or MINSTXIT run-time option or the STXIT=YES or STXIT=MIN
| installation option is specified or defaulted, DFSORT/VSE saves the
| caller's STXIT and establishes its STXIT to trap system or user abends.
The following benefits are available:

- In general, you get a dump closer to the time of the abend.
- You get additional information useful in diagnosing the problem causing the abend.

| **Note:** If the MINSTXIT run-time option or the STXIT=MIN installation
| option is specified or defaulted, then it is possible that the E15/E35
| user exit routine's own STXIT may be in effect and DFSORT/VSE will not
| receive control. Therefore, DFSORT/VSE will not be able to do its
| abend recovery processing.

| At the end of its abend recovery processing, DFSORT/VSE returns control to
| the caller (that is, to the invoking program when DFSORT/VSE is program
| invoked or to the system when DFSORT/VSE is directly invoked) with a
| return code of 16 (unsuccessful completion) to allow a caller's recovery
| processing or termination to continue.

You can turn off the DFSORT/VSE STXIT routine by specifying the NOSTXIT operand of the OPTION control statement or the STXIT=NO installation parameter.

For more information on the Abend Processing, see the *VSE/ESA Solving Problems and Diagnosis Tools*.

D.2 Checkpoint/Restart

Checkpoint/Restart is a facility of the operating system that allows information about a 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.

To have DFSORT/VSE record checkpoints you must define a SORTCKP file and specify the CKPT operand of the SORT control statement. See "Defining Files" in topic 2.2 and "SORT Control Statement" in topic 3.17 for more information on the SORTCKP file and CKPT operand, respectively.

In general, checkpoints are not taken if any of the following conditions exist:

- Work files are not specified or not used.
- The application is a copy or merge.
- DFSORT/VSE is subtasked.
- A E31 user exit routine is specified in the MODS control statement. In this case, DFSORT/VSE will pass a device list to your E31 user exit so that you can take a checkpoint there.
- DFSORT/VSE executes in a partition that crosses 16 MB line. (Checkpoint/restart does not support the 31-bit environment.)
- DFSORT/VSE executes in a dynamic partition. (Checkpoint/restart does not support dynamic partitions.)

Data spaces used by DFSORT/VSE are not recorded because Checkpoint/Restart does not support data spaces.

For more information on the Checkpoint/Restart facility, see the *VSE/ESA System Macro Reference* and *VSE/ESA Guide to System Functions*.

D.3 DFSORT/VSE Abend Categories

There are two categories of abends for DFSORT/VSE:

- Unexpected abends

  These are system abends or user abends not issued by DFSORT/VSE.

- Abends issued by DFSORT/VSE under the following circumstances:

  - DFSORT/VSE's STXIT is in effect and DFSORT/VSE encounters an error that prevents normal completion of the application.
  - DFSORT/VSE detects an error in its internal logic.

D.4 Abend Recovery Processing for Unexpected Abends

DFSORT/VSE allows a user invoking program or user exit routine to control the use of DFSORT/VSE's STXIT routine, thus allowing other STXIT routines to have control instead of DFSORT/VSE's STXIT routine.

DFSORT/VSE normally has a STXIT routine established to trap system or user exit routine abends for sort, merge, and copy applications. If an abend occurs, the system will pass control to this routine. The DFSORT/VSE STXIT 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/VSE executes termination functions such as closing the files, issuing an error message, and so on.

- DFSORT/VSE returns control to the system at the end of its abend recovery processing so that system recovery routines can be invoked.

The DFSORT/VSE STXIT routine functions described above cannot be performed after an abend if the NOSTXIT operand of the OPTION control statement is specified. The DFSORT/VSE STXIT routine is not used in DFSORT/VSE processing if the NOSTXIT operand of the OPTION control statement is specified. If an invoking program or a user exit routine has established its own STXIT routine, it will be active during DFSORT/VSE processing. If the DUMP operand of the OPTION control statement is specified, a dump will be produced for DFSORT/VSE detected errors.

D.5 Processing of Error Abends with Error Messages

When DFSORT/VSE encounters a critical error, it issues an error message and terminates. You can specify that DFSORT/VSE is to terminate the application with an abend by using the DUMP operand of the OPTION control statement.

If abend termination is in effect and DFSORT/VSE encounters a critical error, DFSORT/VSE first causes an abend dump to capture the environment at the time of the error. Then, it issues the error message. It also runs the termination functions described earlier before terminating with an abend.

When the DUMP operand of the OPTION control statement is specified, the abend dump is produced after the error message is printed and other termination functions are run.

E.0 Appendix E. Locales Supplied with C Run-Time Library

| The following table lists the locales supplied with the LE/VSE 1.4 C

| Run-time library or VSE/ESA 2.3 C Run-time library. Consult your system programmer to determine whether they have been installed at your site.

For details regarding locales and their customization, see *LE/VSE Library*. The table of supported locales has been reproduced here for your convenience.

Throughout this book, references to LE/VSE also apply to the VSE C

| Language Run-Time Support feature of VSE/ESA Version 2 Release 3.

Figure 85. Compiled Locales Supplied with LE/VSE C

Locale name	Language	Country	Codeset	Phase name

Bg_BG.IBM-1025	Bulgarian	Bulgaria	IBM-1025	EDC$BGFE
Cs_CZ.IBM-870	Czech	Czechoslovaki	IBM-870	EDC$CZEQ
Da_DK.IBM-277	Danish	Denmark	IBM-277	EDC$DAEE
Da_DK.IBM-1047	Danish	Denmark	IBM-1047	EDC$DAEY
De_CH.IBM-500	German	Switzerland	IBM-500	EDC$DCEO
De_CH.IBM-1047	German	Switzerland	IBM-1047	EDC$DCEY
De_DE.IBM-273	German	Germany	IBM-273	EDC$DDEB
De_DE.IBM-1047	German	Germany	IBM-1047	EDC$DDEY
El_GR.IBM-875	Ellinika	Greece	IBM-875	EDC$ELES
En_GB.IBM-285	English	United Kingdom	IBM-285	EDC$EKEK
En_GB.IBM-1047	English	United Kingdom	IBM-1047	EDC$EKEY
En_JP.IBM-1027	English	Japan	IBM-1027	EDC$EJEX
En_US.IBM-037	English	United States	IBM-037	EDC$EUEA
En_US.IBM-1047	English	United States	IBM-1047	EDC$EUEY
Es_ES.IBM-284	Spanish	Spain	IBM-284	EDC$ESEJ
Es_ES.IBM-1047	Spanish	Spain	IBM-1047	EDC$ESEY
Et_EE.IBM-1122	Estonian	Estonia	IBM-1122	EDC$EEFD
Fi_FI.IBM-278	Finnish	Finland	IBM-278	EDC$FIEF
Fi_FI.IBM-1047	Finnish	Finland	IBM-1047	EDC$FIEY
Fr_BE.IBM-500	French	Belgium	IBM-500	EDC$FBEO
Fr_BE.IBM-1047	French	Belgium	IBM-1047	EDC$FBEY
Fr_CA.IBM-037	French	Canada	IBM-037	EDC$FCEA
Fr_CA.IBM-1047	French	Canada	IBM-1047	EDC$FCEY
Fr_FR.IBM-297	French	France	IBM-297	EDC$FFEM
Fr_FR.IBM-1047	French	France	IBM-1047	EDC$FFEY
Fr_CH.IBM-500	French	Switzerland	IBM-500	EDC$FSEO
Fr_CH.IBM-1047	French	Switzerland	IBM-1047	EDC$FSEY
Hr_HR.IBM-870	Croatian	Croatia	IBM-870	EDC$HREQ
Hu_HU.IBM-870	Hungarian	Hungary	IBM-870	EDC$HUEQ

Is_IS.IBM-871	Iceland	Iceland	IBM-871	EDC$ISER
Is_IS.IBM-1047	Iceland	Iceland	IBM-1047	EDC$ISEY
It_IT.IBM-280	Italian	Italy	IBM-280	EDC$ITEG
It_IT.IBM-1047	Italian	Italy	IBM-1047	EDC$ITEY
Iw_IL.IBM-424	Israeli	Israel	IBM-424	EDC$ILFB
Ja_JP.IBM-290	Japanese	Japan	IBM-290	EDC$JAEL
Ja_JP.IBM-930	Japanese	Japan	IBM-930	EDC$JAEU
Ja_JP.IBM-939	Japanese	Japan	IBM-939	EDC$JAEV
Ja_JP.IBM-1027	Japanese	Japan	IBM-1027	EDC$JAEX
Ko_KR.IBM-933	Korean	Korea	IBM-933	EDC$KRGZ
Lt_LT.IBM-1112	Lithuanian	Lithuania	IBM-1112	EDC$LTGD
Mk_MK.IBM-1025	Macedonian	Macedonia	IBM-1025	EDC$MMFE
Nl_BE.IBM-500	Dutch	Belgium	IBM-500	EDC$NBEO
Nl_BE.IBM-1047	Dutch	Belgium	IBM-1047	EDC$NBEY
Nl_NL.IBM-037	Dutch	Netherlands	IBM-037	EDC$NNEA
Nl_NL.IBM-1047	Dutch	Netherlands	IBM-1047	EDC$NNEY
No_NO.IBM-277	Norwegian	Norway	IBM-277	EDC$NOEE
No_NO.IBM-1047	Norwegian	Norway	IBM-1047	EDC$NOEY
Pl_PL.IBM-870	Polish	Poland	IBM-870	EDC$PLEQ
Pt_BR.IBM-037	Portugese	Brazil	IBM-037	EDC$BREA
Pt_BR.IBM-1047	Portugese	Brazil	IBM-1047	EDC$BREY
Pt_PT.IBM-037	Portugese	Portugal	IBM-037	EDC$PTEA
Pt_PT.IBM-1047	Portugese	Portugal	IBM-1047	EDC$PTEY
Ro_RO.IBM-870	Romanian	Romania	IBM-870	EDC$ROEQ
Ru_RU.IBM-1025	Russian	Russia	IBM-1025	EDC$RUFE
Sh_SP.IBM-870	Serbian	Serbia	IBM-870	EDC$SLEQ
	(Latin			

		alphabet)			
	Si_SI.IBM-870	Slovenian	Slovenia	IBM-870	EDC$SIEQ
	Sk_SK.IBM-870	Slovakian	Slovakia	IBM-870	EDC$SKEQ
	Sq_AL.IBM-1047	Albanian	Albania	IBM-1047	EDC$SAEY
	Sr_SP.IBM-1025	Serbian	Serbia	IBM-1025	EDC$SCFE
		(Cyrillic			
		alphabet)			
	Sv_SE.IBM-278	Swedish	Sweden	IBM-278	EDC$SVEF
	Sv_SE.IBM-1047	Swedish	Sweden	IBM-1047	EDC$SVEY
	Th_TH.IBM-838	Thai	Thailand	IBM-838	EDC$THEP
	Tr_TR.IBM-1026	Turkish	Turkey	IBM-1026	EDC$TREW
	Zh_CN.IBM-935	Chinese	China	IBM-935	EDC$ZCGY
		(simplified)			
	Zh_TW.IBM-937	Chinese	Taiwan	IBM-937	EDC$ZTGW
		(traditional)			

BACK_1 Summary of Changes for Previous Releases of DFSORT/VSE

Subtopics:

BACK_1.1 Third Edition, February 1997

Subtopics:

BACK_1.1.1 Programming Support for Release 3

Subtopics:

BACK_1.1.1.1 National Language Support

Cultural Sort and Merge: DFSORT/VSE 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/VSE 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.

DFSORT/VSE Messages Translation: DFSORT/VSE provides the ability for easy translation of messages into different languages. Flexible positioning of variables within a message is supported.

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. (Note that this support was first made available in DFSORT/VSE Version 3 Release 2.)

BACK_1.1.1.2 Productivity

Year 2000 Features: DFSORT/VSE's Year 2000 support will help you prepare for the turn of the century by correctly processing your two-digit year data. 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 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 (or zoned decimal) year data according to the century window using OUTREC (for example, transform 96 to 1996 and 00 to 2000).

A new PD0 format allows you to order parts of packed decimal fields, such as month and day in date fields, using SORT or MERGE.

New PZ, PSI and ZSI formats allow you to transform packed decimal and zoned decimal fields, such as month and day in date fields, to character fields using OUTREC.

New character and hexadecimal string separators allow you to insert literals in reformatted fields using OUTREC (for example, '/' in mm/dd/yyyy fields).

File Management Systems: New installation option FMS allows you to specify that DFSORT/VSE should attempt to take advantage of benefits provided by the File Management System installed at your site, such as:

° Dynamic logical and physical device assignment
° Dynamic primary and secondary extent allocation

INCLUDE and OMIT Enhancements: Do sophisticated filtering with new INCLUDE and OMIT features:

° Use the substring search capability to allow inclusion 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).

° Use bit level logic to allow inclusion of records based on:

  - Bit comparison tests of a binary field against a bit string constant. The bit string constant allows you to specify which bits of the input field must be on, which must be off and which can be ignored.

  - Bit operator tests of a binary field against a bit or hexadecimal mask. The mask allows you to test many different possible bit combinations with a single operation, similar to what you can do using the Test Under Mask (TM) machine instruction.

° Have unlimited levels of parentheses within logical expressions which allows you to create more complex conditions for inclusion of records.

SORT or MERGE Bit Control Fields: Use DFSORT/VSE's new bit control fields in conjunction with byte control fields to give you additional ways to sort or merge your data.

Input/Output Improvements: You can take advantage of the following input/output handling improvements:

° Copy up to nine input files to one output file.

° Process records and output them to a printer or punch device for sort, merge, or copy applications.

° Process input and output files on the IBM 3590 Tape Subsystems for sort, merge, and copy applications.

ICETOOL: ICETOOL is now more versatile as a result of enhancements to the existing operators. The improvements include:

° Allowing up to nine input files to be processed with the COPY, COUNT, DISPLAY, RANGE, STATS, and VERIFY operators.

° Allowing the active locale to be specified for the COPY, COUNT, and SORT operators, overriding the installation default for the active locale. Locale processing provides a way to SORT or COPY and INCLUDE or OMIT records according to the language and country rules defined in

the active locale.

BACK_1.1.1.3 Performance

Performance enhancements for DFSORT/VSE Version 3 Release 3 include the following:

- Dataspace sorting, introduced in DFSORT/VSE Version 3 Release 1 for fixed-length record sort applications, is now available for variable-length record sort applications.
- Improved Extended Count-Key-Data (ECKD) disk device support for work files.
- Improved data processing methods for:
  - Incore and non-incore dataspace sorting
  - Incore getvis sorting
  - Copy and merge applications
  - Input and output tape file processing for sort, merge, and copy applications.

BACK_1.1.1.4 Additional Enhancements

EQUALS/NOEQUALS: Preserve the original sequence of records that collate identically from input to output for merge applications using the EQUALS option.

Virtual Storage Constraint Relief (VSCR): Reduction in the number of DFSORT/VSE modules that are loaded into the 24-bit Shared Virtual Area (SVA) and partition program area.

HDR2: HDR2 support for tape output files.

BACK_1.2 Second Edition, October 1995

Subtopics:

BACK_1.2.1 Programming Support for Release 2

DFSORT/VSE Release 2 continues the strategy of providing performance improvements and productivity features. These improvements and features are described in more detail in the subsections that follow.

Subtopics:

BACK_1.2.1.1 Performance

With DFSORT/VSE Release 2, performance enhancements for dataspace sorting of fixed-length records and for getvis sorting of fixed-length and variable-length records include:

- Dynamic control of storage for dataspace sorting and getvis sorting
- Improved data processing methods for incore getvis sorting
- Improved data processing methods for non-incore getvis and dataspace sorting
- Reduced amount of work space required for SAM ESDS work files

---

BACK_1.2.1.2 Productivity

With DFSORT/VSE Release 2, your productivity is improved because you can:

- Create reports and analyze data using the new ICETOOL utility
- Select subsets of data using the STOPAFT and SKIPREC options
- Tailor DFSORT/VSE to suit your needs by using the new and modified ILUINST installation defaults and run-time options
- Access both introductory and reference information more quickly using a new DFSORT/VSE publication and a new DFSORT/VSE CD-ROM

BACK_1.2.1.3 ICETOOL

With ICETOOL, a versatile new DFSORT/VSE utility, you can do reporting and analysis of data at your site. ICETOOL allows you to perform multiple operations on one or more files in a single job step. This batch front-end utility uses the capabilities of DFSORT/VSE to perform the operations you request.

ICETOOL has thirteen operators: COPY, COUNT, DEFAULTS, DEFINE, DISPLAY, MODE, OCCUR, RANGE, SELECT, SORT, STATS, UNIQUE, and VERIFY. By using one operator or a combination of these operators, you can easily create applications that perform a variety of tasks including:

- Sorting input files to one or more output files
- Creating multiple copies of input files
- Creating output files containing subsets of input files based on various criteria
- Creating detailed reports allowing control of title, date, time, page numbers, headings, lines per page, field formats, and total, maximum, minimum, and average values
- Creating reports showing unique values for selected character and numeric fields and the number of times each occurs
- Creating reports or output files for records with: duplicate values, non-duplicate values, or values that occur n times, less than n times, or more than n times
- Creating a wide variety of reports using the preferred date, time, and numeric notations of individual countries
- Creating a report showing the DFSORT/VSE installation defaults in use
- 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

- Using three different modes, (stop, continue, and scan) to control error checking and actions after error detection for groups of operators

ICETOOL can be called directly or from a program. It 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/VSE messages produced as a result of an ICETOOL run, they are available if you need them.

BACK_1.2.1.4 SKIPREC and STOPAFT Options

With DFSORT/VSE Release 2, there are two new options that allow you to select subsets of data for sorting or copying. These options are:

- The SKIPREC option which enables you to specify the number of records you want to skip before starting to sort or copy the input files.

- The STOPAFT option which enables you to specify the maximum number of records you want accepted for sorting or copying.

BACK_1.2.1.5 Installation Defaults and Run-time Options

Several ILUINST installation defaults have been added or changed providing you with more flexibility in using installation defaults and run-time options. These defaults include:

- GVSRLOW and GVSRANY which enable you to specify the reserved partition GETVIS area for user application requirements.

- WRKSEC which enables you to allow or suppress the dynamic secondary allocation for SAM ESDS work files.

- STORAGE which may be specified in megabytes.

To give you even more flexibility on a daily basis, several run-time options have been added or changed. These run-time options include:

- WRKSEC and NOWRKSEC which enable you to override the installation defaults.

- GVSRLOW and GVSRANY which enable you to override the installation defaults.

- STORAGE which may be specified in megabytes.

- CKPT which enables you to restart DFSORT/VSE when the DFSORT/VSE modules are placed in SVA.

BACK_1.2.1.6 Additional Enhancements

Additional changes that you will find helpful are:

- An extended DFSORT/VSE parameter list feature which enables you to supply the *end of parameter* list indicator.

- Tagging of the error position in an invalid parameter list control statement image.
- The END control statement which enables you to discontinue accepting control statements.
- Improved work file processing through internal enhancements to the SAM ESDS work files secondary allocation algorithm which allows you to use multivolume SAM ESDS work files.
- Improved system and application availability with additional program modules placed above 16MB virtual.

With DFSORT/VSE, Release 2, a new book is included in the DFSORT/VSE library, *Getting Started with DFSORT/VSE*, SC26-7101. This book gives you all of the information and instructions you need to start using the features of DFSORT/VSE.

Also new with this release is the *DFSORT/VSE Online Product Library*, SK2T-8730. This CD-ROM contains all of the books in the DFSORT/VSE library except for the *DFSORT/VSE Reference Summary* and the *DFSORT/VSE Licensed Program Specifications*.

BACK_1.3 First Edition, September 1994

Subtopics:

BACK_1.3.1 Programming Support for Version 3 Release 1

Subtopics:

BACK_1.3.1.1 Performance

Enhancements include:

- Dataspace sorting, a DFSORT/VSE capability that uses data space available with VSE/ESA 1.3 in supervisor MODE=ESA in place of intermediate work space for fixed-length record sort applications.
- Getvis sorting, a DFSORT/VSE capability that uses main storage above and below 16 MB virtual for fixed-length and variable-length record sort applications.

BACK_1.3.1.2 31-bit Addressing Support

ESA/370 and ESA/390 technology in VSE/ESA 1.3 provides the following enhancements for DFSORT/VSE:

- 31-bit addressing support for a calling program and user exit

routines.

° DFSORT/VSE phases can be loaded into 31-bit SVA.

° DFSORT/VSE and VSAM working storage can be located above 16 MB virtual.

° Reduction in the use of work files by using partition storage above 16 MB virtual with getvis sorting.

BACK_1.3.1.3 Other Enhancements

Secondary allocation for SAM ESDS work files is supported with the VSE/VSAM Space Management for SAM Feature installed.

Several ILUINST installation and run-time options have been added:

° DSPSIZE, a new operand on the OPTION control statement, controls the use of dataspace sorting for fixed-length record sort applications.

° GVSIZE, a new operand on the OPTION control statement, controls the use of getvis sorting for fixed-length and variable-length record sort applications.

° STXIT or NOSTXIT, new operands on the OPTION control statement, control the use of DFSORT/VSE's STXIT routine for sort, merge, and copy applications.

° FIELDS=NONE, a new operand on the SUM control statement, allows the elimination of duplicate records with equal control fields, without summing, for sort and merge applications.

BACK_1.3.1.4 Operating System Environment

DFSORT/VSE Version 3 Release 1 can run only in the VSE/ESA environment.

DFSORT/VSE Version 3 Release 1 does not support the CMS/DOS environment of VM/ESA. DFSORT/CMS 5684-134 Version 2 Release 1 is available for the CMS environment to satisfy user's needs for a sort product.

INDEX Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Y

Z

BACK_2 Communicating Your Comments to IBM

DFSORT/VSE
Application Programming Guide
Version 3 Release 4

Publication No. SC26-7040-03

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

° If you prefer to send comments by mail, use the RCF at the back of this book.

° If you prefer to send comments by FAX, use this number:

- United States: 1-800-426-6209
- Other countries: (+1)+408+256-7896

° If you prefer to send comments electronically, use this network ID:

- IBMLink from U.S. and IBM Network: STARPUBS at SJEVM5
- IBMLink from Canada: STARPUBS at TORIBM
- IBM Mail Exchange: USIB3VVD at IBMMAIL
- Internet: starpubs@vnet.ibm.com

Make sure to include the following in your note:

° Title and publication number of this book
° Page number or topic to which your comment applies.

COMMENTS Readers' Comments -- We'd Like to Hear from You

DFSORT/VSE
Application Programming Guide
Version 3 Release 4

Publication No. SC26-7040-03

Overall, how satisfied are you with the information in this book?

Legend:

1 Very satisfied
2 Satisfied
3 Neutral
4 Dissatisfied
5 Very dissatisfied

	1	2	3	4	5
Overall satisfaction					

How satisfied are you that the information in this book is:

	1	2	3	4	5
Accurate					
Complete					
Easy to find					
Easy to understand					
Well organized					
Applicable to your tasks					

Please tell us how we can improve this book:

International Business Machines Corporation
RCF Processing Department
M86/050
5600 Cottle Road
SAN JOSE, CA 95193-0001

Name . . . . . _____
Company or Organization _____
Address . . . . . _____
_____
_____
Phone No. . . . . _____

IBM Library Server Print Preview

DOCNUM = SC26-7040-03

DATETIME = 04/23/98 10:42:53
BLDVERS = 1.3.0
TITLE = DFSORT/VSE Application Programming Guide
AUTHOR =
COPYR = © Copyright IBM Corp. 1976, 1998
PATH = /home/webapps/epubs/htdocs/book