# OS/360 Sort/Merge for MVS 3.8

# Application Programming Guide

# Version 1.03

**March 22, 2024**

# Contents

# Figures

# Tables

# Preface

## Acknowledgements

This project to refurbish and upgrade the Sort/Merge Program distributed with OS/360 Release 21 to the MVS 3.8 Operating System environment has been work in progress for some years. The project was suspended after completion of the initial investigation and scoping work revealed the size and complexity of the project to refurbish and upgrade the code. However, with the development and the free contribution of suitable tools required for such a complex project, it was time to recommence this project.

The tools that enabled this project to be completed were Review developed by Greg Price and DDT developed by Shelby Beach. Editing in excess of 360 source members with 88,700 lines of source code was a sizeable task and Review performed this without error or loss of any data. Debugging the code, without using DDT, would have been an extremely difficult and a very time consuming task. The Sort/Merge Program consists of a large number of modules with very tight, non-standard, interaction between modules dynamically loaded at run time. A very powerful and fully featured debugging program was essential to debug such an environment. During the course of this project Shelby Beach has twice made major enhancements to DDT, providing additional features to further assist in debugging code in complex environments. I thank both developers for freely contributing their excellent products. I hold their products in highest regard as essential tools for software development.

The refurbishment and upgrade process introduced a significant number of new features. New documentation was required to describe the new features. I gratefully acknowledge the contribution made by Peter Glanzmann towards the preparation of the documents supporting this new version of the Sort/Merge Program. Peter Glanzmann kindly contributed the document styling, font and patterns for the railroad syntax diagrams used extensively throughout the documents. Peter's contribution also included the contents of Appendix C describing the syntax used to document the format and parameters of the control statements.

The new features introduced into the Sort/Merge Program also required thorough testing to ensure that they would run successfully in many different configurations and environments. I am indebted to Phil Roberts for taking the beta release of the Sort/Merge Program and testing it with many different sorting tasks ranging from simple sorts to extensive stress testing at maximum configurations and to generate data for performance estimation purposes. Phil also ran compatibility testing of the Sort/Merge Program for MVS 3.8 against the previous Sort/Merge Program for OS/360 Release 21 to confirm upward compatibility had been preserved. The feedback from Phil's testing has resulted in improvements in usability, improved documentation and a more robust Sort/Merge Program.

I thank all those who have helped me complete this project.

## What this document is about

This document describes the use of the OS/360 Sort/Merge Program for MVS 3.8.

It discusses:

- Program capabilities
- Sorting and Merging techniques
- Sort/Merge program control statements
- Intermediate storage requirements
- Job Control Language requirements
- Program Initiation
- Use of Program Exits
- Efficient Program Use
- Sort/Merge Program messages.

# Who should read this document

This document is intended for users who wish to sort and merge data using the OS/360 Sort/Merge Program for MVS 3.8. It assumes a basic knowledge of sorting and provides a task oriented guide and reference to sorting and merging using the OS/360 Sort Merge Program for MVS 3.8.

# What you need to know to understand this document

To use this document effectively the user should be familiar with the following information:

- Job Control Language
- Data Management and record formats
- DASD and Tape hardware.

In addition, familiarity with the information in the following documents would be of benefit:

- MVS JCL User's Guide
- MVS JCL Reference
- MVS Data Management Services Guide.

# How to use this document

This document is a guide and reference for users of the OS/360 Sort/Merge Program for MVS 3.8. It contains a general description of the Sort/Merge Program and specific information about control statement formats, program operation, the inclusion of user-written routines, efficient use of the Sort/Merge Program, and program generated messages. Merging techniques used by the Sort/Merge Program are briefly described. The various chapters of this document describe the functionality of the OS/360 Sort/Merge Program for MVS 3.8.

## Chapter 1: Introduction

This chapter describes sorting and merging specifications, control fields, sorting and merging techniques used by the Sort/Merge Program, and error correction facilities.

## Chapter 2: How to Use the Sort/Merge Program

This chapter is divided into four main topics:

Defining the Sort or Merge which describes the use of SORT or MERGE control statements to control the sorting or merging operation.

The general format of the Sort/Merge Program's control statements.

Detailed information describing the parameters for each of the Sort/Merge Program's control statements with examples of their usage.

Determining Intermediate Storage Requirements that describes how to calculate the amount of intermediate storage for a given sorting application.

## Chapter 3: Invoking the Sort/Merge Program with JCL Language

This chapter provides information on the required Job Control Language statements that describe the JOB, EXEC, and DD statements necessary for Sort/Merge operation and contains a number of complete JCL and Sort/Merge control statement examples.

## Chapter 4: Invoking Sort from a Program

This chapter describes initiating sorting operations via a macro instruction in another program.

## Chapter 5: Using Exit Routines

This chapter describes Sort/Merge Program exits and the requirements for user-written routines that use them. Users who do not include their own routines to modify records or handle errors during the running of the Sort/Merge Program will not need to reference this chapter.

## Chapter 6: Efficient Program Use

This chapter describes the factors that contribute to efficient use of the Sort/Merge Program.

### Appendix A

Contains a summary of how to use the Sort/Merge Program.

### Appendix B

Contains a program message directory with explanatory text for each message and a suggested user response to each message.

### Appendix C

Contains a description of the syntax used to describe the format of the Sort/Merge control statements and their associated parameters.

## Related Documents

The document, OS/360 Sort/Merge for MVS 3.8 Installation, Customization and Diagnosis Guide is a guide and reference for users responsible for the installation, customization and support of the OS/360 Sort/Merge Program for MVS 3.8. It contains a step by step guide on how to install the OS/360 Sort/Merge Program for MVS 3.8 in the MVS 3.8 environment. A full description of the customization options are provided together with a number of Installation Verification Programs designed to prove the successful installation and customization of the Sort/Merge Program at the installed site. The provided Installation Verification Program job streams can also be of benefit to use as examples for user sorting applications.

# Summary of Changes

OS/360 Sort/Merge for MVS 3.8 is a major enhancement from the version distributed with OS/360 Release 21. It has many new features and enhancements.

## New Information for this release

### All DASD unit types supported

The Sort/Merge Program now supports intermediate storage on all DASD unit types that are supported by MVS 3.8. The geometry and track capacity of all DASD unit types are recognized and utilized. There is a restriction for DASD unit types that have a track capacity greater than 32,767 bytes. For these DASD unit types the Sort/Merge Program is restricted to a maximum of half-track blocking for storage of intermediate data. Using DASD unit types with a large track capacity is recommended for improved sorting efficiency and reduced resource usage.

### Increased maximum sort record length

The maximum length record that can be sorted is increased from the previous limit of approximately 7200 bytes to an approximate maximum of 27,900 bytes. This is achieved by selecting, for use as intermediate storage, DASD unit types with a large track capacity.

### Intermediate Storage data sets can be allocated in Extents

DASD SORTWKdd data sets used for storing intermediate data during sorting operations can now be allocated in extents. The Sort/Merge Program will not cause a SORTWKdd data set to be extended if the initial allocation was insufficient however any allocated extents will be used if the space is needed.

### No restriction on the location of Intermediate Storage data sets on a DASD Volume

The prior restriction that the single contiguous extent of each SORTWKdd data set be allocated entirely within the first 256 cylinders of a DASD volume has been removed.

### Dynamic Allocation of Intermediate Storage data sets

SORTWKdd data sets used for intermediate storage can now be dynamically allocated by the Sort/Merge Program. The DASD space allocated for each data set will be calculated by the Sort/Merge Program based on information provided on control statements, the size of the input data set and installation set defaults. The DASD unit type selected for allocation can be specified for each sort operation or default to an installation defined unit type.

### New OPTION Control Statement

A new OPTION control statement is provided to enable the user to specify additional Sort/Merge Program settings.

### New DEBUG Control Statement

A new DEBUG control statement is provided to control the settings of the various tracing options and issuance of diagnostic messages that can be of assistance in problem resolution.

### Control Statements not case sensitive

All Sort/Merge Program control statements can now be entered in upper or lower case characters.

### Labels on Control Statements

Labels are supported on all Sort/Merge Program control statements.

### Text for all Sort/Merge messages revised and improved

The text of all Sort/Merge Program messages have been extensively revised and improved with the objective of reducing the need to consult the documentation to resolve a problem with the preparation of control statements or the operation of the Sort/Merge program.

## Optional listing of Control Statements

Control statements provided to the Sort/Merge Program can now be optionally listed on the user-selected message facility, being the message data set, the Job Log or console.

## Abend Code can be message number or value

In situations where the Sort/Merge Program must terminate a user-determined value can be used as the user abend code or, alternatively the user abend code can be set to the critical message number identifying the reason for the termination.

## Increased Storage Requirements

The storage requirements for the Sort/Merge Program have increased significantly. This is primarily due to the use of large track capacity DASD types for intermediate data storage. The recommended storage for a sorting operation using the Sort/Merge Program's CRCX sequencing technique and 3390 DASD is approximately 512 KB.

## Additional Installation parameters for the control of storage utilization

Options set at installation customization time can be used to control the Sort/Merge Program's utilization of storage. Additional parameters are provided to ensure there is sufficient storage left available for program invoked sort operations.

## Enhanced parameter list for ATTACH/LINK/XCTL invoked sorting operations

Additional control statements and control parameters can be passed to the Sort/Merge Program when it has been invoked by ATTACH/LINK/XCTL providing increased control over sorting operations.

## Enhanced E15 and E35 Parameter List

A new user address constant is passed to both the E15 and E35 user exits. The initial value of the user address constant can be set in the enhanced parameter list for ATTACH/LINK/XCTL invoked sorting operations.

## Override of ATTACH/LINK/XCTL invoked Sort Control Statements

For sorting operations that have been invoked by a user program via ATTACH/LINK/XCTL, the user program provided control statements passed to the Sort/Merge Program can be overridden by providing control statements in the SORTCNTL data set.

## Override of all Control Statements

All options set by control statements in the SYSIN stream, the SORTCNTL stream if program invoked, or passed to the Sort/Merge Program can be overridden by control statements in the IERPARM input stream.

## STOPAFT

A new parameter, STOPAFT, can be coded on either the SORT or the OPTION control statement to limit the number of records read into a sorting operation.

## Instruction path length reduction

The MVCL instruction is used for internal record movement, in place of MVC loops, when the length of records being sorted is greater than 768 bytes. The code, in many modules, has also been optimized at the local level by use of System/370 instructions to reduce path length.

# Operational Changes that may Require User Action

The following are operational changes that may require user action for existing sorting applications that use certain functions:

## Change to the E35 Exit Parameter List

The third word of the E35 exit parameter list has been repurposed as the user address constant. Prior to this release the third word of the E35 exit parameter list was used to control sequence checking of records leaving the sorting operation on a record by record basis. This function is now controlled, for the entire sorting operation, by the VERIFY/NOVERIFY installation parameter or overridden, for each sorting operation, by the VERIFY/NOVERIFY parameter on the OPTION control statement. E35 exits that use the record by record control of sequence checking will require revision to operate correctly with this release.

# 1.   Introduction

This document explains how to use the Sort/Merge Program for MVS 3.8 to fulfill the sorting and merging requirements for MVS 3.8 users.

The Sort/Merge Program can arrange records in a data set into a predesignated order. The records in a data set are placed in a sequence according to the contents of user-specified fields present in each record. The Sort/Merge Program is generalized to perform a variety of sorts and merges. Because of this ability, the Sort/Merge Program can simplify many data processing applications that require the sequential updating of previously created data sets.

Input to sorting or merging operations and output from sorting or merging operations can be any data set that consists of fixed-length or variable-length, blocked or unblocked records (except Undefined format). All input and output data sets must be able to be read or written by the queued sequential access method (QSAM). Any I/O device that operates with QSAM can be used for input and output.

Sorting and merging techniques are used that take advantage of the hardware configuration and data set sizes. These techniques are designed to provide efficient operation for a great variety of different sorts and merges. The technique selected for use with a specific sorting or merging application depends upon information supplied through control statements that define the application to be performed. The control statements can be supplied to the Sort/Merge Program in the Operating System input stream or as parameters passed by another program.

User-written exit routines can operate in conjunction with the Sort/Merge Program to perform many functions while the Sort/Merge Program is running. Control is passed to user-written exit routines at various stages during the operation of the Sort/Merge Program. When the user exit routines receive control, the routines can insert, summarize, discard, and alter the records being sorted or merged.

## 1.1   Relationship to the Operating System

The Sort/Merge Program operates under the supervisory control of the MVS 3.8 Operating System. A Sort/Merge Program operation must be initiated according to MVS 3.8 conventions. All data sets used by the Sort/Merge Program must be defined according to MVS 3.8 standards.

The Sort/Merge Program makes extensive use of MVS 3.8 data management facilities and system services. All data sets necessary for program operation must be defined in JCL data definition (DD) statements except where the Dynamic Allocation facility is used to allocate intermediate storage on DASD unit types. Required JCL DD statements must be placed in the input stream with the job step that initiates the Sort/Merge Program's operation. JCL DD statements are described in the document MVS Job Control Language.

The Sort/Merge Program can be tailored to the needs of a particular installation when the Sort/Merge Program is initially installed and customized. The customization process can be rerun if changes need to be made to the installation configuration options.

## 1.2   Minimum Machine Requirements

### Storage Requirements

Sorting performance usually improves as the amount of storage available for use increases. Approximately 512 KB of storage are required for efficient operation with a large number of records and use of 3390 DASD for intermediate storage. Refer to Chapter 6: Efficient Program Use for more information about storage requirements.

Sort size is the amount of storage assigned to the Sort/Merge Program at installation customization time. If the user overrides the installation value at run time, then the overriding value is used for the sort storage size.

## Intermediate Working Storage Requirements

The amount of intermediate working storage needed to perform sorting applications depends upon the size of the input data set. Working storage can be allocated on either DASD or magnetic tape units.

The amount of storage available and the DASD unit type allocated to intermediate working storage effects the maximum record size and the number of the records that can processed. Table 1 shows the approximate maximum record size that can be sorted for a given DASD unit type with either fixed or variable-length records. The minimum record length, using DASD or tape for intermediate storage is 18 bytes. Conditions such as control field extraction, or large numbers of intermediate storage data sets require additional storage. As a work area is used for VBS format records, the available storage for buffers and sorting is decreased. In general, the largest record length that can be sorted is approximately equal to the largest record that can fit on the selected intermediate DASD unit type minus the internal block overhead which can be up to 28 bytes in length. For DASD unit types with a track capacity greater than 32,767 bytes then the maximum length of records able to be sorted is reduced to approximately half the track capacity minus the internal block overhead.

Table 1 DASD Unit Type – Maximum Record Length

| DASD Type | Approximate Max Record Length |
|---|---|
| 2314 | 7,200 |
| 3330 | 12,600 |
| 3340 | 8,100 |
| 3350 | 18,900 |
| 3375 | 17,520 |
| 3380 | 23,400 |
| 3390 | 27,900 |

# 1.3   Control Fields

Each record in a data set is sorted or merged on the basis of data contained in the record's control fields. A control field which can be up to 256 bytes long, and there can be 1 to 64 control fields. Control fields can overlap, the end of one control field can share data with the beginning of another control field. Figure 1 shows a record with four control fields defined.
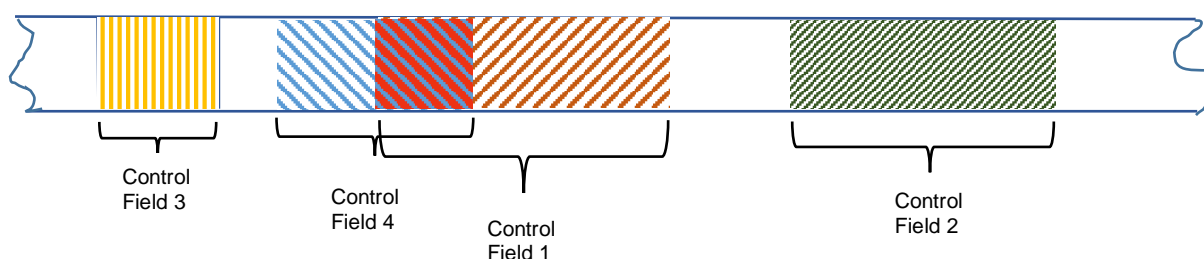


Figure 1 Sort with Four Control Fields

Each control field, along with the record in which it appears, is sorted into either ascending or descending order, using standard collating sequences. The collating sequence for character data and binary data is absolute; that is, character and binary fields are not interpreted as having signs. For packed decimal, zoned decimal, fixed point, and normalized floating-point data, collating is algebraic; that is, each quantity is interpreted as having an algebraic sign.

Nonstandard sequencing can be achieved without physically changing the control fields. A user-written exit routine can modify one or more of the control fields each time the Sort/Merge Program sequences a record. The modified control fields are used for sequencing purposes only; they do not replace the fields in the records. User-written routines can be invoked at Sort/Merge Program exit points. These exits and the requirements for user-written routines that use them are discussed in Chapter 5: Using Exit Routines.

Control field lengths for the various control field data formats accepted by the Sort/Merge Program are:

- Character, fixed-point, or normalized floating point data - 1 through 256 bytes.
- Packed or zoned decimal data - 1 through 16 bytes.
- Binary data - 1 bit through 256 bytes.

Control fields must be contained within the first 4,092 bytes of a record.

# 1.4   Sort Requirements

Control fields for a sorting application are defined in a SORT control statement such as

SORT FIELDS=(10,30,A),FORMAT=CH

described in Chapter 2: How to Use the Sort/Merge Program. Input, output, and intermediate storage data sets are defined on standard Job Control Language DD statements such as

```
//SORTOUT DD DSN=PROD.OUTPUT,UNIT=3390,DISP=(NEW,CATLG),
//  DCB=(RECFM=FB,LRECL=90,BLKSIZE=900),SPACE=(TRK,10)
```

described in Chapter 3: Invoking the Sort/Merge Program with JCL Language.

## Input

Input records can be blocked or unblocked records in a sequential data set containing fixed or variable-length records on any I/O device that is supported by QSAM for read operations.

## Output

Output records can be written to a blocked or unblocked sequential data set containing fixed or variable-length records. The output device can be any device that is supported by QSAM for write operations and need not be related in any way to the input device.

## Intermediate Storage

All intermediate storage used for a specific sorting operation must be allocated to the same unit type. Up to 32 tape units or up to 17 DASD data sets can be used for intermediate storage. The amount of intermediate storage required is based primarily on the number of records and the length of the records in the input data set. The amount of storage available to the Sort/Merge Program is also a factor in determining intermediate storage requirements. Intermediate storage is discussed in greater detail and formulas for the amount of storage needed are given in Chapter 2 in the topic Determining Intermediate Storage Requirements.

## User Modifications

User-written exit routines can summarize, insert, discard, shorten, lengthen, or otherwise alter records while the records are being sorted. A detailed discussion of exits from the Sort/Merge Program that provide for control to be transferred to user-written routines is given in Chapter 5: Using Exit Routines.

## Invoking the Sort

A sorting operation is initiated by control statements in the Operating System job input stream, or by another program through the use of an ATTACH, LINK, or XCTL Operating System service.

## 1.5   Merge Requirements

Control fields for a merging application are defined in a MERGE control statement. An example being

MERGE FIELDS=(10,30,A),FORMAT=CH

described in Chapter 2: How to Use the Sort or Merge Program. Input and output data sets are defined by standard Job Control Language statements. An example being

//SORTIN01 DD DSN=APPL.PARTDATA.YEAR2015,DISP=SHR
//SORTIN02 DD DSN=APPL.PARTDATA.YEAR2016,DISP=SHR

described in Chapter 3: Invoking the Sort/Merge Program with JCL Language.

### Input

Input to a merging operation can be up to 16 blocked or unblocked sequential data sets containing fixed or variable-length records. For a given application all records must be of the same format, only the blocking factor can differ. The records in the input data sets must be in proper sequence as defined by the MERGE control statement. The input devices must be supported by QSAM for read operations.

### Output

Output from a merging operation can be a blocked or unblocked sequential data set containing fixed or variable-length records. The output device must be supported by QSAM for write operations and need not be related in any way to the input device type.

### Intermediate Storage

Not needed for a merge only operations.

### User Modification

Exit points are provided for user-written routines to summarize, insert, discard, lengthen, shorten, or otherwise alter output records. A detailed discussion of these exits and the requirements for routines that use them is given in Chapter 5: Using Exit Routines.

### Invoking the Merge

A merge operation can only be initiated by control statements in the Operating System input stream.


## 1.6   Sorting Techniques

The input data set is almost always too large to be brought into storage and sorted all at once. Instead, it is broken up into sections. Each section is placed in sequence and stored on an intermediate storage device. The sorted sections of the input data set are called sequence sets.

### Sequence Distribution Techniques

The Sort/Merge Program selects one of five sequence distribution techniques based on the information it has been provided about a specific sorting application. The objective of all five techniques is to enable the intermediate merge phase of the sorting operation to combine the many small sequences of records produced by the sort phase into a few longer sequences. The number of sequences must be reduced to the point where the final merge phase can combine all the sequences into a single sequence in one pass.

### Tape Techniques

If the intermediate storage medium is tape, the balanced tape technique, the polyphase tape technique, or the oscillating tape technique can be candidates for selection.

## DASD techniques

If the intermediate storage medium is DASD, the balanced direct access technique (BALN) or the crisscross direct access technique (CRCX) can be candidates for selection.

## DASD techniques intermediate storage considerations

To use the BALN sequencing technique at least three intermediate storage data sets are required with a maximum number of six intermediate storage data sets. For the CRCX technique, which is recommended for large sorting operations, at least six intermediate storage data sets are required, with a maximum of 17 intermediate storage data sets. With both the BALN and CRCX sequencing techniques it is more efficient to use the minimum number of intermediate storage data sets for each technique, three for BALN and six for CRCX, as less storage is required for input/output buffers leaving more storage available for internal record storage. Depending on the number of records being sorted, the length of the records being sorted and the capacity of a volume of the DASD unit type selected for intermediate storage it may not be possible to use the minimum number of data sets to provide the required amount of intermediate storage. In that case an increased number of intermediate storage data sets must be provided to ensure there is sufficient intermediate storage allocated to complete the sorting operation.

Note that if six intermediate storage data sets are provided then the Sort/Merge Program will always select the BALN sequencing technique. The CRCX sequencing technique will be automatically selected if seven or more intermediate storage data sets are provided. The CRCX sequencing technique can be used with six intermediate storage data sets only if the CRCX sequencing technique is forced. If the CRCX sequencing technique is the preferred sequencing technique then providing seven intermediate storage data sets is more convenient than forcing the CRCX technique with only six intermediate storage data sets. Forcing a sequence technique is discussed in the next section.

Both the BALN and CRCX sequencing techniques benefit if the assigned intermediate storage data sets are all of the same size.

## 1.7 Forcing a Sequence Distribution Technique

If, for a particular sorting application, the Sort/Merge Program does not select the most efficient technique, then specify the use of another technique. The specified technique will only be used if there is enough storage and the appropriate number of intermediate work areas have been provided to satisfy the technique's requirements. Table 2 lists the specific requirements and limitations for each technique.

If the specific requirements for a technique are not satisfied then another technique will be selected rather than the sorting operation being terminated.

Caution is recommended when deciding to force a technique. The Sort/Merge Program attempts to select the most efficient technique for a given sorting application. If it is forced to use another technique, performance is usually not as efficient.

For information on how to force the selection of sequence distribution technique refer to the discussion of the EXEC statement PARM field in Chapter 3: Invoking the Sort/Merge Program with JCL Language or Chapter 4: Invoking Sort from a Program if the sorting operation is being invoked by another program.

## Requirements and limitations for each sequence distribution technique

Table 2 Sequence Distribution Technique Requirements

| Technique | Maximum Sort Input Allowed | Required Minimum Number of Intermediate Storage Areas | Maximum Number of Intermediate Storage Areas Allowed | Comments |
|---|---|---|---|---|
| Balanced DASD BALN | No fixed maximum. Limited by intermediate storage | 3 areas | 6 areas | Always used when less than 6 work areas are available. Used when 6 areas are available unless CRCX is forced. |
| Crisscross DASD CRCX | No fixed maximum. Limited by intermediate storage | 6 areas | 17 areas | Always used when more than 6 areas are available. Can be forced for 6 areas. |
| Balanced Tape BALN | 15 Volumes | $2(x+1)$ where x is the number of input volumes | 32 Tape Units | Always used if there are more than 3 intermediate storage tapes available and the input data set size is not specified or estimated. |
| Polyphase Tape POLY | 1 Volume | 3 Tape Units | 17 Tape Units | Always used if only 3 intermediate storage tapes are available. |
| Oscillating Tape OSCL | 15 Volumes | X+2 or 4, whichever is greater where x is the number of input volumes | 17 Tape Units | Input data set size must be given or closely estimated. The tape drive containing SORTIN cannot be assigned as an intermediate storage unit. |

## Error Correction Facilities

The Sort/Merge Program provides exits where control can be transferred to user-written routines. Refer to Chapter 5: Using Exit Routines. These user written routines might be able to correct:

- I/O errors that cannot be corrected by the Operating System.
- Errors that arise because the input data set is larger than the intermediate storage capacity estimated by the Sort/Merge Program for a given application.

## I/O Errors

The Sort/Merge Program passes control to a user-written I/O error routine only when the Operating System cannot correct the error condition. In the case of a permanent read error the user-written routine can:

- accept the block as is,
- attempt to correct the error,
- skip the block, or,
- request termination.

For an uncorrectable write error, the user-written exit routine can perform any necessary abnormal end-of-task operations before the sorting or merging operation is terminated.

If no user-written exit routines are supplied, the Sort/Merge Program issues the message IER061A I/O Error with details identifying the error. The sorting or merging operation then terminates.

## Exceeding Intermediate Storage Capacity

The Sort/Merge Program estimates a maximum intermediate storage capacity (Nmax) from the information supplied to it at the beginning of the sorting operation.

An actual or an estimated input data set size can be supplied by a control statement. This is done via the SIZE parameter on the SORT control statement described in Defining the Sort or Merge in Chapter 2. If an actual data set size is supplied, and the size is greater than Nmax, the sorting operation is terminated. If an estimated data set size is supplied, or if no data set size is supplied, and the number of records processed while sorting reaches Nmax, then control is given to a user-written Nmax user exit, if one is supplied. The Nmax user exit can take one of the following actions:

- Indicate that sorting should continue with processing the entire input data set using the available intermediate storage. If the estimated input data set size was high, there might be enough intermediate storage left to complete the application.
- Direct that sorting is to continue with only that part of the input data set that has been already been read into the sorting operation. The remainder of the data set could be sorted later and the two results merged to complete the application.
- Terminate sorting without any further processing.

If an Nmax user exit is not supplied then processing continues to read records beyond the calculated Nmax value. If the intermediate storage capacity is sufficient to contain all the records in the input data set, the sorting operation completes normally. When the allocation of intermediate storage is not sufficient, the sorting operation is terminated.

A separate message is generated for each of the three possible error conditions. These messages are:

IER041A SIZE parameter is greater than Estimated Maximum Records

> Generated before sorting begins when the exact data set size supplied on a SORT control statement is greater than Nmax.

IER046A Sort capacity exceeded

> Generated when all available intermediate storage has been used and additional records still remain to be processed.

IER048I Estimated maximum record capacity exceeded

> Generated when the number of records read in has exceeded Nmax and control has been transferred to a user-written Nmax exit for further action.

A full description and explanation of all messages is contained in Appendix B.

# 2. How to Use the Sort/Merge Program

There are three basic things that must be done to use the Sort/Merge program:

1. Define the sorting or merging job with Sort/Merge control statements. See Defining the Sort or Merge in this chapter.
2. If the application is a sort, determine the amount of intermediate storage the input data will require while it is being sorted and merged. See Determining Intermediate Storage Requirements in this chapter.
3. Prepare Job Control Language statements for the job and combine them in proper order with the Sort/Merge control statements. See Chapter 3: Invoking the Sort/Merge Program with JCL Language for further information on this topic.

## 2.1 Defining the Sort or Merge

The Sort/Merge Program must be provided with instructions on how the input data is to be processed. A general description of the input data is needed together with information about the control fields in the input records, and a description of modification routines, if any, that will be used during a sort or merge operation. Sort/Merge control statements are used to supply the required information.

The Sort/Merge Program control statements are:

### SORT Statement

Provides information about control fields and data set size. Use this statement for sorting applications. Do not use this statement for merging applications.

### MERGE Statement

Provides the same information as a SORT statement. Use this statement for merging applications. Do not use this statement for sorting applications.

### RECORD Statement

Provides both record length and record type information. This statement is required only when modification routines change record lengths during a sort or merge operation or for program invoked sorting operations.

### MODS Statement

Associates user-provided modification routines with specific Sort/Merge Program exits. This statement is required only when modification routines to be invoked at exits points. Chapter 5: Using Exit Routines describes these exits and the requirements for routines that use them.

### OPTION Statement

Provides the ability to override of some of the default options set at installation customization time such as CHECK and VERIFY and to provide other optional information. Some of the options available can also be provided on the Sort or Merge statement.

### DEBUG Statement

This statement is not intended for regular use. For further information refer to the related document OS/360 Sort/Merge for MVS 3.8 Installation, Customization and Diagnosis.

### END Statement

Signifies the end of a related group of Sort/Merge control statements. This statement causes the Sort/Merge Program to cease reading the input control statement stream before receiving an end of file indication.

### Comment Statement

An asterisk in column one of a record in the input stream identifies the statement to be a comment. Comment statements can appear at any time in the input stream including between the continuation records of the other statements.

## Check Statements Carefully

Each statement is checked for validity before it is actioned by the Sort/ Merge Program. If an error is detected then a diagnostic message is issued. See Appendix B for a complete description of all messages. However not all errors or inconsistent combinations of entries can be detected, so care should be taken in preparing control statements.

# 2.2 Control Statement Format

All Sort/Merge Program control statements have the same general format:

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                    COLUMNS 00001 00072
COMMAND                                                         SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
****** *************************************************************************
000001 {Label}  Operation   Operand          Comments
000002                                                  Continuation Column
000003
000004           Column 1 must be blank unless a label is present
000005
000006
000007
```

Figure 2 Control Statement Format

The control statements are free form; that is, the operation definer, operand(s), and comments can appear anywhere in a statement, as long as 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 a label is present or it is a comment statement.

## Label Field

A label can optionally be specified on any control statement provided in the input stream. It must begin in column 1 and can be up to any length provided the operation definer is contained on the same statement. A label can contain any character other than blank and cannot have an asterisk as the first character. A label ends when the first blank character is found. Labels cannot appear on continuation records. The label field contents are not processed so any character string meaningful to the user can be coded.

## Operation Field

This field must appear first on the statement except when it is preceded by a label field. It must not extend beyond column 71 of the first statement. It contains a keyword, for example SORT or MERGE that identifies the statement type to the Sort/Merge Program. In the following Figure 3, the operation definer SORT is in the operation field of the example control statement.

## Operand Field

The operand field is made up of one or more operands separated by commas. This field must be separated from the operation field by at least one blank. No blanks are allowed within the parameters, but a blank is required at the end of all parameters. If the statement occupies more than one line, this field must begin on the first line. Operands supply parameters to the Sort/Merge Program. Each operand is made up of an operand definer or parameter.

A value or values can be associated with a keyword. The three possible operand formats are:

- keyword=(value1,value2,...,value)
- keyword=value
- keyword.

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00               Columns 00001 00072
Command ===>                                                Scroll ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 *       Comment Statement
000002 *         sample of all operand field types   Operand Formats    Comment
000003 *
000004          sort fields=(10,30,a),format=ch,ckpt,  sort on product code  x
000005 Operation       dynalloc=(3390,7)
000006 *                Continuation Statement         Continuation column
```

Figure 3 Example of each Control Statement operand format

## Comments Field

This field can contain any information. It is not required, but if present, it must be separated from the operand field by at least one blank. Informational Message IER009I is generated for each statement containing comments to assist in identifying a possibly incorrectly coded statement.

## Continuation Column (72)

Any character other than a blank in this column indicates that the present statement is continued on the next line. In Figure 3, X is used to specify that the next line contains more information pertaining to this SORT control statement.

## Continuation Statement

Columns 1 through 15 of a continuation statement must be blank. Up to 20 continuation statements per control statement can be coded except for an END statement where continuation is not relevant.

A continuation statement is treated as a logical extension of the preceding control statement. Either an operand or a comments field can begin on one statement and continue on the next. The following rules apply to continuing operands or comments fields:

- If an operand is continued through column 71, the next character of the operand must appear in column 16 of the continuation statement. Columns 1-15 must be left blank. An example of this is shown in Figure 3.
- If an operand field is broken between two lines without filling the first line through to column 71, it must be done in either of two ways:
  1. At the end of a complete operand followed by a comma and a blank (or blanks), or
  2. At the end of any of the values in an operand of the type keyword=(value$_1$, value$_2$, ... ,value$_n$) followed by a comma and a blank.

The following rules apply to control statement format:

- Column 1 of each statement must be blank unless a label is present or it is a comment statement with an asterisk in column 1.
- The operation field must be the first field on the first record or after a label on a control statement and cannot be continued over onto a continuation record.
- The operand field, if present, must begin on the first record of a control statement. The last operand in a statement must be followed by at least one blank.
- Embedded blanks are not allowed in operands. Anything following a blank is considered a comment.
- Values must not be more than eight alphanumeric characters in length.
- Commas and blanks can be used only as field delimiters. They must not be used in values.
- Each type of Sort/Merge control statement can appear only once for each run of the Sort/Merge Program.
- No more than 60 control statement records, including continuation statements, but not including comment statements, are accepted for a sorting or merging operation.
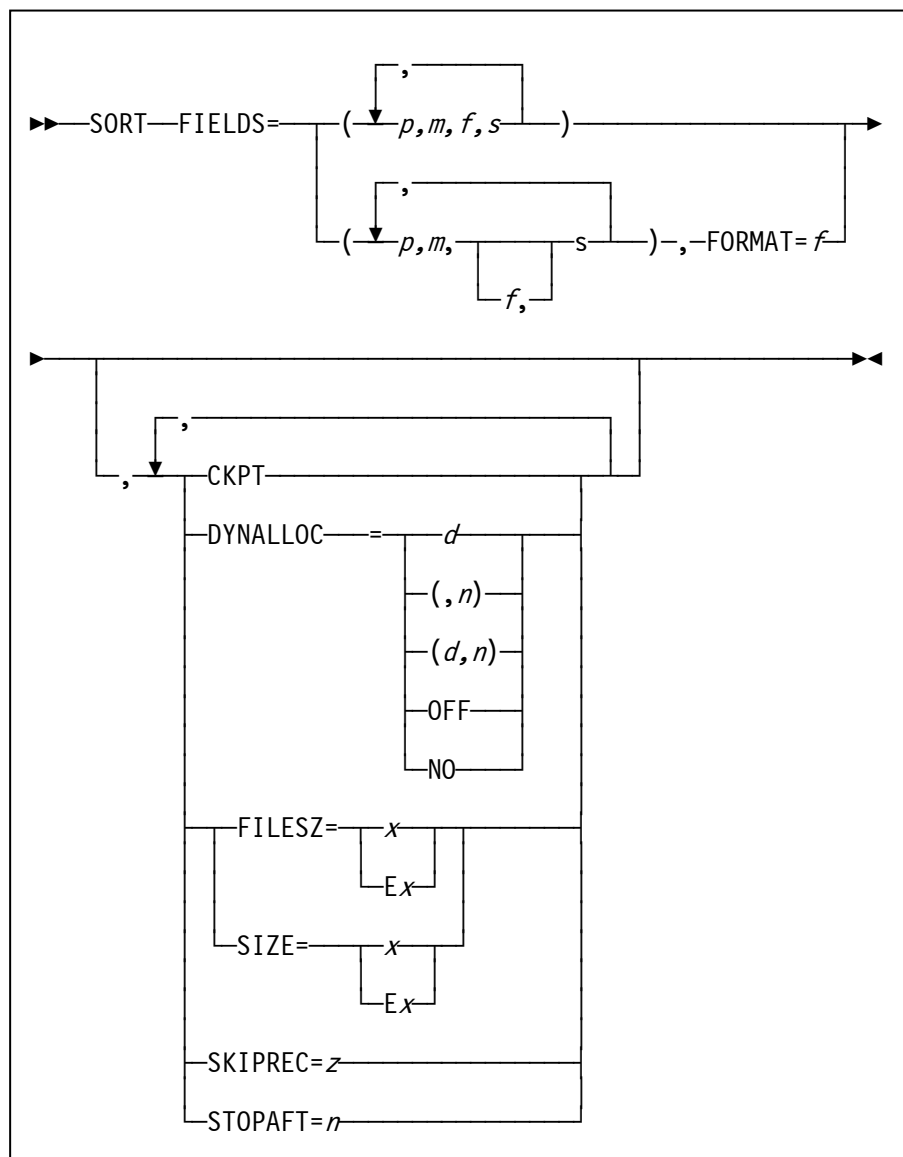
## 2.3 SORT Control Statement



Figure 4 SORT Control Statement

The SORT control statement must be used when a sorting application is to be performed. The statement describes the control fields that define the required sequence for the data output from the sorting operation.

The format of the SORT statement is shown in Figure 4. The first field in the statement, after any label field, must be the operation definer SORT, followed by at least one blank.

Some of the optional parameters available on the SORT control statement can also be specified on the OPTION control statement. If an optional parameter can be specified on an OPTION control statement then it is preferable to specify it on the OPTION control statement.

### FIELDS

The FIELDS operand describes control fields. As shown in Figure 4, it can be written in two ways. Use the first FIELDS format shown in Figure 4 to describe control fields that contain different data formats. Use the second FIELDS format shown in Figure 4 to describe control fields that contain data of the same format. The second format is optional, the first format can always be used.

The Sort/Merge Program requires four facts about each control field in the input records:

- The position of the field within the record,

- The length of the field,

- The format of the data in the field and

- The sequence into which the field is to be sorted.

These facts are communicated to the Sort/Merge Program by the values of the FIELDS operand which are represented by $p$, $m$, $f$, and $s$ in Figure 4.

The major control field, the most significant control field, is specified first. In Figure 4 $p$, $m$, $f$, and $s$ describe the major control field. Successive minor or less significant control fields can be specified, following the major control field, by repeating sets of $p$, $m$, $f$, and $s$ parameters. Up to 64 control fields can be specified.

*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. For variable-length records, the logical record includes the four-byte Record Descriptor Word. The first byte of a variable-length record is byte 5, the second is byte 6, and so on. All control fields, except a binary field, must begin on a byte boundary.

Fields containing binary values are described in bytes and bits as follows:

First, give the byte location relative to the beginning of the record and follow it with a period. Then give the bit location relative to the beginning of that byte. The resulting notation is then bytes.bits. The first (high-order) bit of a byte is bit 0; 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 field falls on a byte boundary, for example, the fourth byte, it can be written it in one of three ways:

    4.0
    4.
    4

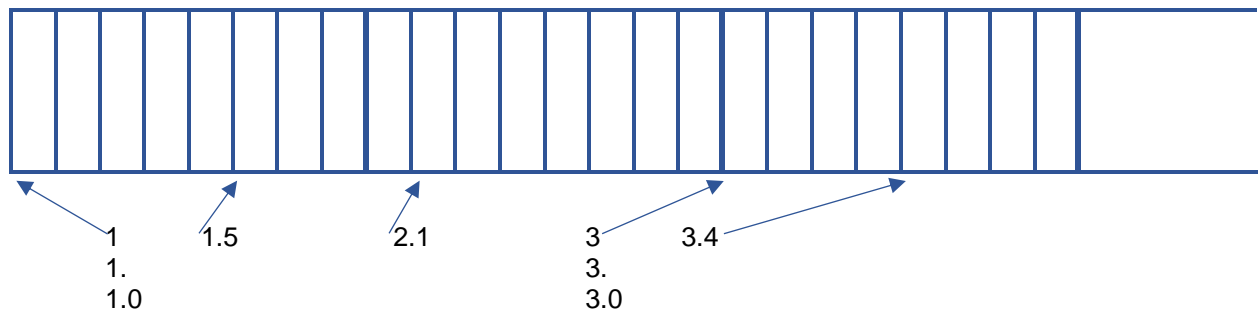Other examples of this notation are shown in Figure 5.



Figure 5 Example of Control Field Addressing

*m*

specifies the length of the control field. All control fields except binary fields must be a whole number of bytes in length. The length of a control field that is a whole number (d) bytes long can be expressed in one of three ways:

d.0

d.

d

Binary field lengths are expressed in the notation bytes.bits. The number of bits specified must not exceed 7. A control field two bits long would be represented as 0.2.

*f*

specifies the format of the data in the control field. Acceptable control field lengths and the available formats are shown in Table 3.

Table 3 Control Field Lengths and Formats

| Format | Length | Description |
|--------|--------|-------------|
| CH | 1 to 256 bytes | Character |
| ZD | 1 to 16 bytes | Signed Zoned Decimal |
| PD | 1 to 16 bytes | Signed Packed Decimal |
| FI | 1 to 256 bytes | Signed Fixed point |
| BI | 1 bit to 256 bytes | Unsigned Binary |
| FL | 1 to 256 bytes | Signed Floating Point |

If all the control fields contain the same type of data the f parameters can be omitted and the optional FORMAT=xx operand can be used.

*s*

specifies how the control field is to be ordered. One of the following one-character codes must be used for s:

A Ascending sequence

D Descending sequence

E User modification

If an E61 exit routine is used to modify control fields prior to the Sort/Merge Program sequencing the control fields then use E. After the provided exit has modified the control fields, the Sort/Merge Program orders the fields in absolute ascending sequence. See Exit E61, described in Chapter 5: Using Exit Routines, for further information about modifying control fields.

The following rules apply to all control fields described on a SORT control statement:

- All control fields must be located within the first 4,092 bytes of a record.
- The first byte of a floating-point control field is interpreted as a signed exponent. The rest of the control field is interpreted as the fraction.
- All floating-point data must be normalized before the Sort/Merge Program can collate floating-point data correctly. User exit routines can be provided to do this at run time. See Exit E61 in Chapter 5: Using Exit Routines. Specify the E option for the value of s in the FIELDS operand for each control field that will be modified by an E61 exit routine.
- The total number of bytes occupied by all control fields must not exceed 256. A binary field is considered to occupy an entire byte if it occupies any part of a byte. For example, a binary field that begins on byte 2.6 and is 3 bits long occupies two bytes.

## FORMAT

```
►►──FORMAT=f─────────────────────────────────────────────►◄
```
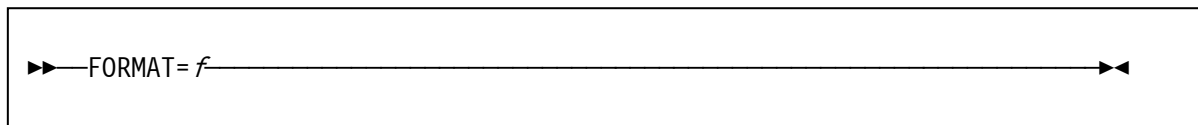
If all the control fields on a SORT control statement contain the same type of data, then optionally use this operand instead of the *f* parameter of the FIELDS operand to specify the data format. If all the control fields are not of the same type, the *f* parameter of the FIELDS operand must be used. The possible values for use are the same as those for the *f* parameter. Format=*f* must be provided when a control field or fields are described only by *p*, *m*, and *s* fields.

## CKPT

```
►►──CKPT──────────────────────────────────────────────────►◄
```
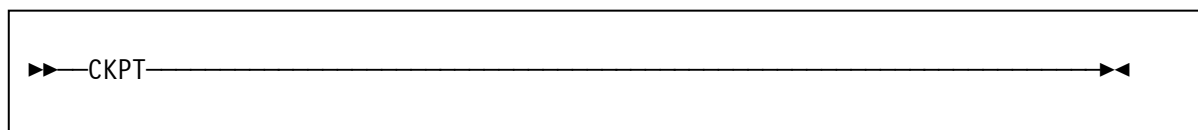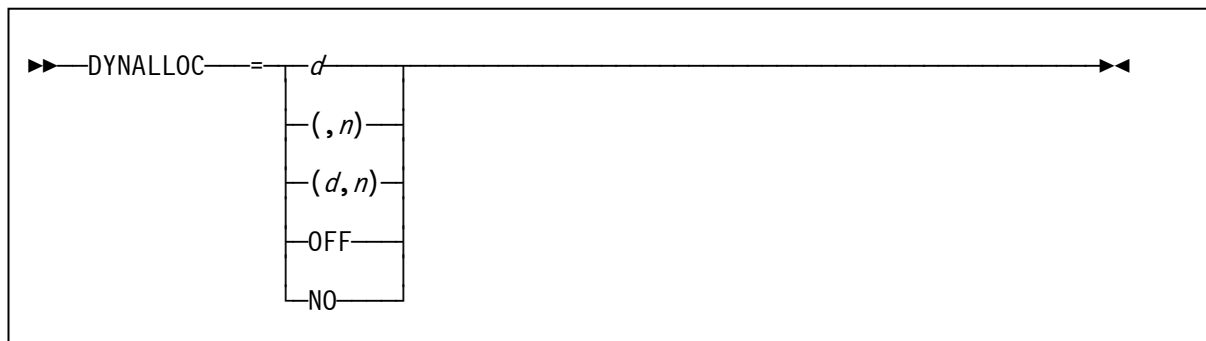
The CKPT operand activates the Sort/Merge Program's use of the checkpoint/restart facility of the Operating System. The Sort/Merge Program takes checkpoints at the start of the sort phase and at the start of the final merge phase. When the balanced or polyphase tape techniques are used, checkpoints are also taken at the start of each intermediate merge phase pass. If the oscillating tape technique is used then checkpoints are taken at intervals during the intermediate merge phase.

In addition to those taken at the beginning of each pass, the balanced direct access technique takes checkpoints at selected intervals during the intermediate merge phase.

The Sort/Merge Program can be restarted from the last checkpoint taken or from the checkpoint written at the start of the sort phase.

When the checkpoint/restart facility is being used, a data set must be defined for the checkpoint records. The data set is described further in Chapter 3: Invoking the Sort/Merge Program with JCL Language.

`DYNALLOC`

```
►►──DYNALLOC──=──┬──d───────┬──────────────────────────────►◄
                 ├─( , n)───┤
                 ├─( d, n)──┤
                 ├─OFF──────┤
                 └─NO───────┘
```

When DASD is selected for intermediate storage then using the DYNALLOC parameter avoids the need to calculate the amount of intermediate storage required for the sorting operation and to provide JCL to allocate the required intermediate storage. If the input data set is DASD resident then the amount of work space required can be calculated by using information provided by control statements and input data set space usage. The dynamic allocation facility of the Operating System is then used to allocate the intermediate storage data sets.

*d*

specifies the device name for the allocation in the same way as specified on the JCL DD statement UNIT parameter. All DASD unit types supported by the Operating System can be specified. Allocation across different DASD unit types for a specific sorting operation is not supported

User assigned group names or esoteric names can be used to direct the allocation to a specific pool of DASD units established at the time of the Operating System generation. Do not select a user assigned group name or esoteric name that contains a number of different DASD unit types. The Operating System can allocate the intermediate storage data sets on any DASD unit included in the user assigned group name or esoteric name. If the allocation results in more than one DASD unit type being allocated then the sorting operation will fail.

If *d* is not provided or omitted then the default device name set at installation customization time will be used.

*n*

specifies the number of work data sets to be allocated. The amount of intermediate working storage that the Sort/Merge Program calculated would be required for the sorting operation is divided equally across the *n* work data sets.
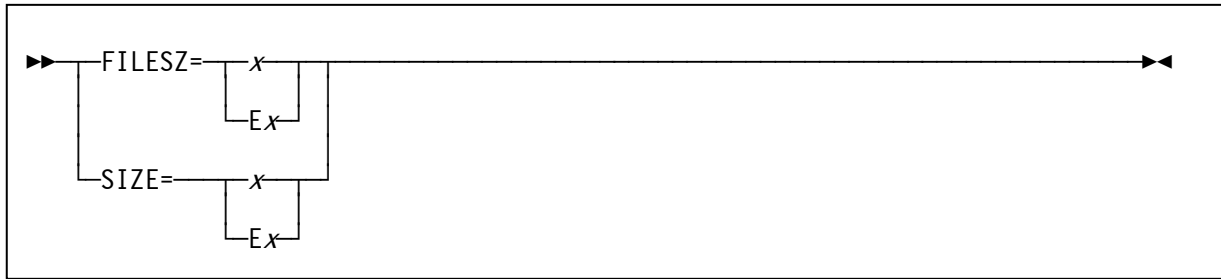
If *n* is not provided or omitted then the default number of work data sets set at installation customization time will be used.

`OFF or NO`

The intermediate storage requirements are not calculated and no intermediate storage work data sets are allocated. The sorting operation will fail unless JCL statements are used to provide the required work data sets.

Note that if the installation customization option DYNAUTO=NO is in effect then the DYNALLOC parameter will be ignored and JCL statements will be required to provide the work data sets. If DYNAUTO=IGNWKDD was specified at installation customization time then any JCL provided intermediate storage data sets will be deleted and dynamic allocation will be used to allocate new work data sets.

## SIZE or FILSZ

```
►►─┬─FILESZ=─┬─x──┬─────────────────────────────────►◄
   │         └─Ex─┘
   └─SIZE=──┬─x──┬─
            └─Ex─┘
```

FILESZ or SIZE specifies the number of records in the input data set. The value *x* can be either the actual number of records or an estimate of the number of records.

If the actual number of records is provided, and not an estimate, then the number of records in the input data set, as counted by the Sort/Merge Program, must match the value provided by the FILESZ parameter. The sort is terminated if the values do not match. The value specified in the FILESZ parameter is placed in the IN field of message IER047A or IER054I. If an estimated number of records is provided, precede the value by E (for example, E5000). The actual or estimated number of records, if provided, is used by the dynamic allocation facility to calculate the amount of DASD space required for intermediate storage.

If the FILESZ operand is omitted the Sort/Merge Program assumes that:

- If intermediate storage is tape, the input data set can be contained on one volume at the blocking factor used by the Sort/Merge Program.
- If intermediate storage is DASD, all the records in the input data set will fit into the space allocated for intermediate storage.

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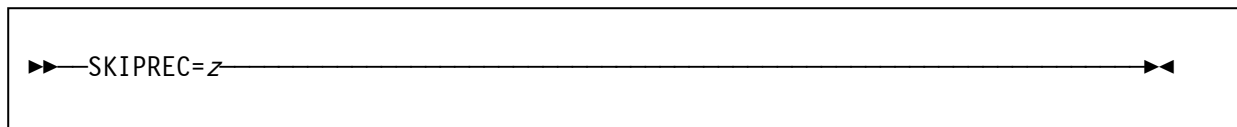
```
►►──SKIPREC=z───────────────────────────────────────►◄
```

SKIPREC causes a number of records to be skipped over from the start of the input data set before the remaining records in the input data set are read for sorting. Substitute the number of records desired to be skipped for *z*. This operand only applies to records read from SORTIN. It does not apply to input records provided by user exit routines.

## STOPAFT
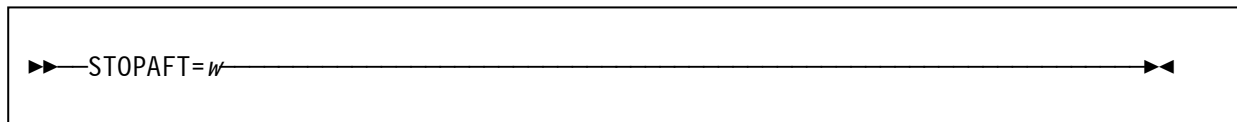
```
►►──STOPAFT=w───────────────────────────────────────►◄
```

STOPAFT sets a limit on the number of records read from the input data set for sorting. Set *w* to the desired maximum number of input records to be sorted. This operand only applies to records read in from SORTIN. It does not apply to input records provided by user exit routines.

## 2.3.1 SORT Statement Examples

### Example 1

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                                SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001     SORT FIELDS=(2,5,CH,A),SIZE=29483
000002 *
000003 *
```

Figure 6 SORT Control Statement Example 1- One Control Field and Size Option

#### FIELDS

The control field begins in the second byte of the record and extends for 5 bytes. The data is character data and will be sorted into an ascending sequence.

#### SIZE

The input data set contains exactly 29,483 records.

### Example 2

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                                SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001          sort fields=(7,3,ch,d,1,5,fi,a,398,7,bi,d,99,230,bi,a,       x
000002             452,8,fl,a),stopaft=10000
000003 *
000004 *
```

Figure 7 SORT Control Statement Example 2 - Five Control Fields and STOPAFT Option

#### FIELDS

The first four values describe the major control field. It begins on byte 7 of each record, is 3 bytes long, contains character data, and is to be sorted into a descending sequence.

The next four values describe the second control field. It begins on byte 1, is 5 bytes long, contains fixed-point data, and is to be sorted into an ascending sequence.

The third control field begins on byte 398. The field is 7 bytes in length, and contains binary data to be placed in a descending order.

The fourth control field begins on byte 99, is 230 bytes long and contains binary data, and is to be sorted into an ascending order.

The fifth control field begins on byte 452, is 8 bytes long, contains normalized floating-point data which is to be sorted into an ascending order.

#### STOPAFT

A maximum of 10000 records will be read from SORTIN or until end of file indication, whichever occurs first, before commencing the sorting operation.

## Example 3

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                                 SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 *
000002     sort fields=(3,8,ZR,E,40,6,CH,D),SIZE=E30000
000003 *
000004
```

Figure 8 SORT Control Statement Example 3 - Two Control Fields, User Modification, SIZE Option

### FIELDS

The first four values describe the major control field. It begins on byte 3 of each record, is 8 bytes long, contains zoned decimal data that will be modified by an exit routine before being sequenced.

The second field begins on byte 40, is 6 bytes long, contains character data and will be sorted into a descending sequence.

### SIZE

The input data set contains approximately 30000 records.

## Example 4

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                                 SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001     sort fields=(25,4,a,48,8,a),format=ch,size=20000,                 X
000002             dynalloc=(3390,3),                                        X
000003             skiprec=10000,stopaft=20000
000004 *
000005 *
```

Figure 9 SORT Control Statement Example 4 - Two Control Fields, Multiple Options

### FIELDS

The major control field begins on byte 25 of each record, is 4 bytes long, contains character data and is to be sorted into an ascending sequence.

The second control field begins on byte 48, is 8 bytes long, has the same data format as the first field, and is also to be sorted into an ascending sequence.

### FORMAT

The FORMAT=CH option can be used because both control fields have the same data format.

### SIZE

Exactly 20,000 records will be sorted.

### DYNALLOC

The DASD space required for intermediate work data sets will be calculated and then 3 work data sets will be allocated on 3390 DASD.

## SKIPREC and STOPAFT

A selection of the records in the input data set will be sorted. The first 10,000 records will be skipped over and then the next 20,000 records will be read and sequenced according to the specified control fields.
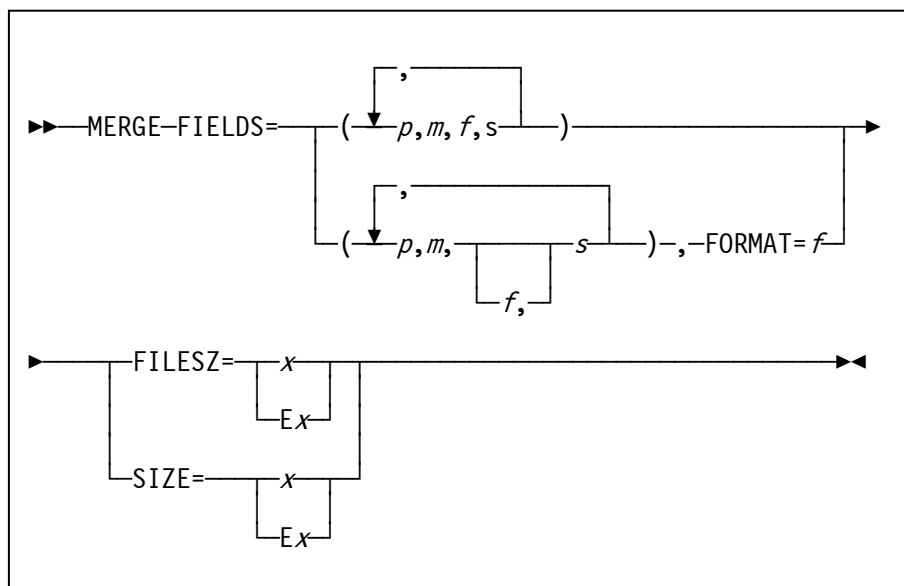
## 2.4 MERGE Control Statement



Figure 10 MERGE Control Statement

The MERGE control statement must be used when a merge-only operation is to be performed. It provides essentially the same information for a merging operation as the SORT control statement does for a sorting operating. The format of the MERGE statement is very much like that of the SORT control statement. There are the following differences:

- The operation definer is MERGE.
- The SKIPREC, STOPAFT, CKPT and DYNALLOC parameters are not used.
- The value of the SIZE operand is the sum of the number of records in all input data sets.

### 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* were described previously in the discussion of the SORT control statement.

### FILESZ or SIZE

The FILESZ or SIZE operand is optional. Its value can be either exact or estimated. The value refers to the total number of records in all the input data sets to be merged.

### Example 1

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                                 SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001          MERGE FIELDS=(25,4,A,48,8,A),FORMAT=ZD,SIZE=E30000
000002 *
```

Figure 11 MERGE Control Statement Example 1 - Two Control Fields, FORMAT and SIZE Options

### FIELDS

The major control field begins on byte 25 of each record, is 4 bytes long, and contains zoned decimal data that has been placed in an ascending sequence.

The second control field begins on byte 48, is 8 bytes long, is also in zoned decimal format, and is also in an ascending sequence.

`FORMAT`

The FORMAT=ZD option can be used because both control fields have the same data format.

`SIZE`

The total number of records to be merged in all input data sets is estimated to be 30,000.
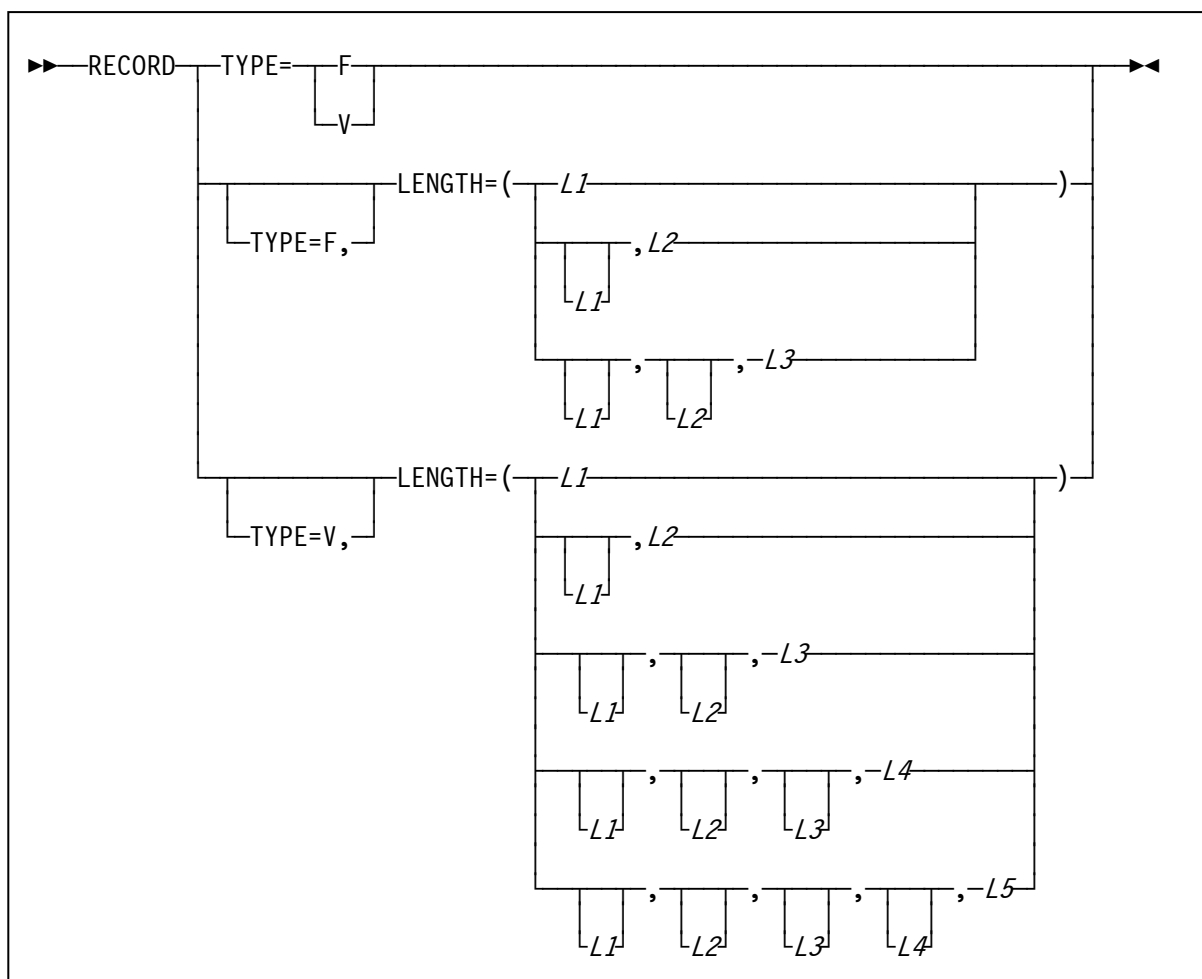
## 2.5 RECORD Control Statement



Figure 12 RECORD Control Statement

The RECORD control statement is used to specify the type and lengths of the records being sorted or merged. For variable-length records the minimum and average record lengths can also be provided to enable a more efficient sorting process.

A RECORD statement is required when:

- User exit routines change record lengths during a Sort/Merge Program run.
- The default values calculated for a variable-length record sort are not appropriate.
- A user exit routine supplies all the input records.

### TYPE

The TYPE operand specifies whether the input records are fixed or variable-length format.

TYPE=F indicates fixed-length records. There is no Record Descriptor Word (RDW) at the start of each record so the first data position for SORT control statements is byte 1.

TYPE=V indicates variable-length records. There is a Record Descriptor Word (RDW) at the start of each record so the first data position for SORT control statements is byte 5.

## LENGTH

The LENGTH operand specifies the length in bytes of the input records, the length in bytes of the records that enter the sort phase of the Sort/Merge Program, (exit routines can change record lengths before the records are sorted), and the length in bytes of the records in the output data set (exit routines can change record lengths during the final merge phase).

The *L1* value is required whenever the RECORD statement is used. The *L2* and *L3* values are only required when exit routines change record lengths before the sort or during the final merge. The *L4* and *L5* values are only used for variable-length records.

### Defining Fixed-Length Records

If the input records are fixed-length, use *L1*, *L2* and *L3* as follows:

### *L1*

is the length of each record in the input data set. If the RECORD control statement is used then this value must be specified. The value should be the same as the value specified in the LRECL sub-parameter of the DCB parameter on the SORTIN DD statement that is discussed later in this section. If the values are not the same, the Sort/Merge Program uses the value specified on the DD statement.

### *L2*

is the length of each record processed by the sort phase. If a value is not specified for *L2*, it defaults to the same value as *L1*. If user exits are going to change record lengths in the sort phase, a value for *L2* must be specified. *L2* is not needed for a merging application.

### *L3*

is the length of each record in the output data set. If a value for *L3* is not specified it defaults to the same value as *L2* for a sorting application and *L1* for a merging application. If exit routines change record lengths during the final merge phase then a value must be specified for *L3*. This value should be the same as the value specified for the LRECL sub-parameter of the DCB parameter on the SORTOUT DD statement, discussed later in this section. If the values are different, the value given on the DD statement is used.

### Defining Variable-Length Records

If the input records are variable-length, use *L1*, *L2*, *L3*, *L4*, and *L5* as follows:

### *L1*

is the maximum length of the records in the input data set. If the RECORD statement is used, a value for *L1* must be specified. The value should be the same as the value specified in the LRECL sub-parameter of the DCB parameter on the SORTIN DD statement, discussed later in this section. If the values are not the same, the DCB LRECL value is used.

### *L2*

is the maximum length of the records handled by the sort phase. If a value is not specified for *L2* it defaults to the same value as *L1*. If an exit routine changes record lengths in the sort phase, then a value for *L2* must be provided. *L2* is not needed for a merging application.

### *L3*

is the maximum length of each record in the output data set. If a value for *L3* is not specified it defaults to the same value as *L2* for a sort and *L1* for a merge. If an exit routine that changes record lengths in the final merge phase is used then a value for *L3* must be specified. The value should be the same as the value provided for the LRECL sub-parameter of the DCB parameter on the SORTOUT DD statement. If it is not, the DCB LRECL value is used.

*L4*

is the minimum length of records in the input data set. If a value is not specified for *L4*, the default value is the minimum record size necessary to contain the control fields defined on the SORT or MERGE control statement, or the minimum record length allowed by the Operating System, whichever is greater. *L4* is not needed for a merge.

*L5*

is the record length that occurs most frequently in the input data set (modal length). Use this value to help define a data set biased toward a particular length. If a value is not specified for *L5* the default value is an average of the maximum and minimum record lengths in the input data set. If, for example, the input data set contains mostly small records and just a few long records, the Sort/Merge Program will default to a high modal length and would allocate a larger record storage area than necessary. Conversely, if the input data set contains just a few short records and many long records, the Sort/Merge Program will default to a low modal length and might not allocate a large enough record storage area to sort the data. The *L5* value will also impact the intermediate work area calculations when using the dynamic allocation facility.

## Notes

When using the RECORD statement, consider the following:

The lengths specified for variable-length records must include the 4-byte Record Descriptor Word that the Operating System places at the beginning of each record.

When using DASD for intermediate storage the record length cannot exceed the track capacity of the selected DASD unit type. Refer to Table 1 in Chapter 1: Introduction for maximum record sizes permitted for specific DASD unit types.

The minimum record length of records is 18 bytes.

The record format specified in the TYPE operand must be the same as the format used in the RECFM sub-parameter of the DCB parameter on the SORTIN and SORTOUT DD statements described later in this section. If the formats are not the same, the one specified in the JCL DD statement is used.

The following rules apply to omitting values from the LENGTH operand:

- Values that are equal to those defaulted to by the Sort/Merge Program can be omitted.
- Values can be dropped from right to left. If, for example, all the values after *L2* are equal to the values defaulted by the Sort/Merge Program, then LENGTH can be written LENGTH=(*L1*,*L2*).
- If values are dropped from the middle or from left to right, commas must be used to indicate their omission. If *L2* will default to the correct value then write LENGTH=(*L1*,,*L3*).

## Example 1

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                             SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001     record type=f,length=(60,40,50)
000002 *
```

Figure 13 RECORD Control Statement Example 1 - Fixed-length, Three Length Values

TYPE

The input records are fixed-length.

LENGTH

The records in the input data set are each 60 bytes long. Exit E15 changes the records to 40 bytes in the sort phase and Exit E35 changes the records to 50 bytes in the final merge phase.

## Example 2

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                Columns 00001 00072
Command ===>                                             Scroll ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001          RECORD type=v,length=(780,      record maximum length      X
000002                 ,                        record length not changed  X
000003          780,                            input record = output len  X
000004          250,                            minimum record length      X
000005          420)                            median record length
000006 *
000007 *
```

Figure 14 RECORD Control Statement Example 2 - Variable-length, Four Length Values provided

### TYPE

The input records are variable-length.

### LENGTH

The records in the input data set have a maximum length of 780 bytes. There are no exits being used so the length remains unchanged during processing. The minimum length of the record is 250 bytes. The average length of the records in the input data set is 420 bytes. The dynamic allocation facility will use this median length value to calculate the amount of intermediate working storage needed for this sorting run.
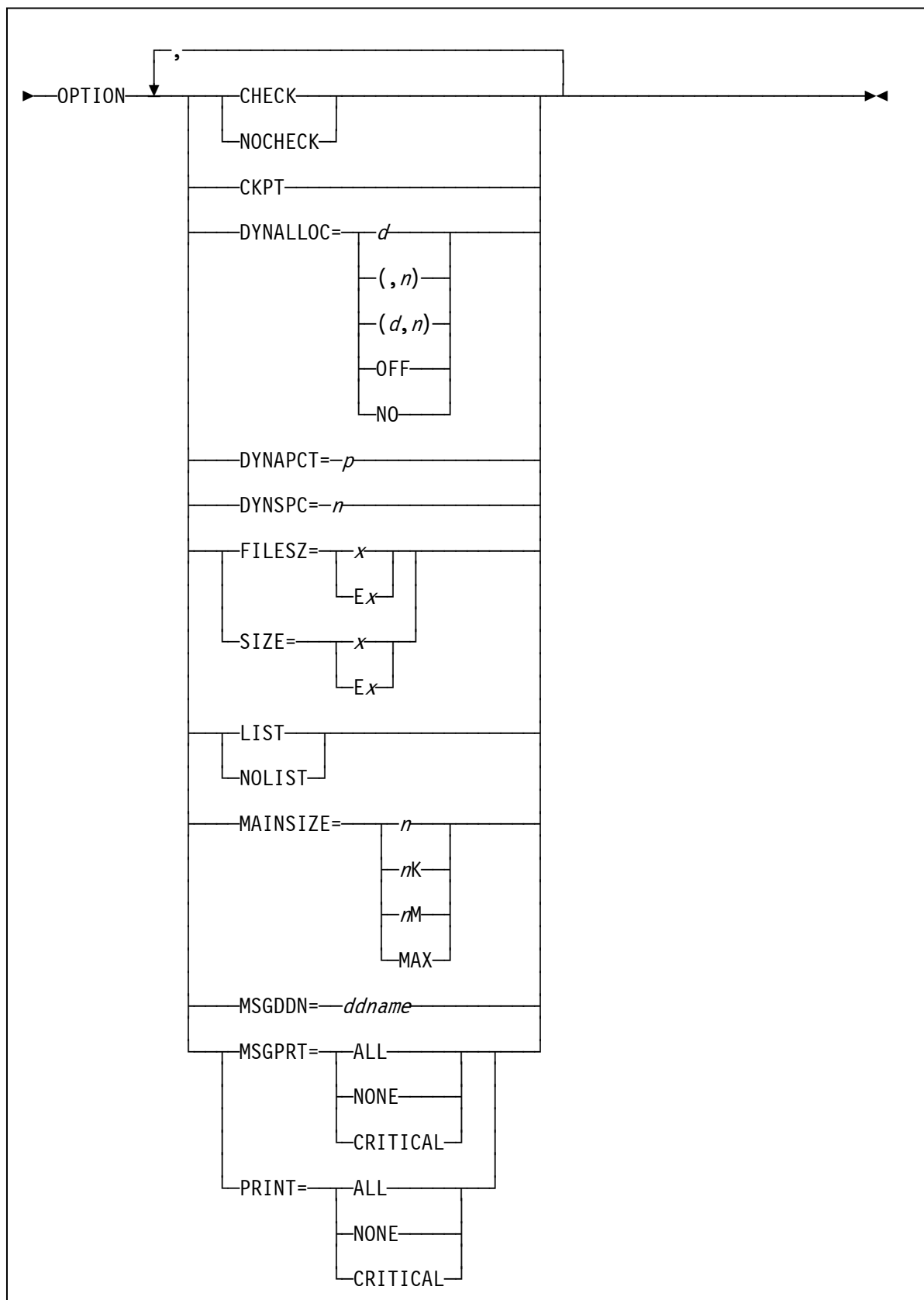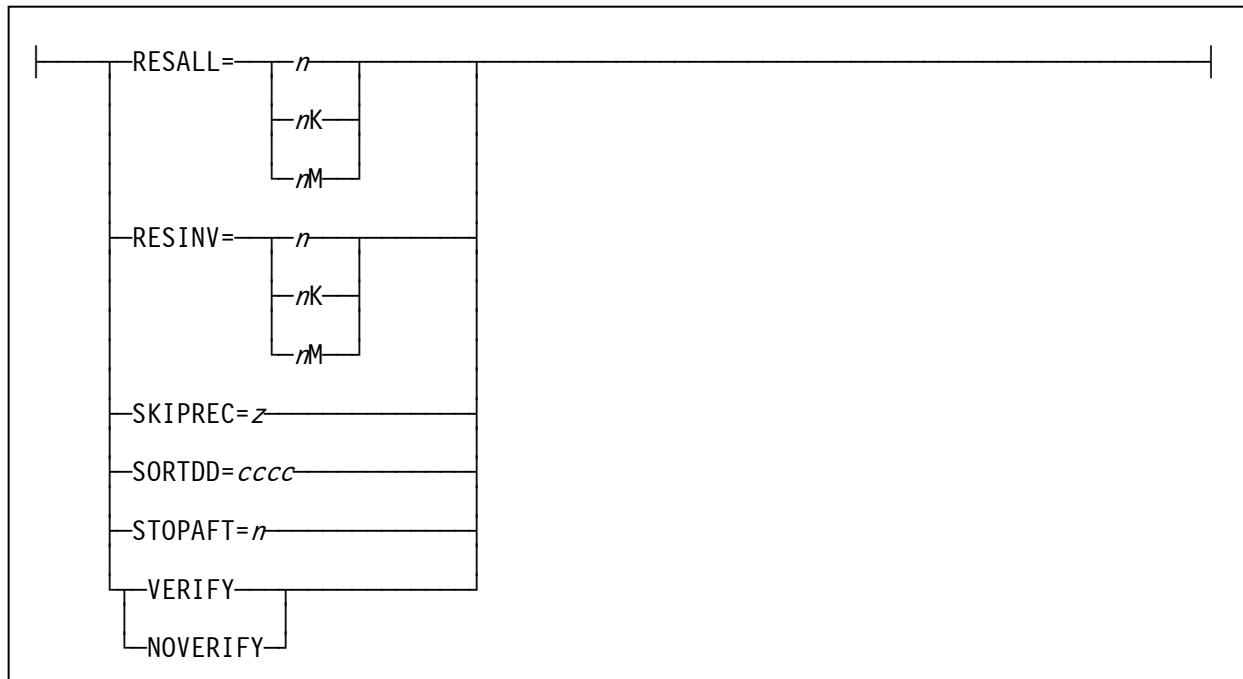
---

## 2.6 OPTION Control Statement



Figure 15 OPTION Control Statement

```
├──────RESALL=──┬───n────┐
│               ├───nK───┤
│               └───nM───┘
│
├──────RESINV=──┬───n────┐
│               ├───nK───┤
│               └───nM───┘
│
├──SKIPREC=z───────────────
│
├──SORTDD=cccc─────────────
│
├──STOPAFT=n───────────────
│
├──VERIFY──────┐
└──NOVERIFY────┘
```

The OPTION control statement provides the ability to override some of the customization options set when the Sort/Merge Program was installed. Some of the OPTION control statement parameters can also be specified on the SORT or the MERGE control statement. Using an OPTION control statement to provide these options assists in separating the description of the input data and required sequencing from parameters that can change on an individual sorting operation such as dynamic allocation of intermediate work areas or storage usage.

## CHECK or NOCHECK

```
►►──┬──CHECK────┬──────────────────────────────────►◄
    └──NOCHECK──┘
```
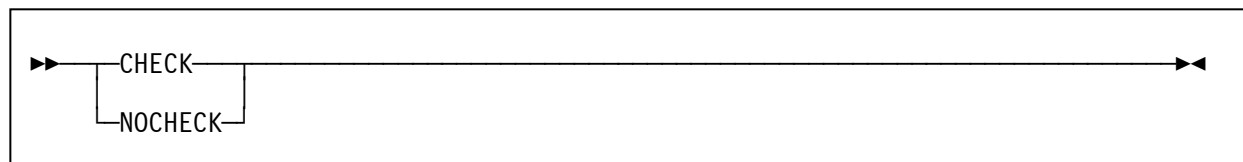
The CHECK/NOCHECK parameter overrides the installation customization setting. Use the CHECK option to specify that the record count check should apply for sorting operations that only use the E35 exit to receive records without a SORTOUT data set. Specify NOCHECK to bypass the record count check.

## CKPT

```
►►──CKPT───────────────────────────────────────────►◄
```
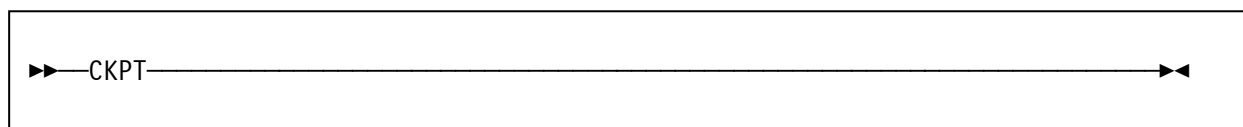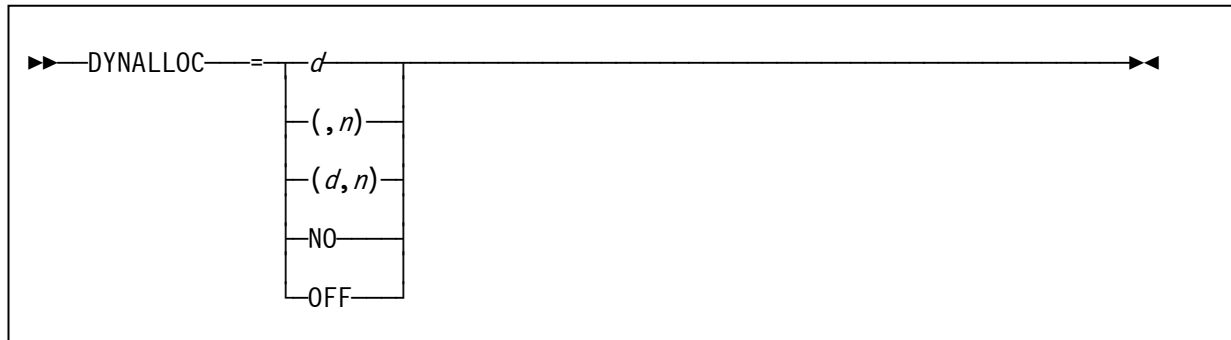
This operand activates the Sort/Merge Program's use of the checkpoint/restart facility of the Operating System. The Sort/Merge Program takes checkpoints at the start of the sort phase and at the start of the final merge phase. When the balanced or polyphase tape techniques are used checkpoints are also taken at the start of each intermediate merge phase pass. If the oscillating tape technique is used checkpoints are taken at intervals during the intermediate merge phase.

In addition to those taken at the beginning of each pass, the balanced direct access technique takes checkpoints at selected intervals during the intermediate merge phase.

The Sort/Merge Program can be restarted from the last checkpoint taken or from the checkpoint written at the start of the sort phase.

When the checkpoint/restart facility is being used, a data set must be defined for the checkpoint records. The data set is described further in Chapter 3: Invoking the Sort/Merge Program with JCL Language.

### DYNALLOC

```
►►──DYNALLOC────=──┬─d────────┬──────────────────────────────►◄
                   ├─(,n)─────┤
                   ├─(d,n)────┤
                   ├─NO───────┤
                   └─OFF──────┘
```

If DASD is selected for intermediate storage then using the DYNALLOC parameter avoids the need to calculate the amount of intermediate storage required for the sorting operation and to provide JCL to allocate the required intermediate storage. The amount of work space required is calculated by using information provided by control statements and input data set space requirements, if the input data set is DASD resident. The dynamic allocation facility of the Operating System is then used to allocate the intermediate storage data sets.

### *d*

specifies the device name for the allocation in the same way as specified on the JCL DD statement UNIT parameter. All DASD unit types supported by the Operating System can be specified. Allocation across different DASD unit types for a specific sorting operation is not supported

User assigned group names or esoteric names can be used to direct the allocation to a specific pool of DASD units established at the time of the Operating System generation. Do not select a user assigned group name or esoteric name that contains a number of different DASD unit types. The Operating System can allocate the intermediate storage data sets on any DASD unit included in the user assigned group name or esoteric name. If the allocation results in more than one DASD unit type being allocated then the sorting operation will fail.

If *d* is not provided or omitted then the default device name set at installation customization time will be used.

### *n*

specifies the number of work data sets to be allocated. The amount of intermediate working storage that the Sort/Merge Program calculated would be required for the sorting operation is divided equally across the *n* work data sets.

If *n* is not provided or omitted then the default number of work data sets set at installation customization time will be used.

### OFF or NO

The intermediate storage requirements are not calculated and no intermediate storage work data sets are allocated. The sorting operation will fail unless JCL statements are used to provide the required work data sets.

Note that if the installation customization option DYNAUTO=NO is in effect then the DYNALLOC parameter will be ignored and JCL statements will be required to provide the work data sets. If DYNAUTO=IGNWKDD was specified at installation customization time then any JCL provided intermediate storage data sets will be deleted and dynamic allocation will be used to allocate new work data sets.

### DYNAPCT

```
►►────DYNAPCT=p──────────────────────────────────────────────►◄
```
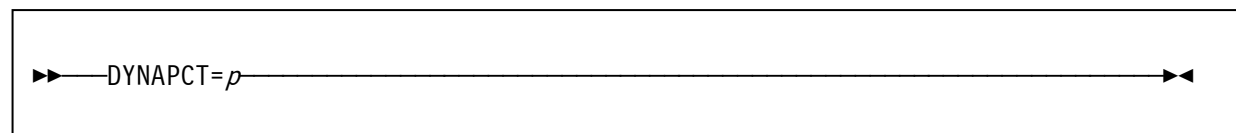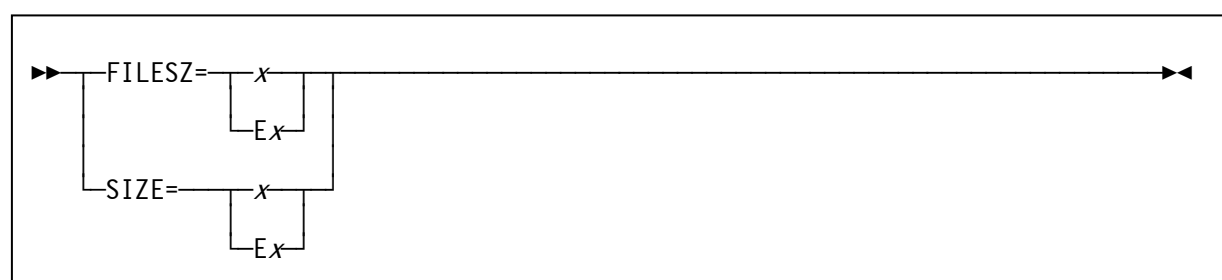
DYNAPCT provides a percentage uplift to the default value, or the value calculated, for the amount of intermediate storage to be allocated for this sorting operation. This parameter can be used for variable-length record sorting when the calculated median record length value is less than the actual value resulting in insufficient intermediate storage being dynamically allocated. Applying a sufficient percentage uplift to the amount of intermediate storage allocated can enable the successful completion of the sorting operation.

### DYNSPC

```
►►────DYNSPC=n──────────────────────────────────────────────►◄
```
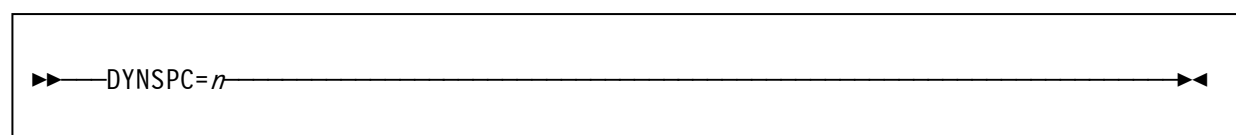
The DYNSPC parameter can be used to override the DYNSPC installation customization setting. When the dynamic allocation feature is used this parameter specifies, in megabytes, the total amount of intermediate storage to be allocated for work data sets. This value is only used when the input record count or FILSZ is not provided and the input data set is not resident on DASD. This situation is most likely to occur when an E15 user exit is used to provide all the input records for sorting.

### FILSZ or SIZE

```
►►──┬─FILESZ=─┬──x──┬───────────────────────────────────►◄
    │         └──Ex─┘
    └─SIZE=───┬──x──┐
              └──Ex─┘
```

FILSZ or SIZE specifies the number of records in the input data set. The value x can be either the actual number of records or an estimate of the number of records.

If the actual number of records is provided, and not an estimate, then the number of records in the input data set, as counted by the Sort/Merge Program, must match the value provided by the FILESZ parameter. The sort is terminated if the values do not match. The value specified in the FILESZ parameter is placed in the IN field of message IER047A or IER054I. If an estimated number of records is provided, precede the value by E (for example, E5000). The actual or estimated number of records, if provided, is used by the dynamic allocation facility to calculate the amount of DASD space required for intermediate storage.

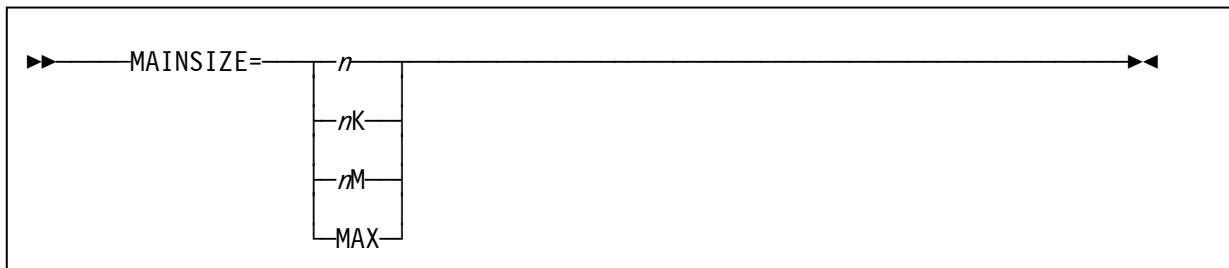If the FILESZ operand is omitted the Sort/Merge Program assumes that:

- If intermediate storage is tape, the input data set can be contained on one volume at the blocking factor used by the Sort/Merge Program.
- If intermediate storage is DASD, all the records in the input data set will fit into the space allocated for intermediate storage.

## LIST or NOLIST

```
►►─────────────LIST──────────────────────────────────────────────────────►◄
              └─NOLIST─┘
```
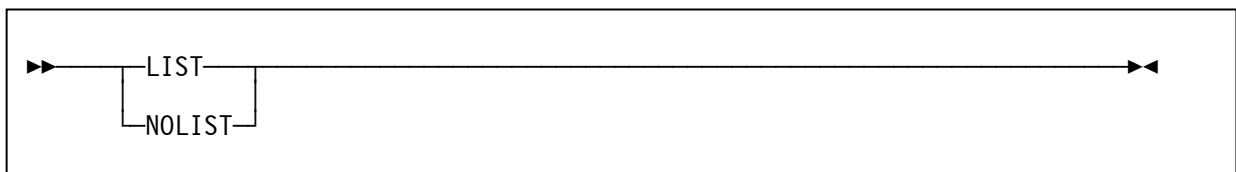
The LIST/NOLIST parameter can be used to override the LIST/NOLIST installation customization setting. If LIST is specified then all input control statements, including control statements passed by program invoked sorts, will be written to the selected message output stream.

## MAINSIZE

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

MAINSIZE provides an override to the SIZE installation customization setting. MAINSIZE sets the limit for the amount of storage used by the sorting operation. The MAINSIZE value cannot exceed the limit set by MAXLIM or be less than the value set for MINLIM installation customization settings. For compatibility with previous versions CORE can be used instead of MAINSIZE.

### $n$

specifies the maximum storage to be used in bytes

### $n$K

specifies the maximum storage to be used as a number of kilobytes
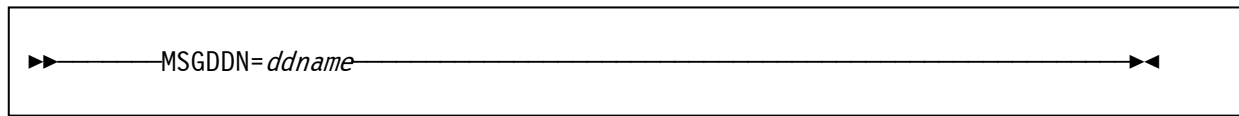
### $n$M

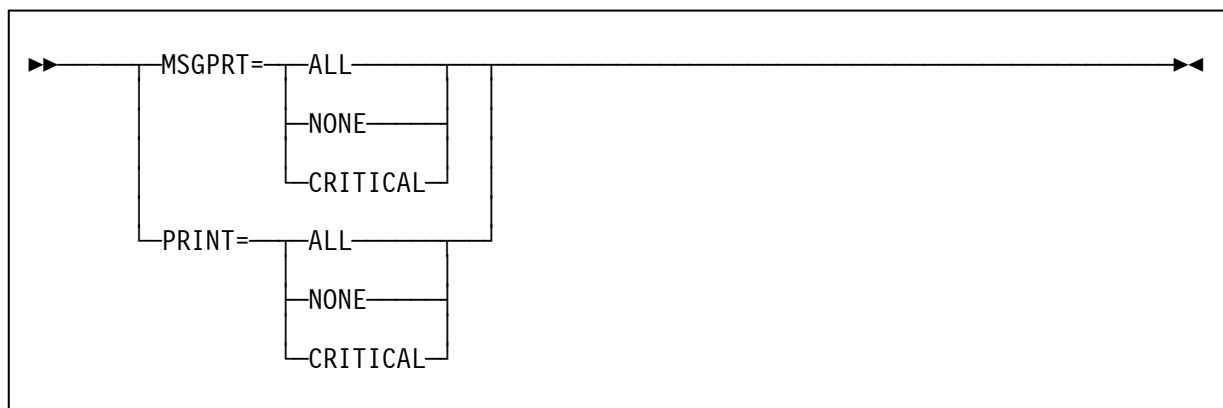specifies the maximum storage to be used as a number of megabytes

### MAX

specifies that the maximum amount of storage, up to the limit set by the MAXLIM installation customization setting, will be used for this sorting operation.

---

**MSGDDN**

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

MSGDDN provides an override to the MSGDDN installation customization setting. The sort message stream will be written to the selected ddname. Care should be taken in selecting a ddname to avoid conflict with a ddname that can be used by for input, output or work data sets.

**MSGPRT or PRINT**

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

MSGPRT or PRINT provides an override to the MSGPRT installation customization setting. This option provides a filter on the flow of messages to the message data set.

**ALL**

all messages except diagnostic messages will be written to the message data set. Control statements input to the Sort/Merge Program will only be listed if the LIST option has been specified.
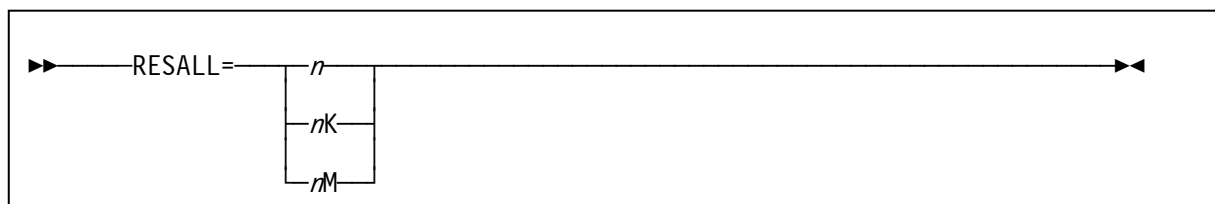
**CRITICAL**

only messages critical to the operation of the Sort/Merge Program will be written to the message data set.

**NONE**

no messages will be written to the message data set. This option is not recommended because if the sorting operation is not successful then there would be no indication of why the sorting operation failed.

**RESALL**

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

RESALL provides a temporary override to the RESALL installation customization setting. The RESALL option is only in effect when:

- the Sort/Merge Program has been invoked with JCL statements and
- the MAINSIZE=MAX option is in effect.

When MAINSIZE=MAX is in effect all available storage up to the limit specified in the MAXLIM installation customization setting is used by the Sort/Merge Program. The RESALL parameter value sets the number of bytes to be reserved for system use by reducing the total storage used by the Sort/Merge Program.

*n*

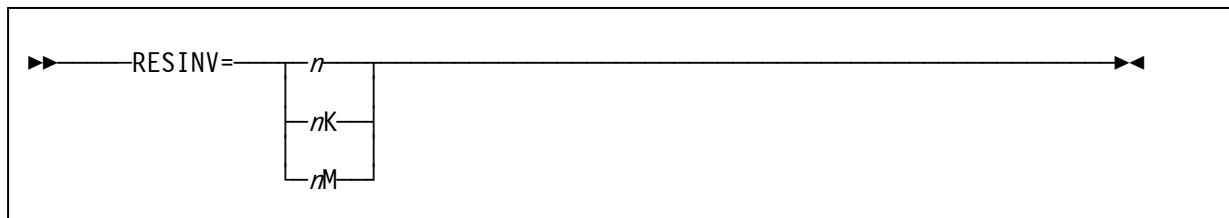specifies the maximum storage to be used in bytes

*n*K

specifies the maximum storage to be used as a number of kilobytes

*n*M

specifies the maximum storage to be used as a number of megabytes


## RESINV

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

Provides a temporary override to the RESINV installation customization setting. The RESINV option is only in effect when:

- the Sort/Merge Program has been dynamically invoked by another program and
- the MAINSIZE=MAX option is in effect.

When MAINSIZE=MAX is in effect all available storage, up to the limit specified in the MAXLIM installation customization setting, is used by the Sort/Merge Program. The RESINV parameter value sets the number of bytes to be reserved for system use by reducing the total storage used by the Sort/Merge Program.

*n*

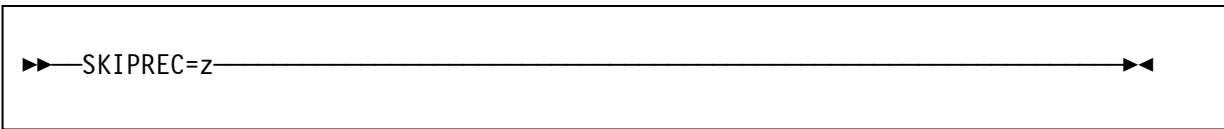specifies the maximum storage to be used in bytes

*n*K

specifies the maximum storage to be used as a number of kilobytes

*n*M

specifies the maximum storage to be used as a number of megabytes

### SKIPREC

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

SKIPREC causes a number of records to be skipped over from the start of the input data set before the remaining records in the input data set are read for sorting. Substitute the number of records desired to be skipped for *z*. This operand only applies to records read from SORTIN. It does not apply to input records provided by user exit routines.

### SORTDD

```
►►──SORTDD=cccc───────────────────────────────────────►◄
```
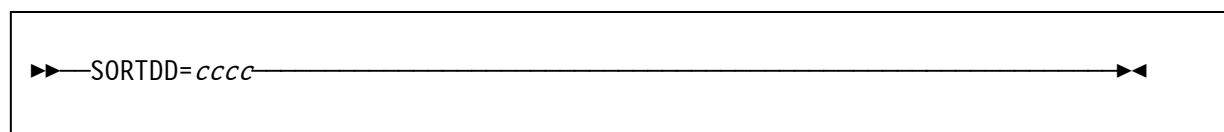
SORTDD specifies the four character prefix for ddnames used by the Sort/Merge Program. The four characters replace the default SORTDD installation customization setting. The four characters replace the first four characters in the following ddnames: SORTIN, SORTOUT, SORTINnn, SORTWKnn and SORTCNTL. This parameter does not apply to the ddname used for the message data set. That ddname is set by the MSGDDN parameter.

### STOPAFT
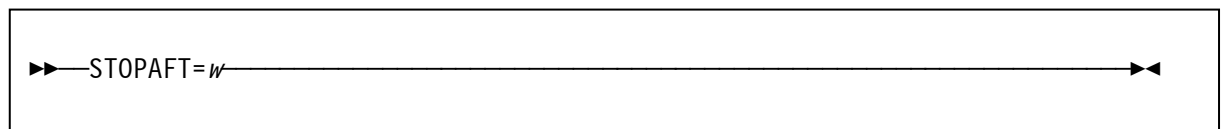
```
►►──STOPAFT=w─────────────────────────────────────────►◄
```

STOPAFT sets a limit on the number of records read from the input data set for sorting. Set *w* to the desired maximum number of input records to be sorted. This operand only applies to records read in from SORTIN. It does not apply to input records provided by user exit routines.

### VERIFY or NOVERIFY

```
►►──┬─VERIFY───┬──────────────────────────────────────►◄
    └─NOVERIFY─┘
```
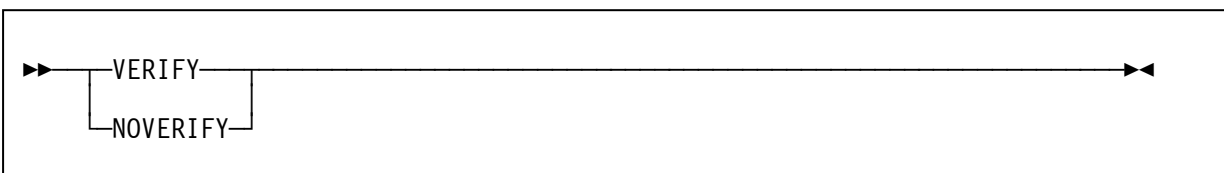
Provides a temporary override to the VERIFY/NOVERIFY installation customization setting. This parameter determines if the Sort/Merge Program is to perform a sequence check on the final output of records from the sorting operation.

**VERIFY**

specifies that sequence checking is performed

**NOVERIFY**

specifies that no sequence checking is to be performed

## Example 1

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                                  SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001          SORT FIELDS=(15,10,CH,A),
000002          OPTION SIZE=E50000,STOPAFT=50000,DYNALLOC=YES
000003 *
000004 *
```

Figure 16 OPTION Control Statement Example 1

**FIELDS**

The single control field is 10 bytes in length, beginning in the 15$^{th}$ byte of each record, contains character data and is to be sequenced in an ascending order.

**DYNALLOC**

The intermediate storage requirements will be dynamically allocated using the DASD unit type and number of intermediate storage data sets determined by the installation customization default values.

**SIZE**

The Sort/Merge Program will calculate the intermediate storage requirements needed for sorting 50,000 records. An estimated value has been provided in case there are less than 50,000 records in the input data set.

**STOPAFT**

The Sort/Merge Program will read a maximum of 50,000 records from SORTIN or until end of file indication, whichever occurs first, before commencing the sorting operation.

## Example 2

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                Columns 00001 00072
Command ===>                                                  Scroll ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001          option filsz=2000,skiprec=500,dynalloc=3390
000002          sort fields=(25,20,ch,a),skiprec=100,dynalloc=(3350,7)
000003 *
000004 *
```

Figure 17 OPTION Control Statement Example 2

**FILSZ**

The Sort/Merge Program will calculate the intermediate storage requirements needed for sorting 2000 records.

**SKIPREC**

Five hundred records at the beginning of the input data set are skipped over by the Sort/Merge Program. The SKIPREC value on the SORT control statement is overridden by the SKIPREC value coded on the OPTION statement.

### DYNALLOC

The intermediate work data sets will be dynamically allocated to 3390 DASD. The DYNALLOC parameter on the SORT control statement is partially overridden. The work data sets will not be allocated to 3350 DASD. There will however, be seven work data sets allocated on 3390 DASD.

### FIELDS

The single control field is 20 bytes in length, beginning in the 25th byte of each record, contains character data and is to be sequenced in an ascending order.

## Example 3

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                             SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001          option resinv=64000,sortdd=srt1,mainsize=768k
000002 *
000003 *
```

Figure 18 OPTION Control Statement Example 3

The Sort/Merge Program has been invoked by another program. Some of the parameters passed by the invoking program need to be changed. To avoid changing the invoking program an OPTION control statement has been provided to override some of the settings passed to the Sort/Merge Program.

### MAINSIZE

The MINSIZE value has been set to 768 KB

### RESINV

The storage reserved for the invoking program is set to 64,000 bytes

### SORTDD

The four character prefix for ddnames used by the Sort/Merge Program has been set to SRT1. This parameter does not apply to the ddname used for the message data set. That ddname can be changed by use of the MSGDDN parameter.
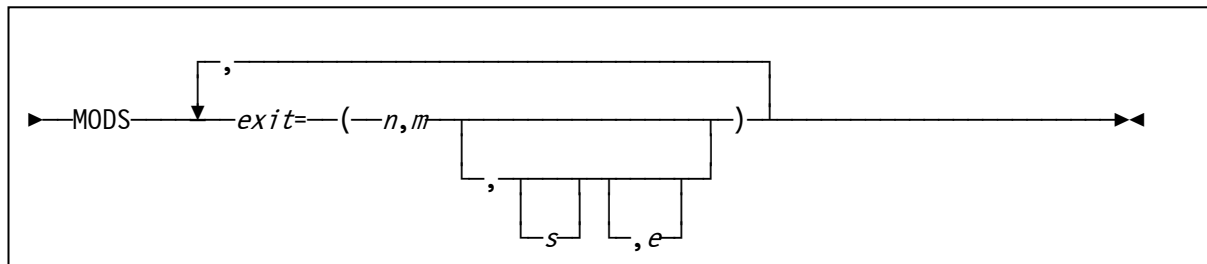
## 2.7　MODS Control Statement



Figure 19 MODS Control Statement

The MODS statement is only required if control is to be passed to exit routine(s) at various points during a sorting or merging operation. The statement associates user-provided routines with specific exit points together with the storage requirement and location of the routines. For details about exits in the Sort/Merge Program and how to use them, refer to Chapter 5: Using Exit Routines.

Control can be transferred to user-provided routines at seventeen different exit points. These exits are described in detail in Chapter 5: Using Exit Routines. The exits have 3 character names such as E11, E15, E16, E28, etc. To use one of these exits, substitute its 3 character name for the word *exit* in the MODS Control Statement format in Figure 19.

### *exit*
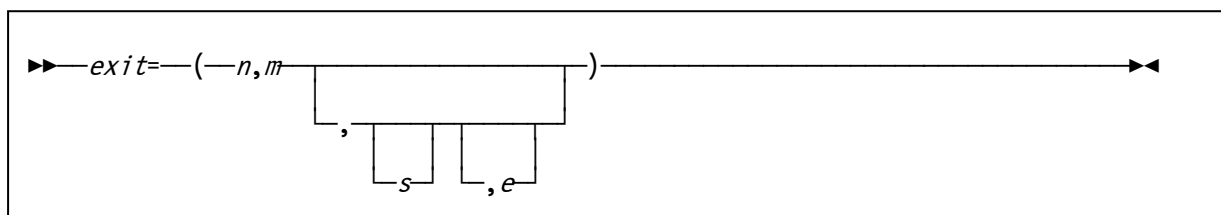


The values associated with the 3 character name describe the routine. These values are:

### *n*

the name of the routine, member name if the routine is in a load module library. If the routine has been link edited previously and it does not require link editing again, then the name must be the same as the three-character exit name with which it is associated.

### *m*

the number of bytes, exact or approximate, of storage required by the routine.

### *s*

either the name of the DD statement in the Sort/Merge job step that defines the load module library in which the routine is located, or SYSIN if the routine is in the input stream. If the routines do not require link editing and are in different data sets then concatenate the data sets and provide the DD name of the concatenated data set as the value of s for all of the routines.

### *e*

indicates the linkage editor requirements of the exit routine.

N

means that the routine has already been link edited and can be used in the Sort/Merge run without further link editing. This is the default value.

S

means that the routine requires link editing but that it can be link edited separately from the other routines used in a particular program phase. Only routines at exits E11, E21, and E31 are eligible for separate link editing. Refer to Chapter 5: Using Exit Routines for further information.

Absence of these parameters means that the routine must be link edited together with the other routines being used in a particular Sort/Merge Program phase.

Refer to the topic Bypassing the Linkage Editor in Chapter 5: Using Exit Routines for details on how to design the exit routines.

When preparing the MODS control statement, consider the following:

- The Sort/Merge Program must know the amount of main storage the exit routine needs so that it can allocate storage correctly for its use. If the exact number of bytes the exit requires is not known, make a slightly high estimate. The value of $m$ in the MODS statement is written the same whether it is an exact figure or an estimate. In other words, do not precede the value by E for an estimate as coded on the SORT or the MERGE statement.
- If the exit routines being used for a particular Sort or Merge run are in several load module libraries, a DD statement is needed for each library concatenated to the first load module exit library. DD statements required for the exit routines are described later in this section.
- If the exit routines are in the system input stream (SYSIN), they must be provided in numerical order, for example the E11 routine before the E15 routine. If the same routine is used in several program phases a separate copy of the routine must be provided for each use.
- Routines can also reside in private libraries

## Example 1

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                              SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001       MODS  E15=(E15,1600,EXITLIB,N),E35=(E35,16384,EXITLIB,N)
000002 *
000003 *
```

Figure 20 MODS Control Statement Example 1

### E15

At exit point E15, control will be transferred to the provided E15 routine. The exit routine is in the library defined by the EXITLIB DD statement. Its member name is E15, it is 1,600 bytes long, and has been link edited previously, and does not require further link editing.

### E35

At exit point E35, control will be transferred to the provided E35 routine. The routine is in the library defined by the EXITLIB DD statement, its member name is E35, and is 16,384 bytes long and has been link edited previously.

**Example 2**

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPLE) - 1.01              Columns 00001 00072
Command ===>                                             Scroll ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001       mods e16=(nmaxerr,1200,modlib),e21=(e21,600,modlib,n),        X
000002            e31=(e31,760,modlib,n),e35=(sumup,6000,sysin)
000003 *
000004 *
```

Figure 21 MODS Control Statement Example 2

**E16**

Control will be transferred to the exit routine NMAXERR at exit point E16. The exit routine is located in the load module library defined by the MODLIB DD statement, and is approximately 1,200 bytes long. As the module name does not match the exit name it will be link edited together with other routines in the phase.

**E21**

At exit point E21 control will be passed to the exit routine that resides in the library defined by the MODLIB DD statement under the member name E21. The exit routine is 600 bytes long and does not require additional link editing.

**E31**

Another of the routines in the library defined by the MODLIB DD statement will gain control at exit E31. Its member name is E31, it is 760 bytes long and does not require additional link editing.

**E35**

For exit point E35 a routine named SUMUP in object form is in the input stream. It is approximately 6000 bytes long and must be link edited together with other routines in its phase. No N or S parameter is provided so the E35 routine must be link edited together with other routines used for that phase.

## 2.8   DEBUG Control Statement

The DEBUG control statement is not intended for regular use. This control statement is described in the document OS/360 Sort/Merge for MVS 3.8 Installation, Customization and Diagnosis Guide.

## 2.9   END Control Statement

```
►►──END────────────────────────────────────────────────────────────►◄
```

The END control statement signals the end of a related group of Sort/Merge Program control statements. This statement causes the Sort/Merge Program to cease reading the input steam prior to receiving an end of file indication. Use of the END control statement is entirely optional except when a user exit routine in object deck format is included in the input stream.

## 2.10  Determining Intermediate Storage Requirements

To perform a sorting application the Sort/Merge Program needs to store data on intermediate storage. The basic factors to consider when allocating intermediate storage are the device type and the number of records in the input data set together with the number of records that will be inserted into the sort by user exits. Another factor which must sometimes be considered is the amount of storage assigned to the Sort/Merge Program. In general the less storage the Sort/Merge Program has available for use then the more intermediate storage is required to complete the sorting application. The user can either calculate the amount and then allocate sufficient intermediate storage or alternatively use the dynamic allocation facility of the Sort/Merge Program to automatically allocate the required intermediate storage.

### Intermediate Storage Devices

Intermediate storage can either be on DASD or magnetic tape, but not on a mixture of both. The dynamic allocation facility is only available for the allocation of intermediate storage on DASD.

All models of 2400 or 3400 Magnetic Tape Units can be used for intermediate storage. Intermediate storage can be on any DASD unit type supported by MVS 3.8. If DASD is selected for intermediate storage, then only one DASD unit type is permitted for a given sorting application.

### Intermediate Storage Space Requirements

If the dynamic allocation facility is not being used for intermediate storage then use the following formulas to calculate the amount of intermediate storage necessary for a specific sorting application. Unless a sequence distribution technique is forced the user will not know which technique will be used by the Sort/Merge Program. This causes no difficulty, however. The amount of intermediate storage assigned can affect the Sort/Merge Program's choice of a technique. In other words, it is possible to implicitly rule out one technique by not providing enough intermediate storage for its use. To avoid this possibility, calculate the intermediate storage required by all the techniques and provide the largest amount required.

For information on assigning intermediate storage for efficient program operation, refer to Chapter 6: Efficient Program Use.

### Tape Intermediate Storage Requirements Calculation

Use the following formulas to calculate the minimum number of tape volumes ($N$) needed to complete a tape sort for a given data set size and sequence distribution technique:

Formula 1 for balanced tape technique - maximum input is 15 volumes.

$$N = 2(X + 1)$$

Formula 2 for oscillating tape technique - maximum input is 15 volumes.

$$N = X + 2$$

Formula 3 for polyphase tape technique - maximum input is 1 volume.

$$N = 3$$

where:

$N$ is the number of tape volumes needed to complete a tape sort

$X$ represents the number of volumes required to contain the input data set with a blocking factor equal to that used for intermediate storage by the Sort/Merge Program.

The maximum number of tape units that can be used for intermediate storage are:

- 32 for the balanced technique.
- 17 for the oscillating technique.
- 17 for the polyphase technique.

These maximums permit the sorting of 15 volumes of input with the balanced and oscillating techniques. The polyphase technique will only support one volume of input.

## DASD Balanced Technique (BALN) Intermediate Storage Requirements Calculation

Use the following formula to calculate the approximate number of tracks (T) required to complete a DASD sort for a given data set size. If the data set tends to be ordered in reverse of the sequence required for output, more intermediate storage might be needed. Conversely, if the input data set tends to be ordered in the desired sequence, less intermediate storage is required.

$$T = \frac{S(N)}{k(N-1)} + 2N$$

where:

$T$ is the total number of tracks required.

$N$ is the number of intermediate storage areas. At least three are required, but no more than six.

$S$ is the number of records in the input data set, exact or approximate.

$$k = \frac{B * 0.95}{L}$$

where:

$B$ is the maximum record size permitted on a track for the selected DASD unit type.

$L$ is the length in bytes of each record in the input data set. For variable-length records, $L$ is the maximum length.

Only the integer portion of $k$ is used for calculating $T$. Disregard the remainder, whatever its value. If the formula yields $k = 0$, use the value 1.

At least 3 intermediate storage areas must be available to use the BALN technique with each area defined as a separate data set. All data sets must be at least 5 tracks in length. Up to 6 areas can be provided. Divide the number of tracks ($T$) among the data sets rounding up each allocation to at least 5 tracks. The formula is based on areas of equal size. More tracks will be needed if $T$ is not divided equally.

## DASD Balanced Technique (BALN) Intermediate Storage Example

Determine $T$ for 3350 DASD using 4 intermediate storage data sets, fixed-length records; length 120 bytes, estimated input data set size 25,500 records.

S(N) = 25500 * 4 = 102000

k = (19069*0.95)/120 = 150

T = 102000/(150*3) + 8

T = 234

Divide $T$ among the 4 data sets. Each data set is 58 tracks.

## DASD Crisscross Technique (CRCX) Intermediate Storage Requirements Calculation

Use the following formula to calculate the approximate total number of tracks ($T$) required to complete a sort when intermediate storage is on DASD and the crisscross (CRCX) sequence distribution technique is used

$T = 1.35S/k$

where:

$S$ is the number of records in the input data set, either actual or approximate.

$k = B * 0.95/L$

where:

$B$ is the maximum record size permitted on a track for the selected DASD unit type

$L$ is the number of bytes in each record in the input data set.

For variable-length records, $L$ is the maximum record length. Use only the integer portion of $k$. Disregard the remainder, whatever its value. If the formula yields $k=0$, use the value 1.

When the input data set DASD space allocation is known, there is no need to use the above formula to determine intermediate storage space. Assign intermediate storage space that is at least 50% larger than the space occupied by the input data set.

If the data set tends to be ordered in reverse of the desired output sequence, more intermediate storage space is necessary. Conversely, if the data set tends to be ordered in the desired sequence, less space is required. The Sort/Merge Program requires a minimum of 6 DASD intermediate storage data sets when the crisscross (CRCX) technique is selected by a user-provided parameter. The CRCX technique will always be selected when at least 7 DASD intermediate storage data sets are provided. A maximum of 17 DASD intermediate storage data sets can be used. Each area must contain at least five tracks.

## DASD Crisscross Technique (CRCX) Intermediate Storage Example

Determine $T$ for 3350 DASD using 7 intermediate storage data sets, fixed-length records; length 120 bytes, estimated input data set size 25,500 records.

k = (19069*0.95)/120 = 150

T = (1.35*25500)/150

T = 230

Divide $T$ among the 7 data sets. Each data set is 33 tracks.

# 3. Invoking the Sort/Merge Program with JCL Language

When the Sort/Merge Program is initiated via the system input stream JCL statements are used to:
- Identify the sorting or merging job to the Operating System.
- Allocate the necessary resources to run the job.
- Run the job.
- Provide status information concerning the successful completion or failure of the job.
- Terminate and release resources on job completion.

The JCL required includes a JOB statement, an EXEC statement, and DD statements. The statements needed and their parameters depend upon installation standards, how the Sort/Merge Program is invoked, use of cataloged procedures and specification of user exit routines.

The JCL statements and their functions are listed below. Specific details on the requirements for each DD statement are provided in following sections.

| Sort JCL DD Statement | Function |
| --- | --- |
| STEPLIB | Defines the load module library for sort definition phase modules. Generally not required as these modules are placed in a system library |
| SORTLIB | Defines the load module library for the sort modules required for the assignment and run time phases |
| SYSOUT | Sort message output data set |
| SORTIN | Source of input records for a sorting operation |
| SORTINnn | Source of input records for a merging operation |
| SORTOUT | Output data set for records from a sort or merge operation |
| SORTWKdd | Intermediate storage data sets for sorting operations |
| IERPARM | Sort/Merge Program control statements |
| SYSIN | Sort/Merge Program control statements |
| SORTCNTL | Sort/Merge Program control statements for a program invoked sort |
| SORTDIAG | Sort diagnostic message output data set. Not for general use |
| SORTCKPT | Checkpoint data set used when CHKP option is specified |

Additional DD statements are required if the linkage editor is invoked to dynamically link user exit routines.

| Linkage Editor JCL DD Statement | Function |
| --- | --- |
| SYSPRINT | Message output data set |
| SYSUT1 | Intermediate storage |
| SYSLIN | Control statement input |
| SYSLMOD | Load module storage |
| SORTMODS | Temporary storage for user exit routine object decks |

Some of the default DD names listed above can be changed either by installation customization settings or can be overridden for each sorting or merging operation. Review the installation customization settings and other control statements to determine if a different DD name must be used.

## JOB Statement

The JOB statement for a Sort/Merge job is a standard MVS 3.8 JCL JOB statement. This must be the first JCL statement in the job stream. A valid job name in the name field is required with all other parameters being optional. However, in most installations, certain fields and their values are mandatory. Determine the installation's requirements before submitting the sort job.

```
//JOBNAME JOB 'ACCT INFO',PROGNAME,CLASS=x,MSGCLASS=y etc
```
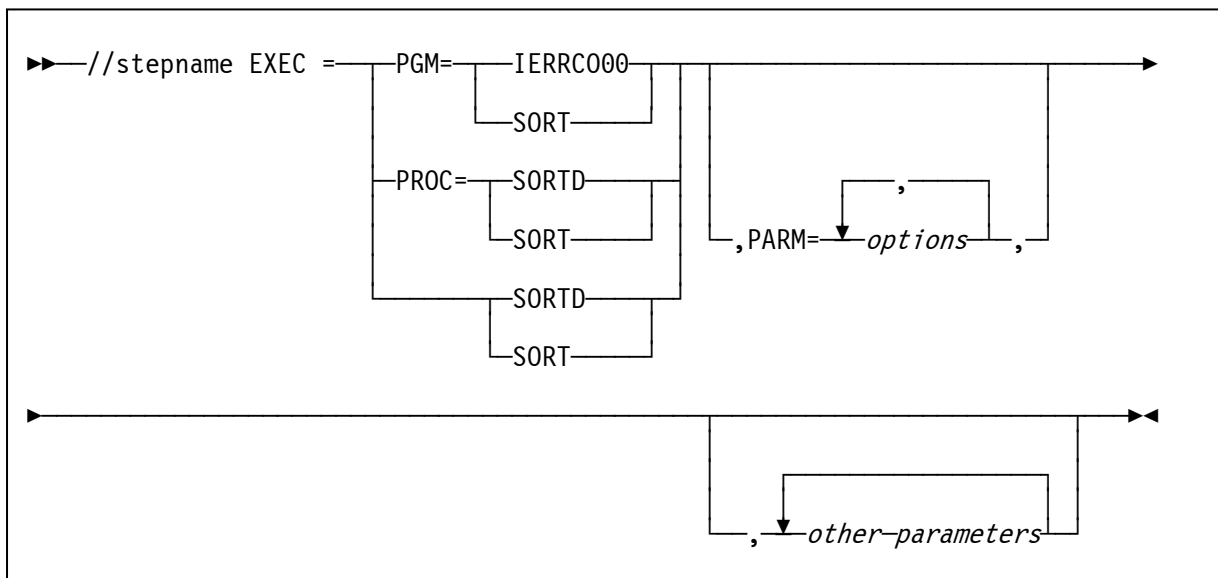
## EXEC Statement



Figure 22 EXEC Statement Parameters

The EXEC statement is the first JCL statement for each job step. For a sorting job it identifies either a Sort/Merge cataloged procedure or the Sort/ Merge Program to MVS 3.8. The PROC= notation serves as a reminder that a cataloged procedure is being invoked.

Two cataloged procedures are supplied with the Sort/Merge Program. They are named SORTD and SORT. Either procedure can be used. However the SORT cataloged procedure contains the additional DD statements required to invoke the Linkage Editor to link edit user-provided exit routines. If the Linkage Editor is not going to be invoked then the SORTD procedure is the more efficient procedure as the additional overhead of allocating unused data sets is avoided.

When the SORTD procedure is used the following JCL statements are generated.

```
REVEDIT  SYSD.SORTMVS.CNTL(SORTD) - 1.00             COLUMNS 00001 00072
COMMAND ===>                                          SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 //SORT    EXEC PGM=IERRCO000
000002 //SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
000003 //SYSOUT   DD  SYSOUT=*
```

Figure 23 SORTD Cataloged Procedure

| Line | Explanation |
|---|---|
| 000001 | The stepname is SORT and the program being invoked is IERRCO000 |
| 000002 | The SORTLIB DD statement defines the load module library containing the load modules for the assignment and run phases of the Sort/Merge Program |
| 000003 | Messages are written to the output class set by the MSGCLASS parameter. |

When the SORT procedure is used the following JCL statements are generated.

```
REVEDIT  SYSD.SORTMVS.CNTL(SORT) - 1.00                 COLUMNS 00001 00072
COMMAND ===>                                            SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 //SORT    EXEC PGM=IERRCO000
000002 //SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
000003 //SYSOUT   DD  SYSOUT=*
000004 //SYSPRINT DD  SYSOUT=*
000005 //SYSUT1   DD  UNIT=VIO,SPACE=(TRK,(10,10))
000006 //SYSLIN   DD  UNIT=VIO,SPACE=(TRK,(10,10))
000007 //SYSLMOD  DD  UNIT=VIO,SPACE=(TRK,(10,10,1))
```

Figure 24 SORT Cataloged Procedure

| Line | Explanation |
|------|-------------|
| 000001 | The stepname is SORT and the program being invoked is IERRCO00 |
| 000002 | The SORTLIB DD statement defines the load module library containing the load modules for the assignment and run phases of the Sort/Merge Program |
| 000003 | The output message stream. Directed to the output class set by the MSGCLASS parameter on the JOB statement |
| 000004 | The linkage editor message stream. Directed to the output class set by the MSGCLASS parameter on the JOB statement |
| 000005 | Linkage editor work data set |
| 000006 | Linkage editor input control data set |
| 000007 | Linkage editor load module output data set. Used to store the user exit load modules |

## 3.1 Specifying PARM Options

When the Sort/Merge Program is invoked with JCL certain options can be provided by a JCL EXEC PARM parameter. The options that can be specified are a subset of those that can be specified on an OPTION control statement.
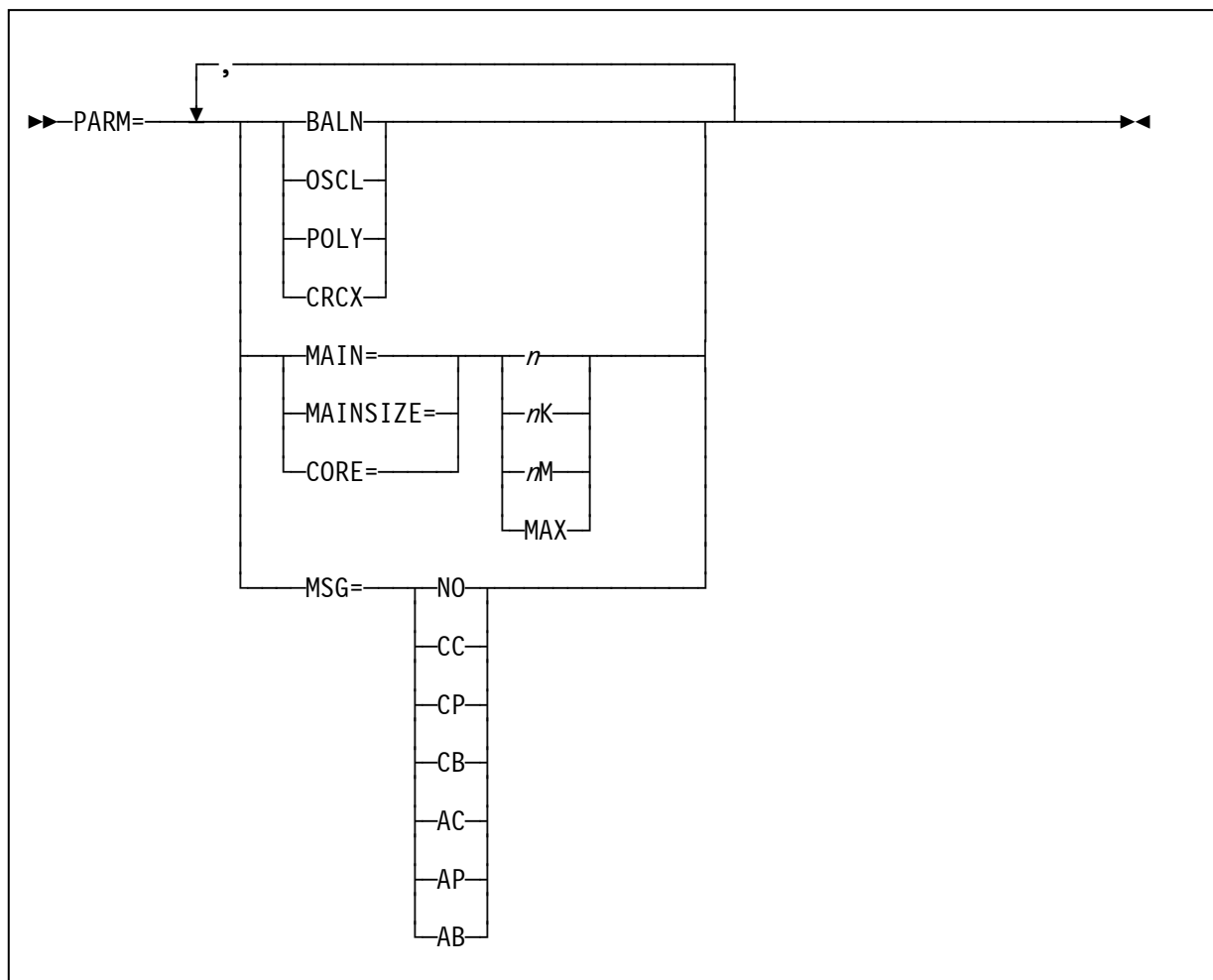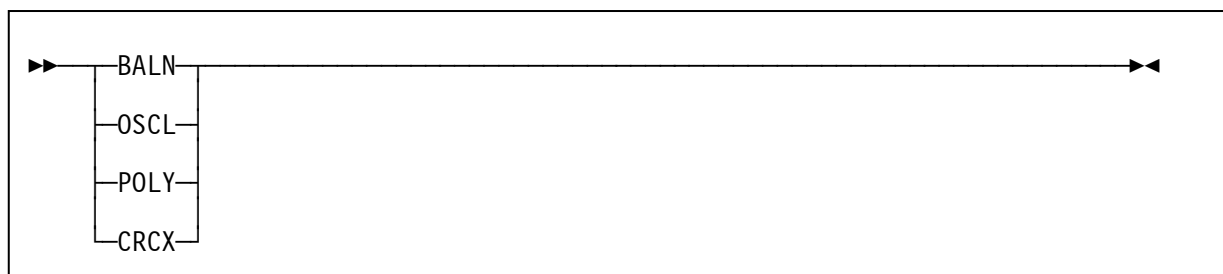


Figure 25 EXEC Statement PARM field Parameters

```
BALN or OSCL or POLY or CRCX
```



Specifies a sequence distribution technique to be used by the Sort/Merge Program. If the intermediate storage medium is tape, BALN means use the balanced tape technique, OSCL means use the oscillating tape technique, and POLY means use the polyphase tape technique. If the intermediate storage medium is DASD then BALN means use the balanced direct access technique and CRCX means use the crisscross direct access technique.

There are restrictions on the selection of technique if DASD intermediate storage has been selected:

- If less than six work areas are provided, then only the balanced technique is used.
- If more than six work areas are provided, then only the crisscross technique can be used.
- If exactly six work areas are provided, the balanced technique is used unless CRCX is specified.

Caution is recommended when forcing the use of a specific technique. The Sort/Merge Program determines the most efficient technique for a given application. If another technique is forced then performance might not be as efficient. Refer to Table 2 in Chapter 1 for information about the requirements for the different sequence distribution techniques.

## MAIN or MAINSIZE or CORE
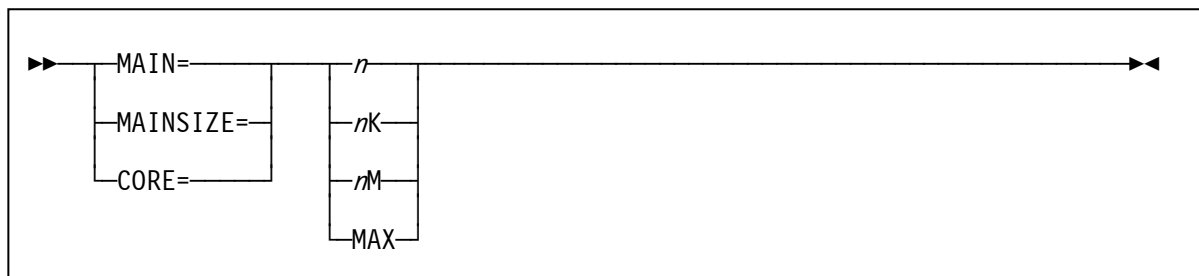


Optional main storage value which will temporarily override the Sort/Merge Program's storage default values set at installation customization time.

### $n$

specifies the maximum storage to be used in bytes

### $n$K

specifies the maximum storage to be used as a number of kilobytes

### $n$M

specifies the maximum storage to be used as a number of megabytes.

### MAX

specifies all available storage up to the limit specified in the MAXLIM installation customization setting is available for use.

### MSG



---

The MSG parameter can be used to temporarily override the message option specified at installation customization time. Valid entries are:

| Parameter | Description |
| --- | --- |
| NO | No messages are generated. |
| CC | Critical messages only are generated. They appear on the system console. |
| CP | Critical messages only are generated. They are written to the SYSOUT message data set. |
| CB | Critical messages only are generated. They appear on the system console and are written to the SYSOUT message data set. |
| AP | All messages are generated. They are written to the SYSOUT message data set. |
| AC | All messages are generated. They appear on the system console. |
| AB | All messages are generated. They appear on the system console and are written to the SYSOUT message data set. |

## 3.2   Specifying JCL DD Statements

A sort or merge job always requires JCL DD statements in the system input stream after an EXEC statement. Some of the JCL DD statements can be provided by the use of cataloged procedures as previously discussed. If the relevant Sort/Merge cataloged procedure is not used to invoke the Sort/Merge Program then the required JCL DD statements must be included in the system input stream.

The DD statements that would usually be contained in the cataloged procedure are:

SORTLIB    Defines the load module library containing the load modules for the assignment and run phases of the Sort/Merge Program This statement is always required.

SYSOUT    Sort output message data set. This statement is always required unless all sort messages have been turned off using the MSG=NO option. The default DD name of SYSOUT can be changed at the time of installation customization or by the OPTION control statement parameter MSGDDN.The DCB RECFM parameter is set to FBA and the LRECL is set to 121. The BLKSIZE can be set to any value which is a multiple of the LRECL value of 121.

SYSPRINT    Defines the message data set used by the linkage editor. This statement is only required when user-provided exit routines that require link editing are included in the Sort/Merge job.

SYSUT1    Linkage editor work data set. This statement is only required when user-provided exit routines that require link editing are included in the Sort/Merge job.

SYSLIN    Linkage editor input control data set. This statement is only required when user-provided exit routines that require link editing are included in the Sort/Merge job

SYSLMOD    Linkage editor load module output data set. Used to store the user exit load modules. This statement is only required when user-provided exit routines that require link editing are included in the Sort/Merge job

The following JCL DD statements are required whether the Sort/Merge Program is initiated directly or through a cataloged procedure:

SORTIN        Defines the input data set for a sorting job. Not required for a merging job. A SORTIN DD statement is required for all sorting operations unless an E15 user exit is used to supply all input records together with a RECORD control statement included in the control statement input stream. The DCB RECFM must specify fixed or variable-length records either on the JCL DD statement, the data set DASD VTOC entry or the tape label. Undefined format records are not supported.

SORTIN01
..
SORTIN16        Defines the input data sets for a merging job. Always required for a merging job. The DCB RECFM and LRECL parameters must be the same for all the SORTINnn data sets except in the case of variable-length records where the DCB LRECL value can vary. The data set with the largest BLKSIZE (and LRECL for varying-length records) must be the SORTIN01 data set. Alternatively, a BLKSIZE parameter (and LRECL parameter for varying-length records) can be coded on the SORTIN01 JCL DD statement setting the BLKSIZE (and LRECL for variable-length records) to the largest value of all the input data sets.

SORTWK01
..
SORTWK32        Defines the intermediate storage data sets for a sorting job. Not required for a merging job. For DASD sorts the BALN technique requires a minimum of three data sets and a maximum of six data sets. The CRCX technique requires at least six data sets and a maximum of 17 data sets. Tape sorts can use up to maximum of 32 tape drives. When using DASD three or more large SORTWKdd data sets are preferable to a larger number of smaller data sets as the reduced control block requirements releases more storage to keep more records in storage. The data sets can be allocated in extents however no additional extents will be allocated during a sorting operation. Any DCB parameters coded will be ignored as EXCP programming is used to access these data sets. VIO data sets can be used for small volume sorts to reduce system overhead. Use of VIO is not recommended for medium and large volume sorting operations as the system paging data sets can become full resulting in the entire system being adversely impacted. Using the dynamic allocation facility to allocate SORTWKdd data sets avoids the need to provide JCL DD statements for SORTWKdd data sets.

SORTOUT        Defines the single output data set for sorting and merging jobs. All output records from a sorting or merging job will be written to this data set. A SORTOUT JCL DD statement is always required except when an E35 user exit is provided to receive all output records from a sorting job. The DCB characteristics must match that of the SORTIN or SORTINnn data sets with the exception of the BLKSIZE where a different value can be used to increase or decrease the blocking factor of the output data set compared to the blocking factor used for the input data.

SORTMODS        Defines a temporary partitioned data set large enough to contain all of the modification routines that appear in the input stream for a given application. If the routines are not in the input stream, this statement is not required. If the routines are in libraries, then the DD statements defining the libraries must be included.

SORTCKPT        Defines a data set for checkpoint records. If the checkpoint facility is not being used this DD statement is not required.

SYSIN        Defines the control statement input stream. The SYSIN data set, with SORT or MERGE control statements, must be provided except when a sorting job is invoked by another program or the input control stream is provided in the IERPARM data set. The SYSIN data set is ignored for program invoked sorting jobs. The format of the control statements are described in Chapter 2: How to Use the Sort/Merge Program. The data set must be defined with a RECFM of F or FB and the LRECL must be 80. Any BLKSIZE which is a multiple of 80 is accepted.

---

SORTCNTL    Defines the optional control statement override stream for program invoked sorting jobs. The format of the control statements are described in Chapter 2: How to Use the Sort/Merge Program. The data set must be defined with a RECFM of F or FB and the LRECL must be 80. Any BLKSIZE which is a multiple of 80 is accepted.

IERPARM    Defines the optional control statement stream to override all previously provided control statements. The format of the control statements are described in Chapter 2: How to Use the Sort/Merge Program. The data set must be defined with a RECFM of F or FB and the LRECL must be 80. Any BLKSIZE which is a multiple of 80 is accepted.

SORTDIAG    Defines the output message data set for all diagnostic messages. Not for general use. More fully described in the document OS/360 Sort/Merge for MVS 3.8 Installation, Customization and Diagnosis Guide.

## Required JCL DD Statement Parameters

The Sort/Merge Program requires that certain parameters be included in the JCL DD statements described above. These parameters, the conditions under which they are required, a summary of the information contained in them, and the value assumed (default) if the parameter is not included are shown in Table 4. The parameters and sub-parameters which are not required are not discussed.

Table 4 JCL DD Statement Parameters used by the Sort/Merge Program

| Parameter | When Required | Parameter Values | Default Value |
|---|---|---|---|
| DSN or DSNAME | When the JCL DD statement defines a labeled input data set (for example, SORTIN), or when the data set being created is to be kept or cataloged (for example, SORTOUT), or passed to another step. | Specifies the fully qualified or temporary name of the data set. | The system assigns a unique name. |
| DCB | When tape is being used for the input or output data set. | Specifies information to complete the DCB associated with the data set. | |
| UNIT | When the input data set is neither cataloged nor passed or when the data set is being created. | Specifies (symbolically or actually) the type and quantity of I/O units required by the data set. | |
| SPACE | When the JCL DD statement defines a new data set on DASD. | Specifies the amount of space needed to contain the data set. | |
| VOL or VOLUME | When the input data set is neither cataloged nor passed, for multi-reel input or when the output data set is on DASD and to be kept or cataloged. | Specifies the volume or volumes containing the data set. | |
| LABEL | When the default value is not applicable. | Specifies information about labeling and retention status of the data set. | The system assumes standard labeling. |
| DISP | When the default value is not applicable. | Indicates the status and disposition of the data set. | The system assumes (NEW,DELETE). |

Table 5 DCB Sub-parameters used by the Sort/Merge Program

| Sub-parameter | When required | Parameter Value | Default Value |
|---|---|---|---|
| RECFM | When the DCB parameter is required | Specifies the format of the records in the data set. | |
| BLKSIZE | When the DCB parameter is required. | Specifies the maximum length (in bytes) of the physical records in the data set. | |
| LRECL | When the DCB parameter is required. | Specifies the maximum length (in bytes) of the logical records in the data set. | |

## 3.3   JCL and Sort/Merge Statement Examples

Following are a number of examples showing all the JCL and Sort/Merge control statements required to complete a sort or a merge operation.

### Example 1

| | |
|---|---|
| INPUT | Blocked fixed-length records |
| OUTPUT | Blocked fixed-length records |
| INTERMEDIATE STORAGE | Three dynamically allocated 3390 DASD data sets |
| USER EXIT | None |
| OPTIONS | DYNALLOC |

```
REVEDIT  SYSD.SORTMVS.CNTL(SAMPJCL1) - 1.00              COLUMNS 00001 00072
COMMAND ===>                                                SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 //T1S1     JOB SORTPROJ,TJA,MSGCLASS=C,CLASS=A
000002 //SORT     EXEC PGM=IERRCO000
000003 //SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
000004 //SYSOUT   DD  SYSOUT=*
000005 //SORTIN   DD  DSN=PROD39.F39001,DISP=OLD
000006 //SORTOUT  DD  DSN=PROD39.F39002,UNIT=3390,VOL=SER=PROD00,
000007 //             DISP=(,CATLG),SPACE=(TRK,(20,10)),
000008 //             DCB=(RECFM=FB,BLKSIZE=27750,LRECL=250)
000009 //SYSIN    DD  *
000010 *
000011 *          SEQUENCE INPUT DATA ON PRODUCT CODE AND DATE (DDMMYY)
000012 *
000013      SORT FIELDS=(1,10,A,15,2,A,13,2,A,11,2),FORMAT=CH
000014      RECORD TYPE=F,LENGTH=(250)
000015      OPTION DYNALLOC=(3390,3),SIZE=E20000
000016 /*
```

Figure 26 Sort JCL Example 1

| Line | Explanation |
|------|-------------|
| 000001 | JOB statement to identify the job to the Operating System. The parameters must conform to the standards in effect at the installation for the JOB statement to be successfully processed. |
| 000002 | EXEC statement. The Sort/Merge Program is called directly by its name IERRCO00. The alias of SORT can also be used. |
| 000003 | SORTLIB DD statement. Defines the load module library containing the load modules for the assignment and run phases of the Sort/Merge Program |
| 000004 | SYSOUT DD statement. Defines the output stream for messages. Directed to the output class set for the job. |
| 000005 | SORTIN DD statement. The input data set is named PROD39.F39001 and is cataloged. The DCB RECFM parameter must be F or FB and the LRECL must be 250 to match the values coded on the RECORD statement. |
| 000006-000008 | SORTOUT DD statement. The sorted output is written to a data set named PROD39.F39002 placed on 3390 DASD volume PROD00 and cataloged. The DCB parameters must match the DCB parameters coded on the SORTIN DD statement except for the BLKSIZE parameter which can be different. |
| 000009 | SYSIN DD statement. Provides the Sort/Merge control statements. |
| 000010-000012 | Comment statement. An asterisk in column one identifies the statement as a comment statement. |
| 000013 | SORT control statement. The product code in the data is 10 bytes in length starting in the first byte of the record and will be sequenced into ascending order. The following date field with format DDMMYY has been sorted as three separate fields to sequence the date in an ascending order |
| 000014 | RECORD control statement. TYPE=F identifies fixed-length records. The LENGTH of each record is 250 bytes. |
| 000015 | OPTION control statement. The DYNALLOC parameter is used to allocate 3 SORTWKdd data sets on 3390 DASD. The SIZE parameter has been specified to set the record count to an estimated 20,000. |

## Example 2

Sorting SMF records is a common job in all installations. However, the usual SMF record sort sequencing fields are not included in some of the short variable-length records generated by SMF. All fields selected as sort control fields must be present in every variable-length record processed. Attempting to sort these short records will result in an unsuccessful sorting operation. The problem can be addressed by implementing user exits to filter records selected for sorting and restoring the records to the data set after the sort has sequenced the selected records. Refer to the document OS/VS2 MVS SPL: System Management Facility for a more complete description regarding sorting SMF records.

| | |
|------|-------------|
| INPUT | SMF records written by the SMF dump program IFASMFDP |
| OUTPUT | SMF records sequenced and non-sequenced records passed to a following job step |
| INTERMEDIATE STORAGE | Three data sets allocated on 3390 DASD |
| USER EXIT | E15 and E35 |
| OPTION | DYNALLOC is turned off. Record sequence checking on output is turned off |

```
REVEDIT  SYSD.SORTMVS.CNTL(IVP3)  - 1.00              COLUMNS 00001 00072
COMMAND ===>                                          SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 //T1IVP3  JOB (001),'EXEC S/M IVP3',CLASS=A,MSGCLASS=C
000002 //SORTSMF EXEC PGM=SORT,REGION=2048K
000003 //SORTLIB  DD   DSN=SYS1.SORTLIB,DISP=SHR
000004 //EXITLIB  DD   DSN=SYSD.SORTMVS.EXITLIB,DISP=SHR
000005 //SYSOUT   DD   SYSOUT=*
000006 //SORTIN   DD   DSN=SMF.DAILY.DATA(0),DISP=SHR,
000007 //              DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=27000)
000008 //SORTWK01 DD   UNIT=3390,SPACE=(CYL,20)
000009 //SORTWK02 DD   UNIT=3390,SPACE=(CYL,20)
000010 //SORTWK03 DD   UNIT=3390,SPACE=(CYL,20)
000011 //SORTOUT  DD   DSN=&&SORTOUT,UNIT=3390,VOL=SER=WORK03,DISP=(,PASS),
000012 //              SPACE=(TRK,(100,20)),
000013 //              DCB=(RECFM=VB,BLKSIZE=27998,LRECL=27000)
000014 //HDRDATA  DD   UNIT=3350,SPACE=(TRK,(20,10)),
000015 //              DCB=(RECFM=VB,BLKSIZE=27998,LRECL=27000)
000016 //SORDATA  DD   UNIT=3350,SPACE=(TRK,(20,10)),
000017 //              DCB=(RECFM=VB,BLKSIZE=27998,LRECL=27000)
000018 //SYSIN    DD   *
000019     SORT  FIELDS=(19,16,A,11,4,A,7,4,A),FORMAT=BI,SIZE=E4000
000020     MODS  E15=(E15,60000,EXITLIB,N),E35=(E35,70000,EXITLIB,N)
000021     OPTION DYNALLOC=NO,NOVERIFY
000022     END
000023 /*
```

Figure 27 Sort JCL Example 2

| Line | Explanation |
|------|-------------|
| 00001 | JOB statement to identify the job to the Operating System. The parameters must conform to the standards in effect at the installation for the JOB statement to be successfully processed |
| 00002 | EXEC statement. The Sort/Merge Program is called directly by its alias name of SORT. |
| 00003 | SORTLIB DD statement. Defines the load module library containing the load modules for the assignment and run phases of the Sort/Merge Program. |
| 00004 | EXITLIB DD statement. Defines the load module library containing the previously assembled and link edited exit routines E15 and E35. |
| 00005 | SYSOUT DD statement. Defines the output message data set. Directed to the output class set by the MSGCLASS parameter on the JOB statement. |
| 00006-00007 | SORTIN DD statement. The input data set is the current generation of SMF.DAILY.DATA. The DCB BLKSIZE and LRECL parameters are overridden to provide a more meaningful description of the input records being sorted. The input records will be filtered by the E15 exit routine. |
| 00008-00010 | SORTWKdd DD statements .The data sets are allocated on 3390 DASD. |
| 00011-00013 | SORTOUT DD statement. The output from the sorting operation, plus records re-inserted into the output by the E35 exit routine, is passed to a subsequent job step. The DCB parameters RECFM, BLKSIZE and LRECL have been overridden to provide improved DASD space utilization. |
| 00014-00015 | HDRDATA DD statement. This data set is used by the E15 exit routine to hold records that are not sequenced and later re-inserted into the output by the E35 exit routine. |
| 00016-00017 | SORDATA DD statement. This data set is used to hold the all other system oriented records from the input that are not sequenced and later re-inserted into the output by the E35 exit routine. |

---

| Line | Explanation |
|------|-------------|
| 00018 | SYSIN DD Statement. Provides the control statements to the Sort/Merge Program. |
| 00019 | SORT control statement. SMF records are sequenced on job log identification, being the major control field, and then on time and date as minor control fields. The records are variable-length with an RDW adding 4 bytes to the start of each sort field. An estimated 4,000 records will be sequenced. |
| 00020 | MODS control statement. An exit routine is provided for the E15 exit point. The load module is resident in a library identified by the EXITLIB DD statement, requires approximately 60,000 bytes and does not need re-linking. An exit routine is provided for the E35 exit point. The load module is resident in the same library as the E15 exit routine, does not need relinking and requires 70,000 bytes. |
| 00021 | OPTION control statement. The DYNALLOC facility is not active as SORTWKdd data sets have been provided in the input job stream. As non-sequenced records are being re-inserted into the output sequence checking has been turned off by use of the NOVERIFY option. |
| 00022 | END control statement. As there is no user exit routines being provided in the SYSIN stream the END control statement is optional in this example. |

# 4. Invoking Sort from a Program

## 4.1 Introduction

The Sort/Merge Program can be dynamically invoked by any program that can issue service requests to the Operating System. The run time components of the PL/I and COBOL programming languages provide support specifically for invoking the Sort/Merge Program. These two programming languages also provide an interface between the programming language and the user exit facility of the Sort/Merge Program to pass records for sorting and retrieve records after they have been sorted.

Specifically, any programming language that provides the ability to issue the ATTACH, LINK or XCTL Operating System service requests can be used to invoke the Sort/Merge Program. Further information on the support provided by these various programming languages can be found in their respective programming guides. This chapter explains how the Sort/Merge Program can be invoked by an assembler program. The information provided regarding the JCL requirements will be relevant to all sorting operations invoked by any programming language.

The Sort/Merge Program can only be invoked by another program for sorting operations only. Merging operations are not supported.

## 4.2 Invoking the Sort Program

### Using ATTACH, LINK OR XCTL

The ATTACH, LINK or XCTL macro instruction issued by another program can be used to initiate a sorting application but not a merging application. For a full description of the ATTACH, LINK and XCTL Operating System services refer to the document MVS Supervisor Services and Macro Instructions.

There are four differences between initiating a sorting operation via the input stream and initiating it by a macro instruction:

- Sort/Merge Program JCL DD statements must be placed in the input stream with the job step that issues the macro instruction.
- Information normally contained on sort control statements can be passed to the Sort/Merge Program via a parameter list. The parameter list control statements can optionally be overridden with instream sort control statements.
- Only the E15 and E35 exit routines can be used when the Sort/Merge Program is initiated by an ATTACH, LINK or XCTL macro instruction.
- If ATTACH is used to initiate the sort then checkpoints cannot be taken.

The following tasks must be performed to initiate a sorting operation by invoking the Sort/Merge Program via an ATTACH, LINK or XCTL Operating System service request:

- Construct the sort control statements as character strings assembled as DC instructions in the invoking program.
- Build a parameter list containing:
  - addresses of the sort control statements
  - addresses of the user exit routines (if being used) to supply records to the sort and receive sequenced records from the sort
  - other optional parameter values being passed to the Sort/Merge Program.
- Write the assembler instructions to issue the selected system macro that will request the Operating System service to invoke the Sort/Merge Program.
- Code the optional user exit routines according to the interface rules defined in Chapter 5: Using Exit Routines.
- Assemble and link edit the invoking program and optional exit routines.
- Write the JCL DD statements required for the sorting operation.
- Submit the job stream to the Operating System for processing.

## 4.3   Supplying the Needed JCL DD Statements

When using ATTACH, LINK, or XCTL to invoke the Sort/Merge Program the following JCL DD statements must be included in the input stream with the job step that issues the macro instruction:

SORTLIB         Defines the load module library containing the load modules for the assignment and run phases of the Sort/Merge Program. This DD statement is always required.

SYSOUT          Sort output message data set. This DD statement is always required unless all sort messages have been supressed using the MSG=NO parameter. The default ddname of SYSOUT can be changed at installation customization time or by the OPTION control statement parameter MSGDDN.The DCB RECFM parameter is set to FBA and the LRECL is set to 121. The BLKSIZE can be set to any value which is a multiple of the LRECL value of 121.

SORTWK01        Defines the intermediate storage data sets for a sorting job. For DASD sorts the
..              BALN technique requires a minimum of three data sets and a maximum of six data
SORTWK32        sets. The CRCX technique requires at least six data sets and a maximum of 17 data sets. Tape sorts can use up to maximum of 32 tape drives. When using DASD three or more large SORTWKdd data sets are preferable to a larger number of smaller data sets as the reduced control block requirements releases more storage to keep more records in storage. The data sets can be allocated in extents however no additional extents will be allocated during a sorting operation. Any DCB parameters coded will be ignored as EXCP programming is used to access these data sets. VIO data sets can be used for small volume sorts to reduce system overhead. Use of VIO is not recommended for medium and large volume sorting operations as the system paging data sets can become full resulting in the entire system being adversely impacted. Using the dynamic allocation facility for the allocation of SORTWKdd data sets avoids the need to provide JCL DD statements for SORTWKdd data sets.

The following DD statements are optional depending on the user exits provided and the need to override control statements passed to the sort program by the invoking program

SORTIN          Defines the input data set for a sorting job. A SORTIN JCL DD statement is required for all sorting operations unless an E15 user exit is used to supply all input records together with a RECORD control statement to describe the records being supplied by the E15 user exit. The SORTIN DCB RECFM must specify fixed or variable-length records either on the JCL DD statement, the data set DASD VTOC entry or tape label. Undefined format records are not supported.

SORTOUT         Defines the single output data set for the sort. Output records from a sorting job will be written to this data set. A SORTOUT JCL DD statement is always required except when an E35 user exit is provided to receive all output records after sorting. The DCB characteristics must match that of the SORTIN data set or RECORD control statement with the exception of the BLKSIZE where a different value can be used to increase or decrease the blocking factor of the output data set compared to the blocking factor of the input data.

SORTCNTL        Defines the optional control statement override stream for program invoked sorting jobs. The format of the control statements are described in Chapter 2: How to Use the Sort/Merge Program. The data set must be defined with a RECFM of F or FB and the LRECL must be 80. Any BLKSIZE which is a multiple of 80 is accepted.

IERPARM         Defines the optional control statement stream to override all previously provided control statements. The format of the control statements are described in Chapter 2: How to Use the Sort/Merge Program. The data set must be defined with a RECFM of F or FB and the LRECL must be 80. Any BLKSIZE which is a multiple of 80 is accepted.

## 4.4   Overriding Sort Control Statements from the invoking Program

Sort control statements provided by the invoking program can be overridden or supplemented by control statements provided in the SORTCNTL input stream or alternatively by the IERPARM input stream. Caution is advised in overriding control statements provided by the invoking program. Overriding the RECORD control statement TYPE or LENGTH operands will almost certainly result in the Sort/Merge Program or the invoking program abending. However, overriding the LENGTH subfields when sorting variable-length records can result in a more optimum sorting operation. Overriding sort control fields on the SORT control statement can result in the invoking program not functioning correctly if records are returned in a different sequence than expected by the invoking program.

Using an OPTION control statement to override or set values such as the FILESZ and the DYNALLOC parameters can be of benefit in tailoring a sorting operation without making changes to the invoking program.

## 4.5   Passing Parameters to the Sort

The parameters passed to the Sort/Merge Program consist of control statement strings, addresses of exit routines and additional optional parameters to control the sorting operation. At least one control statement must be provided, usually the SORT control statement. Additional control statements and parameters can be provided in the SORTCNTL or the IERPARM input stream. Either the address of the E15 user exit routine or the address of the E35 exit routine or both or no exit routines can be provided to the Sort/Merge Program. If an exit routine is not going to be used then a value of zero must be provided in place of an address. No other user exit routines are permitted. All other parameters that can be passed to the Sort/Merge Program are optional.

The rules for coding control statements are as follows:

- The contents and format of the control statements are described in Chapter 2: How to Use the Sort/Merge Program.

- The control statements must be provided in EBCDIC character strings coded using assembler DC statements.

- The statement operation name (SORT or RECORD, for example) can be preceded by blanks. At least one blank must follow the statement operation name before the commencement of the operands. The operands can end with blanks but no blanks can be imbedded within the operands.

- Labels on control statements are not supported.

- No comment statements, blank statements or imbedded comments are allowed in the control statement character strings.

- Continuation statements are not permitted or needed as the entire control statement can be coded as one continuous string using multiple DC statements up to a length of 1,100 bytes.

The invoking program must construct the parameter list, the format is shown in Figure 28, and place its address in General Register 1 (GR1) before issuing the macro instruction to invoke the Operating System LINK, ATTACH or XCTL service. The parameter list consists of a 24 byte fixed section that must always be provided and an optional section of varying length depending on the parameters being passed to the Sort/Merge Program.

| Offset (Hex) (Dec) | Byte 1 | Byte 2 | Bytes 3 and 4 | Notes |
|---|---|---|---|---|
| -2    -2 | Unused | Unused | Length of parameter list in Bytes | 1 |
| 2    2 | X'00' | Starting Address of SORT Control Statement | | 2 |
| 6    6 | X'00' | Ending Address of SORT Control Statement | | 3 |
| A    10 | X'00' | Starting Address of RECORD Control Statement | | 3 |
| E    14 | X'00' | Ending Address of RECORD Control Statement | | 3 |
| 12    18 | X'00' | Address of E15 User Exit Routine | | 4 |
| 16    22 | X'00' | Address of E35 User Exit Routine | | 4 |

Register 1 → X'80' Address of parameter list

Figure 28 Required Sort Parameters

Figure 29 lists the optional parameters that can be passed to the Sort/Merge Program. These parameters can be provided in any sequence as their function is identified in the first byte of each parameter.

| Byte 1 | Bytes 2 - 4 | Notes |
|---|---|---|
| X'00' | Main storage value | 5 |
| X'01' | Reserved storage value | 6 |
| X'03' | Address of 8 character message DD name | 7 |
| X'05' | Starting Address of DEBUG Control Statement | 8 |
| X'00' | Ending Address of DEBUG Control Statement | |
| X'11' | Starting Address of OPTION Control Statement | 8 |
| X'00' | Ending Address of OPTION Control Statement | |
| Four character sequence distribution technique | | 9 |
| Four character prefix for SORT DD statements | | 10 |
| X'F7' | User exit address constant | 11 |
| X'FD' | Null parameter list entry | 12 |
| X'FF' | Message option | 13 |

Figure 29 Optional Sort Parameters

**Notes to Figure 28 and Figure 29**

| Note | Explanation |
|---|---|
| 1 | The length of the parameter list does not include the two bytes of the length field. This field must be on a half word boundary as the full word address of the first parameter, the address of the SORT control statement, must immediately follow the parameter list length count. |
| 2 | Required entry and must appear at the position in the parameter list as shown. The first byte of the address must be zero as this byte is tested to determine if it is a program invoked sort or a JCL invoked sort. If the SORT control statement is not provided then the address must be zero. |
| 3 | Required entry and must appear at the position in the parameter list as shown. If the control statement is not provided then both the starting address and the ending address must be zero. |
| 4 | Required entry and must appear at the position in the parameter list as shown. If the user exit routine is not going to be used then the address must be zero. |
| 5 | The maximum storage to be used by the Sort/Merge Program, expressed in bytes. Equivalent to the MAINSIZE parameter on an OPTION statement. |
| 6 | The reserved storage value if MAINSIZE=MAX is in effect. Equivalent to the RESINV parameter on the OPTION statement. |
| 7 | Overrides the JCL DD name of the message data set, usually SYSOUT. Equivalent to the MSGDDN parameter on the OPTION statement. |
| 8 | Optional control statement. If the parameter is provided then the starting and ending addresses must point to the beginning and end of the appropriate control statement. |
| 9 | This parameter directs the Sort/Merge Program to use a specific sequencing technique. The four valid entries are: BALN, CRCX, OSCL or POLY. Depending on other information and resources allocated to the sorting operation this direction to use a specific technique might be ignored without any message being provided. |
| 10 | Provides an override to the 4-character prefix used for sort JCL DD statements. Equivalent to the SORTDD parameter on the OPTION statement. Note that prefix values of BALN, CRCX, OSCL and POLY are not permitted as the Sort/Merge Program would not be able to determine if a sort sequencing technique or a JCL DD name override was being provided. |
| 11 | The value provided, with the X'F7' zeroed, is passed to the first exit routine call as the user exit word. Subsequent calls to either the E15 or the E35 exit routines are passed the 4 byte word updated by any previous calls to the E15 or E35 exit routines. Further information on this parameter is provided in Chapter 6: Using Exit Routines. |
| 12 | The three bytes are ignored. This parameter can be used to set a placeholder in the parameter list where parameter values or addresses can be inserted at a later time by the invoking program prior to the Sort/Merge Program being invoked. This parameter can appear multiple times in the parameter list. |
| 13 | This parameter overrides the installation customization MSGCON and MSGPRT parameters. Valid values are NO, AP, AC, CC, CP, CB or AB right justified in the three byte field. Any value can be coded in the first byte of the field as it is not checked. The explanation of the values is provided in Chapter 3: Invoking the Sort/Merge Program with JCL Language, in the topic Specifying PARM Options. |

# 5.     Considerations when using ATTACH, LINK or XCTL

If an exit routine is provided for the E15 exit point then the SORTIN data set is ignored. All input records into the sorting operation must be provided by the E15 exit routine. The E15 exit routine is therefore restricted to issuing a return code of 12 (insert record) until the exit routine has determined that there are no more records to be inserted for sorting.

When all the records have been inserted a return code of 8 must be issued to signal the completion of the record insertion process. The Sort/Merge Program can now begin sequencing the records inserted by the E15 user exit.

Similarly, the SORTOUT data set is ignored if an E35 exit routine has been provided. The E35 exit routine must accept each record from the Sort/Merge Program returning with a return code of 4 signaling the record is to be discarded. When there are no more records remaining to be passed to the exit routine then the E35 exit routine must return with a return code of 8 to signal completion of the sorting operation.

If the Operating System XCTL service is going to be used to invoke the Sort/Merge Program then special consideration must be given to where, in storage, the invoking program provides the parameter list, sort control statements, user exit routines (if used) and optional parameters.

These components, required to invoke the Sort/Merge Program, must not be resident in the invoking program. When the Operating System processes a XCTL service request the program that issued the XCTL request is deleted from storage. Therefore, the storage containing the required components would no longer be available to the Sort/Merge Program. The Sort/Merge Program would fail when attempting to access the no longer available storage. If the XCTL service request is being used then the invoking program must issue a GETMAIN request to obtain storage. This storage can be used to establish the parameter list, sort control statements, user exit routines (if used) and optional parameters prior to issuing the XCTL service request. This storage will remain available after the invoking program has been deleted from storage.

## 5.1  Parameter Example

```
REVEDIT  SYSD.SORTMVS.CNTL(IVP1) - 1.00                    COLUMNS 00001 00072
COMMAND ===>                                               SCROLL ===> CS
000128 *
000129 *         LINK TO THE SORT
000130 *
000131         LA    R1,SORTPTR            R1 -> PTR TO SORT PARMS
000132 *
000133         LINK  EP=IERRCO00,MF=(E,(1))
000134 *
- - - - - - - - - - - - - - - - - - - - - -  287 LINE(S) EXCLUDED
000422 *
000423 *         SORT PARAMETER LIST
000424 *
000425         DC    0F'0'
000426 SORTPTR  DC    X'80'
000427         DC    AL3(SORTPRM)
000428         DC    H'0'                 FORCE CORRECT ALIGNMENT
000429 SORTPRM  DC    AL2(SORTPRME-SORTPRMS) L'PARAMETER LIST
000430 SORTPRMS DC    A(SORTSTM)           -> SORT CONTROL STATEMENT
000431         DC    A(SORTSTME)
000432         DC    A(RECSTM)            -> RECORD CONTROL STATEMENT
000433         DC    A(RECSTME)
000434         DC    A(E15)               -> E15 EXIT
000435         DC    A(E35)               -> E35 EXIT
000436         DC    C'IVP1'              SORT DDNAME PREFIX
000437         DC    X'03',AL3(MSGDD)     -> SORT MESSAGE OUTPUT DD NAME
000438         DC    X'FF',C' AP'         SET MSG OPTIONS
000439 SORTPRME EQU   *                   FOR L'PARAMETER CALC
000440 *
000441 SORTSTM  DC    C'SORT FIELDS=(10,15,CH,D)'
000442         DC    C',SIZE=E5000'
000443 SORTSTME DC    C' '
000444 RECSTM   DC    C'RECORD LENGTH=200,TYPE=F'
000445 RECSTME  DC    C' '
000446 MSGDD    DC    CL8'IVP1MSG'         DD NAME FOR SORT MESSAGES
000447 *
```

Figure 30 Sort Parameter List Example

The example in Figure 30 shows the length of the parameter list being automatically calculated by the assembler making future changes to the parameter list easier to implement. Following the two byte length count is the fixed 24 byte section of the parameter list providing the beginning and ending addresses of the SORT and RECORD control statements followed by the addresses of the E15 and E35 exit routines. Optional parameters have been provided to:

- set the DD name prefix to IVP1 for the sort data sets.

- set the DD name of the message data set to IVP1MSG.

- set the message option to all messages to the message data set which would be IVP1MSG in this example.

# 6. Using Exit Routines

## 6.1 Introduction

User-written routines can be used during a Sort/Merge Program operation to perform a variety of functions, such as inserting, discarding, altering, and summarizing records.

Control is passed to these exit routines at pre-designated places in the Sort/Merge Program, referred to as Sort/Merge Program exits. Because these exits are located in particular program phases (and in one case, in a particular module), a general understanding of how the Sort/Merge Program operates is necessary to understand the various environments in which the user exit routines are given control.

This chapter discusses and provides examples of user exits written in assembler language. Exit routines can be written in any programming language that provide the following interface requirements:

- Accept and pass addresses of records or parameter lists in General Register 1,
- Construct and access parameter lists of varying length,
- Set a return code in General Register 15 prior to returning control to the Sort/Merge Program.

## 6.2 Program Description

The Sort/Merge Program is a segmented program; that is, it is composed of parts that operate independently. Generally, there are two levels of segmentation:

1. Phases – groups of modules that accomplish a certain task.

2. Modules - independent routines that are selected to form a phase.

The Sort/Merge Program is composed of five phases. All five phases are used for sorting applications, but only the first two and the last phases of the Sort/Merge Program are used for merging applications. The first two phases, the definition and optimization phases, are strictly initialization phases. Each of the remaining three phases, the sort, intermediate merge, and final merge, are divided into two components:

1. An assignment component that initializes for the operation of the phase.

2. A running component that performs the actual sorting or merging operation.

Figure 31 provides a phase-level flowchart of the Sort/Merge Program as it progresses through a sorting or merging operation. An explanation of each phase follows the flowchart.

```
            ┌─────────────────────┐
            │  Definition Phase   │
            └─────────────────────┘
                      │
                      ▼
                  ╱───────╲
                 ╱  User    ╲
                ╱  Routines   ╲        No
                ╲  to be      ╱────────────┐
                 ╲ Link Edited?╱           │
                  ╲───────────╱            │
                      │                    │
                      ▼                    │
            ┌─────────────────────┐        │
            │   Linkage Editor    │        │
            └─────────────────────┘        │
                      │◄───────────────────┘
                      ▼
            ┌─────────────────────┐
            │ Optimization Phase  │
            └─────────────────────┘
                      │
                      ▼
                  ╱───────╲
                 ╱  Sort    ╲        Merge
                ╱   Or        ╲───────────┐
                ╲  Merge      ╱           │
                 ╲───────────╱            │
                      │                   │
                      ▼                   │
            ┌─────────────────────┐       │
            │     Sort Phase      │       │
            └─────────────────────┘       │
                      │                   │
                      ▼                   │
            ┌─────────────────────┐       │
            │Intermediate Merge Phase│    │
            └─────────────────────┘       │
                      │◄──────────────────┘
                      ▼
            ┌─────────────────────┐
            │  Final Merge Phase  │
            └─────────────────────┘
```
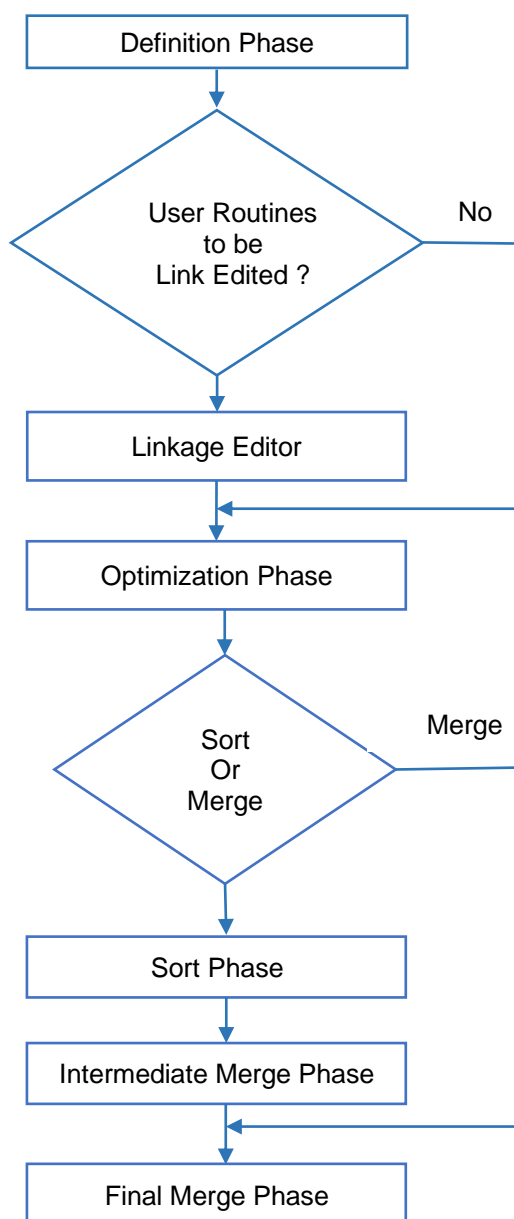
Figure 31 Sort Program Phase Flowchart

## Definition Phase

The definition phase reads and interprets Sort/Merge control statements and decides which phases, and which modules of each phase, should be used. This phase also decides which of the exit routines, if any, must be link edited. This phase has no user exit points.

## Optimization Phase

The optimization phase, using information obtained from the Operating System and from JCL DD statements, determines the optimum method of using the processor, storage and the I/O configuration available for a particular sorting operation.

This phase generates special routines, if necessary, to perform record comparisons. One of two routines, the equals module or the extract module, can be generated to make record comparisons. Neither routine is used when sorting or merging is based on a single control field containing character data or binary data beginning and ending on a byte boundary. If either of these routines are used, the routine remains in storage throughout the running of the sorting or merging operation.

## Equals Module

The equals module is used when there are from two to twelve control fields and all control fields contain character data or binary data beginning and ending on a byte boundary. It is invoked to resolve the collating of records when an equal comparison arises between two major control fields. This is done by comparing successive minor control fields until an unequal comparison is made, thus determining the proper order of the two records. If all control fields are equal, the records are taken in the order which requires the least internal processing time. The order of input might not be preserved.

## Extract Module

The function of the extract module is to extract and group all of the control fields into one field so that a single compare instruction can be used to collate the record.

The extract module is loaded for one of two reasons:

- If there is more than one control field and the equals module cannot be used to resolve the collation order.
- If specified by the user in either the SORT or the MERGE control statement.

User specification is accomplished by using the E option for the s parameters of the FIELDS operand. See Chapter 2: How to Use the Sort/Merge Program for further information.

When the extract module is used, it is invoked each time a logical record is processed. This is done to avoid carrying the extracted information with the records, which would increase I/O time and therefore, total sort or merge resource usage.

## Sort Phase

The sort phase orders the records in the input data set into sequences and distributes these sequences out to the intermediate storage data sets. The method of distribution is determined by the sequence distribution technique that was selected for the sorting operation.

If tape is being used for intermediate storage, the sequences can be written out in both ascending and descending order. This enables the intermediate merge phase, using the tape read-backward feature, to merge the sequences without rewinding tapes.

If DASD intermediate storage is used, the sequences are distributed among the data sets assigned to the Sort/Merge Program so that they can be merged in a minimum number of passes.

This phase has a number of exits at which control can be passed to exit routines.

## Intermediate Merge Phase

The intermediate merge phase is loaded and run following completion of the sort phase. It performs successive merges of the strings produced by the sort phase. The merges are carried out from intermediate storage device to intermediate storage device, each successive merge decreasing the number of strings and increasing the average string length. When one more merge is required to create one long string, the output data set, control is given to the final merge phase. There are several exits in this phase at which exit routines can receive control.

## Final Merge Phase

The final merge phase has two uses:

1. Making the final merge pass of a sorting application, thus creating the output data set.
2. Merging the input data sets for a merging application to create the output data set.

Records output from this phase can be written to any output device supported by QSAM. After the running of this phase, the sort system control component returns control to the Operating System via the RETURN macro instruction. Exit routines can receive control at a number of exits in this phase.

## General Information

There are two types of exits provided by the Sort/Merge Program.

1. Assignment component exits, one each for the sort, intermediate merge, and final merge phases.
2. Running component exits, a number of which are associated with the running component of each program phase.

Assignment component exits can be used to initialize exit routines in each phase or to open data sets needed by exit routines. Assignment components are transitory and their storage areas are reused by the running components. The exit routines at assignment component exits are also transitory unless they are link edited together with the other routines in the phase.

Running component exit routines can be used for a variety of purposes including the insertion, discarding, summarization, or any other alteration of the records coming into or out of the phase. Running component exit routines can also be used to correct some of the errors that can occur during program operation. This includes I/O errors and exceeding Nmax, the maximum record count. These exits provide an opportunity to close any data sets used by other routines. The running component extract module exit, E61, can be used to alter control fields before the Sort/Merge Program sequences them. It is this exit that, for example, would be used to normalize floating-point control fields prior to sequencing.

Figure 14 is a summary of the Sort/Merge Program exits and their uses. The first digit of the exit number represents the phase in which the exit is located:

- 1 for the sort phase.
- 2 for the intermediate merge phase.
- 3 for the final merge phase.

The second digit represents the type of function the routine can perform at the exit.

## Efficiency Considerations

When considering the use of user exit routines the following factors should be considered:

- Exit routines occupy storage that would otherwise be available to the Sort/Merge Program. This can result in the need for additional intermediate merge phase passes. This, of course, increases sorting time.
- Exit routines can add time to the overall Sort/Merge Program running time. Later, in the descriptions of exits, note that most of the exit routines gain control once for each record until the "do not return" return code is passed to the Sort/Merge Program. Modification routines should be designed with this in mind.

## 6.3  Exit Routine Programming

Table 6 Summary of Functions Permitted at each Exit

| Possible Use for Exit | Sort | | | | | | Final Merge | | | | | Extract | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E11 | E15 | E16 | E17 | E18 | E19 | E21 | E27 | E28 | E29 | E31 | E35 | E37 | E38 | E39 | E61 |
| Assignment | X | | | | | | X | | | | X | | | | | |
| Nmax Error | | | X | | | | | | | | | | | | | |
| Logical Record Change | | X | | | | | | | | | | X | | | | |
| Control Field Change | | | | | | | | | | | | | | | | X |
| Opening Data sets | X | | | | | | X | | | | X | | | | | |
| Closing Data sets | | | | X | | | | X | | | | | X | | | |
| Read Error Routine | | | | | X | | | | X | | | | | X | | |
| Write Error Routine | | | | | | X | | | | X | | | | | X | |

### Bypassing the Linkage Editor

To save run time resources, the exit routines should be designed so that they do not require link editing each time they are used in a sorting or merging application. To avoid the use of the linkage editor at run time, the routines must meet the following requirements:

- Each routine must be a load module in a load module library. Its member name must be the same as its exit number, for example, E16. The value *s* on the MODS statement that defines the routine must be the name of the DD statement that defines the library. For example:
  MODS E16=(E16,500,MODLIB,N)
  //MODLIB DD DSN=SORT.LIB,etc.
- Each routine must have only one entry point which must also be the module name.
- The routines cannot have external references that require resolution before running.
- All routines must be on the same library or all the libraries must be concatenated with one DD name to identify all the concatenated libraries.

The parameter N on the MODS statement should be coded for each routine that meets the previously listed requirements. This indicates that the routine was previously link edited and does not require further link editing.

If routines at assignment exits (E11, E21, and E31) do not meet the requirements for bypassing the linkage editor, then processing time can still be saved by designing them for separate link editing. To be eligible for separate link editing, the assignment component routines must meet the following requirements:

- Each routine must be separate.
- The routines cannot contain external references.
- The routines can have several entry points, but one entry point must be the same as the exit number e.g. E11.
- The routine must be designed so that it can be overlaid after assignment time.

To indicate that the routine is eligible for separate link editing, code the parameter S for that routine on the MODS statement. If the routine opens data sets or communicates with running component routines, it will contain external references and therefore cannot be link edited separately.

If the routine does not meet the requirements for bypassing the linkage editor or for separate link editing, do not code a fourth parameter for that routine on the MODS statement. The routine is then link edited together with all other routines in its phase which do not meet the requirements. In any phase, routines can be mixed that do not require additional link editing, routines that can be link edited separately, and routines that must be link edited together.

## Operating Considerations

Each of the exit routines must be assembled or compiled as a separate program and placed in either a load module library by the linkage editor or as an object deck in the system input stream. The Sort/Merge Program general assignment phase includes the names and locations of the exit routines in the list of modules to be run during each program phase. The exit routines are loaded and run with their associated program phase.

No exit routine can appear more than once in a program phase, but the same routine can appear in several phases. For example, the same read error exit routine can be used in all three phases, but not twice in any one phase. If an exit routine is to be used more than once and the exit routines are provided as an object deck in the SYSIN stream, then a copy of the exit routine must be supplied for each use.

Only one load module is allowed at each Sort/Merge Program exit. If more than one routine is needed at an exit point, then the routines must be assembled or compiled, and link edited as one load module.

All routines in a phase that require link editing can be placed in one partitioned data set member. The member must have an entry point for each of the routines. When the routines are arranged in one member, their individual lengths specified on a MODS statement are not important, but the sum of the lengths must be the total length of the module. All but one length can be specified as zero, with the total member length specified for the remaining routine.

## Routines in the SYSIN input stream

The routines, in object deck format, placed in the SYSIN input stream are copied to the SORTMODS data set. They then become input to the linkage editor.

## Exit Routine Linkage Conventions

The Sort/Merge Program follows standard Operating System conventions when it invokes the exit routines. The called exit routine must save and restore all general registers it uses in the routine except for the parameter return registers. The general registers used by the Sort/Merge Program for linkage and communication of parameters are:

| Register | Usage |
|---|---|
| 1 | Contains the address of the parameter list passed to the exit routine. Not used on some exits |
| 13 | Address of a standard Operating System 18 word save area |
| 14 | Return address to the Sort/Merge Program |
| 15 | On entry, the address of the exit routine. On exit, used to communicate a return code to the Sort/Merge Program |

The Sort/Merge Program uses a CALL macro instruction expansion to call exit routines. The exit routines can return control to the Sort/Merge Program with a RETURN macro instruction using the macro to restore the Sort/Merge Program's registers from the provided save area. The RETURN macro instruction can be used to set return codes when multiple actions are available on return from an exit.

All routines must contain an entry point defined by an ENTRY or CSECT statement. The name of the entry point must be the number of the associated Sort/Merge Program exit prefixed by the letter E.

## Linkage Example

The registers at entry into an exit routine are described in the previous Exit Routine Linkage Conventions section. The coding for an E15 exit routine can include the following assembler instructions.

```
REVEDIT  SYSD.SORTMVS.CNTL(E15) - 1.00                 COLUMNS 00001 00072
COMMAND ===>                                             SCROLL ===> CS
000001 E15       TITLE 'E15 - SORT/MERGE USER EXIT'
000002 *
000003 E15       CSECT
000004 *
000005           SAVE  (14,12),,'E15 &SYSDATE &SYSTIME'
000006           LR    R12,R15                 SET UP BASE
000007           USING E15,R12                 DECLARE BASE
000008           ST    R13,SAVEAREA+4          CHAIN SORT & E15 SAVE AREA
000009           LR    R2,R13
000010           LA    R13,SAVEAREA            PRIME SAVE AREA ADDR
000011           ST    R13,8(,R2)              CHAIN E15 & SORT SAVE AREA
- - - - - - - - - - - - - - - - - - - - - - 29 LINE(S) EXCLUDED
000041 *
000042 E15DELET L     R13,SAVEAREA+4          ADDRESS OF SORT SAVEAREA
000043 *
000044           RETURN (14,12),RC=4          DISCARD RECORD
000045 *
- - - - - - - - - - - - - - - - - - - - - - - 60 LINE(S) EXCLUDED
000106           END
```

Figure 32 Exit Routine Sample Linkage

The SAVE macro is used to save the contents of the Sort/Merge Program's registers. Addressability for the E15 exit routine is established. The save areas are then chained according to the standard Operating System convention. After the exit routine has finished processing the record the Sort/Merge Program's registers are restored by use of the RETURN macro. A return code of 4 is set prior to the return to the Sort/Merge Program.

A full explanation of the Operating System's linkage conventions and the macro instructions used to implement the conventions can be found in the document MVS Supervisor Services and Macro Instructions.

## Assignment Component Exits

| | | |
|---|---|---|
| **Exit/Phase Used** | **E11** | Sort phase |
| | **E21** | Intermediate merge phase |
| | **E31** | Final merge phase |
| Possible use of Exit | Use routines at these exit points to open data sets needed by other routines in the associated phases or to initialize other routines | |
| Information supplied to the exit routine | None | |
| Return Codes | None | |
| Linkage Conventions | Standard, no parameters passed in GR1 | |
| Further Considerations | These are the only three routines that can be link edited separately. Refer to the topic Bypassing the Linkage Editor earlier in this chapter. | |

## Running Component Exits

Each Sort/Merge Program phase has a number of running component exits associated with the phase. Many of these exits perform the same function in each of the Sort/Merge Program's three phases. These are explained in the following text.

## Record Change Exits

The record change exits can be used to insert, discard, alter or summarize records.

| | | |
|---|---|---|
| **Exit/Phase Used** | **E15** | Sort phase prior to any records being sorted |
| Possible use of Exit | | Insert records into the sort, create the entire input into the sort, discard records read from the input data set, change records read from the input data set. |
| Information supplied to the exit routine | | The E15 exit is called by the Sort/Merge Program each time a record is brought into the sort. On entry into the E15 exit GR1 has the address of a parameter list. The parameter list is two words long. The first word is the address of the new record. The second word is the user exit address constant. The format of the parameter list is: |

| | |
|---|---|
| | Address of the new record |
| User exit address constant | |

When the Sort/Merge Program reaches end of file on the input data set then the new record address is set to zero. If there are no records in the input data set then the new record address is set to zero on the first call to the exit.

The User exit address constant can be updated by the exit. The updated value will be passed to every subsequent call to the E15 or the E35 exit routine.

Return Codes — The exit routine must return to the Sort/Merge Program with one of the following return codes in GR15 to control the disposition of the record.

| Return Code | Disposition |
|---|---|
| 0 | No action, accept record into the sort. |
| | Place the address of the record passed in the parameter list into GR1. |
| | Alter the record before accepting the record into the sort. |
| | If the exit needs to change the record before it is processed by the sort then the record must be moved to a work area, modified and the address of the modified record returned in GR1. If the routine changes the length of the record then the RECORD control statement must identify the new record length. |
| 4 | Discard the record. |
| | The value of GR1 is not used. |
| 8 | Do not return. |
| | The Sort/Merge Program no longer calls the exit routine and begins sequencing the records inserted into the sorting operation. Unless records are being inserted after the end of the input data set then a return code of 8 must be returned when a record address of zero is passed to the exit. The value of GR1 is not used. |

---

| Return Code | Disposition |
|---|---|
| 12 | Insert a record. |

If the exit routine needs to insert a record into the input record stream before the record whose address is provided in the input parameter list then place the address of the record to be inserted into GR1. The Sort/Merge Program then returns to the exit with the same record address, as previously provided, in the input parameter list. Additional records can be inserted at that point in the input stream or the exit can control the disposition of the record by returning with the appropriate return code. Record insertions can be made after the Sort/Merge Program has signaled the end of the input data set. The Sort/Merge Program will continue to call the exit routine until a return code of 8 is received.

| | |
|---|---|
| Linkage Conventions | Standard. |

On entry GR1 contains the address of the parameter list passed to the exit.

On exit GR1 contains the address of a record if the record is being accepted, altered or inserted otherwise it is not used.

| | |
|---|---|
| Restrictions | If ATTACH, LINK or XCTL is used to invoke the Sort/Merge Program and the E15 exit is provided then the SORTIN data set is ignored. The E15 exit routine is the sole source of input records for sorting. This is achieved by continually using the Insert Record return code to insert records for sorting until all records have been inserted. The Do Not Return value is then passed to the Sort/Merge Program to signal completion of the input process. |

| | | |
|---|---|---|
| **Exit/Phase Used** | **E35** | Final merge phase after the records have been merged |

| | |
|---|---|
| Possible use of Exit | Add records into the SORTOUT data set, discard records from the sort, or change records prior to the records being written to the SORTOUT data set. |

| | |
|---|---|
| Information supplied to the exit routine | The E35 exit is called by the Sort/Merge Program prior to every record being written to the SORTOUT data set. On entry to the E35 exit routine GR1 contains the address of a parameter list. The parameter list is three words long. The first word is the address of the record leaving the merge which would normally follow the record in the output area. This address is set to zero when all records have left the sort. The second word is the address of a record in the output area. This address is zero the first time the routine is entered as there is no record in the output area at that time. The third word is the user exit address constant. The format of the parameter list is: |

| | |
|---|---|
| | Address of Record leaving Merge |
| | Address of Record in Output Area |
| User exit address constant | |

The Sort/Merge Program sequence checks the records leaving the sort. If the E35 exit routine alters the contents of sort control fields or inserts records into the output data set that would result in the sequence check failing then the Sort/Merge Program will terminate the sorting operation. Sequence checking can be turned off with the OPTION control statement NOVERIFY parameter.

The User exit address constant can be updated by the exit. The updated value will be passed to every subsequent call to the E35 exit routine.

| Return Codes | The exit routine must return to the Sort/Merge Program with one of the following return codes in GR15 to control the disposition of the record. |

| Return Code | Disposition |
|---|---|
| 0 | No action, write the record to the SORTOUT data set. |
| | Place the address of the record leaving the merge (the first word of the parameter list) into GR1. |
| | Alter the record before writing the record to the output data set. |
| | If the exit routine needs to change the record before it is written to the SORTOUT data set then the record must be moved to a work area, modified and the address of the modified record returned in GR1. If the exit routine changes the length of the record then the RECORD control statement must identify the new record length. |
| 4 | Discard the record. |
| | The value of GR1 is not used. |
| 8 | Do not return. |
| | The Sort/Merge Program no longer calls the exit routine. Unless records are being inserted after the end of all the records sequenced by the sort then a return code of 8 must be returned to the Sort/Merge Program when a record address of zero has been passed to the exit routine. The value of GR1 is not used. |
| 12 | Insert a record. |
| | If the exit routine needs to insert a record into SORTOUT data set before the record whose address is provided in the parameter list then place the address of the record to be inserted in GR1. The Sort/Merge Program then returns to the exit with the same record address, as previously provided, in the parameter list. Additional records can be inserted at that point into the SORTOUT data set or the exit can control the disposition of the record by returning with the appropriate return code. Record insertions can be made after the Sort/Merge Program has signaled the end of all the sequenced records. The Sort/Merge Program will continue to call the exit routine until it receives a return code of 8 back from the exit routine. |

Summarizing Records

Records can be summarized in the SORTOUT data set by changing the record in the output area (second word in the parameter list) and then, if appropriate, discarding the record leaving the merge by passing back a return code of 4. The Sort/Merge Program will return to the exit routine with the address of a new record leaving the merge so the summarization process can continue. If the record leaving the merge is not discarded then that record will take the place of the record in the output area as the previous record in the output area will have been written to the SORTOUT data set.

| Linkage Conventions | Standard. |
|---|---|
| | On entry GR1 contains the address of the parameter list passed to the exit. |
| | On exit GR1 contains the address of a record if the record are being written to the output data set either unchanged or altered or inserted into the SORTOUT data set. Otherwise GR1 is not used. |
| Restrictions | If ATTACH, LINK or XCTL is used to invoke the Sort/Merge Program and the E35 exit is provided then the SORTOUT data set is ignored The E35 exit routine must discard all records passed to the exit by the Sort/Merge Program. This is achieved by continually using the Discard the Record return code to dispose of all records passed to the exit. When the Sort/Merge Program signals that there are no more records to be passed to the exit routine then the Do Not Return value must be passed to the Sort/Merge Program to signal completion of the output process. |

| **Exit/Phase Used** | **E16** | Nmax Exit – Sort phase |
|---|---|---|
| Possible use of Exit | Determine the action required when the number of input records that have been read has exceeded the calculated maximum number of records (Nmax) that can be sequenced based on the storage and intermediate work area allocations for this job. | |
| Information supplied to the exit routine | None | |
| Return Codes | The exit routine must return to the Sort/Merge Program with one of the following return codes in GR15 to control the continuing operation or termination of the sorting job. | |

| Return Code | Disposition |
|---|---|
| 0 | Sort only the records that have been read. No further records will be read. Those records that have already been read will be sorted. Message IER054I identifies the number of records that will be sorted. |
| 4 | Continue to read and sort additional records beyond the calculated NMAX value. This might or might not be successful depending on the number of additional records that have yet to be read. If sufficient resources are not available then the sorting operation will terminate. |
| 8 | Terminate the sorting operation. |

| Linkage Conventions | Standard. No parameters are passed in GR1. |
|---|---|
| Considerations | When sorting variable-length records the calculation of NMAX is highly dependent on the values provided on the RECORD control statement or the default values, if specific values are not provided. Coding more representative values, based on the characteristics of the input data set, can result in a more accurate NMAX value that will accommodate the number of records in the input data set. |
| | The Sort/Merge Program can calculate the Nmax value accurately for fixed-length records. When the Nmax value is exceeded then setting a return code of 4 to continue processing will, in all probability, result in the sort failing. |
| | If the input records are not naturally sequenced and the BALN sequencing technique is used for DASD intermediate work area storage then the Nmax value can be too low due to the increased number of sequence strings generated. An increase in intermediate work area space allocation can be required. Nmax is recalculated during the final sort phase when the BALN technique is used. The final value can be less than the original estimate. |

## Exits for Closing Data sets

These exits are called once at the end of the phase with which they are associated. They can be used to close data sets used by other exit routines in the phase or to perform any housekeeping functions that might be required.

| | | |
|---|---|---|
| **Exit/Phase Used** | **E17** | Sort phase |
| | **E27** | Intermediate merge phase |
| | **E37** | Final merge phase |
| Possible use of Exit | Use routines at these exit points to close data sets needed by other routines in the associated phases or for housekeeping functions within the phase. |
| Information supplied to the exit routine | None |
| Return Codes | None |
| Linkage Conventions | Standard, no parameters passed in GR1 |

## Exits for I/O Error Handling and DCB Modification

These exits are unique in that they are called early in each phase to establish various DCB exit points to modify the DCB at the time of OPEN or to supplement the Operating System's error recovery processing when I/O errors are detected or to enable an End of Data exit. The exit routines are not called at OPEN time or on the occurrence of an I/O error or at the End of Data. Instead, it is the routines whose address(s) were provided by the exit, and placed in the DCB by the Sort/Merge Program, that are called later at the appropriate time. When these DCB exit routines are invoked the parameters and registers passed to the routines are defined by the data management services of the Operating System and not by the Sort/Merge Program. The DCB exit routines must be located in the same load module as the exit routine that requested their installation in the DCB. DCB exit routines and coding interfaces are defined in the documents MVS Data Management Services Guide and MVS Data Management Macros.

No Sort/Merge parameter list is passed to the exit routines. Instead the exit routines pass a parameter list back to the Sort/Merge Program identifying the DCB exits to be used and the addresses of the routine to be called for the specific DCB exit. The general format of the parameter list is defined in Table 7 however each exit can only support a subset of the parameters listed in Table 7. The parameters supported for each exit are discussed in the topic specific to the exit.

| | | |
|---|---|---|
| **Exit/Phase Used** | **E18** | Sort phase |
| | **E28** | Intermediate merge phase |
| | **E38** | Final merge phase |
| | **E19** | Sort Phase |
| | **E29** | Intermediate merge phase |
| | **E39** | Final merge phase |
| Possible use of Exit | • Insert DCB exit routines into a DCB to modify the DCB at the time of OPEN. |
| | • Supplement the Operating System's error recovery processing when I/O errors are detected. |
| | • Enable an End of Data exit. |
| Information supplied to the exit routine | None |
| Return Codes | None |
| Linkage Conventions | Standard, A user-provided parameter list returned in GR1. |

The address of this parameter list is returned to the Sort/Merge Program in GR1. The list must begin on a full word boundary. It is a variable-length parameter list terminated by a word with a zero value. The high order byte of each entry identifies the particular exit parameter address that is provided in the following three bytes.

Table 7 Ex8 and Ex9 Exits Parameter List returned to the Sort/Merge Program

| Byte 1 | Byte 2 | Byte 3 | Byte |
|--------|--------|--------|------|
| X'01' | Address of DCB SYNAD Exit | | |
| X'02' | Address of DCB EXLST | | |
| X'03' | X'00' | X'00' | EROPT Code |
| X'04' | Address of End of Data exit | | |
| X'00' | X'00' | X'00' | X'00' |

| | |
|---|---|
| SYNAD | The routine is entered only after the Operating System has tried unsuccessfully to correct the I/O error. On entry the parameters passed and register contents are described in the document MVS Data Management Services Guide in the topic Synchronous Error Routine Exit (SYNAD). |
| EXLST | Contains a list of addresses of special routines that will be invoked during the processing of DCB OPEN, CLOSE and EOV service requests. The services available and the appropriate parameter list for each special routine are described in the document MVS Data Management Services Guide in the topic Exit List (EXLST). |
| EROPT | The EROPT parameter provides a facility to set the DCB EROPT parameter. This parameter specifies what action the Sort/Merge Program will take if an uncorrectable I/O error is encountered. The three possible codes and actions associated with each code are:<br>X'80'  Accept the record (block) as is<br>X'40'  Skip the record (block)<br>X'20'  Terminate the Sort/Merge Program. |
| End of Data | Contains the address of an End of Data routine that is called by the Sort/Merge Program after it encounters an end of file indication when reading the SORTIN data set. The exit routine is called after the Sort/Merge Program has closed the SORTIN data set. No parameters are passed to this exit. This exit is not the DCBEOD exit routine, it is an exit from the Sort/Merge Program. |

| | | |
|---|---|---|
| **Exit/Phase Used** | **E18** | Sort phase |
| Possible use of Exit | • | Insert DCB exit routines into the SORTIN DCB at the time of OPEN to modify the SORTIN DCB. |
| | • | Supplement the Operating System's error recovery processing when I/O errors are detected when reading records from SORTIN. |
| | • | Enable an End of Data exit when an end of file indication is encountered reading records from SORTIN. |
| Information supplied to the exit routine | None | |
| Return Codes | None | |
| Linkage Conventions | Standard, A user-provided parameter list returned to the Sort/Merge Program in GR1 as shown in Table 7. | |

---

| | |
|---|---|
| SYNAD | Supported. The contents of the registers and parameters passed to the exit are as defined for a QSAM DCB SYNAD exit. |
| EXLST | All QSAM supported EXLST entries are available for use. |
| EROPT | Supported |
| End of Data | Supported. Called when all records have been read from SORTIN and the SORTIN DCB closed. |

| | | |
|---|---|---|
| **Exit/Phase Used** | **E19** | Sort phase |
| Possible use of Exit | • Insert DCB exit routines into the SORTWKdd DCBs at the time of OPEN to modify the SORTWKdd DCBs.<br>• Supplement the Operating System's error recovery processing when I/O errors are detected when writing records to the SORTWKdd data sets. | |
| Information supplied to the exit routine | None | |
| Return Codes | None | |
| Linkage Conventions | Standard, A user-provided parameter list returned to the Sort/Merge Program in GR1 as shown in Table 7. | |
| SYNAD | Supported. The contents of the registers and parameters passed to the exit are as defined for an EXCP DCB SYNAD exit. | |
| EXLST | All EXCP supported EXLST entries are available for use. | |
| EROPT | Not supported for this exit. | |
| End of Data | Not supported for this exit. | |

| | | |
|---|---|---|
| **Exit/Phase Used** | **E28** | Intermediate merge phase |
| Possible use of Exit | Supplement the Operating System's error recovery processing when I/O errors are detected when reading records from SORTWKdd data sets. | |
| Information supplied to the exit routine | None | |
| Return Codes | None | |
| Linkage Conventions | Standard, A user-provided parameter list returned to the Sort/Merge Program in GR1 as shown in Table 7. | |
| SYNAD | Supported. The contents of the registers and parameters passed to the exit are as defined for an EXCP DCB SYNAD exit. | |
| EXLST | Not supported | |
| EROPT | Supported | |
| End of Data | Not supported | |

| | | |
|---|---|---|
| **Exit/Phase Used** | **E29** | Intermediate merge phase |
| Possible use of Exit | Supplement the Operating System's error recovery processing when I/O errors are detected when writing records to SORTWKdd data sets. | |
| Information supplied to the exit routine | None | |
| Return Codes | None | |

| | |
|---|---|
| Linkage Conventions | Standard, A user-provided parameter list returned to the Sort/Merge Program in GR1 as shown in Table 7. |
| SYNAD | Supported. The contents of the registers and parameters passed to the exit are as defined for an EXCP DCB SYNAD exit. |
| EXLST | Not supported |
| EROPT | Supported |
| End of Data | Not supported |

| | | |
|---|---|---|
| **Exit/Phase Used** | **E38** | Final merge phase |
| Possible use of Exit | | Supplement the Operating System's error recovery processing when I/O errors are detected when reading records from SORTWKdd data sets. |
| Information supplied to the exit routine | | None |
| Return Codes | | None |
| Linkage Conventions | | Standard, A user-provided parameter list returned to the Sort/Merge Program in GR1 as shown in Table 7. |
| SYNAD | | Supported. The contents of the registers and parameters passed to the exit are as defined for an EXCP DCB SYNAD exit. |
| EXLST | | Not supported |
| EROPT | | Supported |
| End of Data | | Not supported |

| | | |
|---|---|---|
| **Exit/Phase Used** | **E39** | Final merge phase |
| Possible use of Exit | | <ul><li>Insert DCB exit routines into the SORTOUT DCB at the time of OPEN to modify the SORTOUT DCB</li><li>Supplement the Operating System's error recovery processing when I/O errors are detected when writing records to the SORTOUT data set</li></ul> |
| Information supplied to the exit routine | | None |
| Return Codes | | None |
| Linkage Conventions | | Standard, A user-provided parameter list returned to the Sort/Merge Program in GR1 as shown in Table 7. |
| SYNAD | | Supported. The contents of the registers and parameters passed to the exit are as defined for a QSAM DCB SYNAD exit. |
| EXLST | | All QSAM supported EXLST entries are available for use. |
| EROPT | | Not supported |
| End of Data | | Not supported. |

## Exit for Control Field Modification

This exit can be used to lengthen, shorten or alter any control field within a record. The E option for the s parameter on the SORT or MODS control statement must be specified for control fields changed by this exit as described previously in the topics SORT Control Statement or MERGE Control Statement respectively. After the E61 exit routine has modified the control field the Sort/Merge Program sequences the control field in absolute ascending order. Although the exit modifies the control field prior to sequencing the data in the actual record remains unchanged.

---

| **Exit/Phase Used** | **E61** | The exit routine is loaded with each running phase and invoked whenever the Sort/Merge Program encounters a control field with the E specification. |
|---|---|---|
| Possible use of Exit | | Change any control field prior to sequencing but leaving the record unchanged. |
| Information supplied to the exit routine | | The Sort/Merge Program provides a parameter list with the address of the list in GR1. The list in on a full word boundary and is two words in length. The parameter list is as follows: |

| Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|
| X'00' | X'00' | X'00' | Control Field No. |
| X'00' | Control Field Address | | |

| Return Codes | None |
|---|---|
| Linkage Conventions | Standard, A parameter list is passed by the Sort/Merge Program in GR1 as described above. |

Before the E61 exit is invoked the Sort/Merge Program moves the control field to an extraction area, out of the record. The exit changes a copy of the control field, not in the record itself. Because of the internal changes made by the Sort/Merge Program the control field data appears in the extraction area in the following format:

Table 8 Control Field Format passed to E61 Exit

| **Format** | **Control Field Format.** |
|---|---|
| Binary | Unchanged. |
| Character | Unchanged. |
| Fixed Point | The sign bit is inverted. |
| Positive Floating Point | The sign bit is inverted. |
| Negative Floating Point | The sign bit is inverted and the numeric portion of the number is in one's complement. |
| Packed Decimal | The sign is considered a separate control field. It is inverted and placed before the numeric portion of the number. If the records are to be sequenced in descending order then the numeric portion is in one's complement format. For ascending sequence the number portion is unchanged. |
| Zoned Decimal | The control field is converted to packed decimal and treated as packed decimal. |

For all fields, except binary, the number of bytes the Sort/Merge Program passed to the exit routine is equal to the length specified in the m parameter of the SORT or the MERGE control statement. The control field returned to the Sort/Merge Program must be the same length.

All binary fields passed to the exit contain a whole number of bytes. If a binary field does not begin or end on a byte boundary then the Sort/Merge Program pads it with zero bits at the beginning and/or end.

The exit routine cannot physically change the length of the control field. If the length must be increased for sequencing purposes then that length must be specified in the m parameter of the SORT or the MERGE control statement. If the control must be shortened then the control field must be padded, to the length specified by the m parameter, before it is returned to the Sort/Merge Program.

The Sort/Merge Program sequences the modified control field in absolute ascending order.

# 7. Efficient Program Use

In this chapter the following subjects relating to improving the Sort/Merge Program's efficiency are discussed:

- Information provided to the Sort/Merge Program to optimize its operation.
- Intermediate storage device selection and assignment for optimum performance.
- Installation customization settings.

## Supplying Information to the Sort/Merge Program

The information provided to the Sort/Merge Program about the current sorting or merging operation aids the sort and the merge phases to run a fast and efficient sort or merge. When information such as the number of records to be sorted, and for variable-length records the average record length, are not provided then the Sort/Merge Program must make assumptions. These assumptions which, if incorrect, can result in an inefficient sorting operation.

## Data set size

The most important information to provide to the Sort/Merge Program is an accurate number of records to be sorted using the FILSZ or SIZE parameter of the SORT or the OPTION statement. If the exact number of records to be sorted is known then use that number as the value of the FILSZ parameter. If the exact number is not known, provide the best possible estimate using the estimate indication.

When the Sort/Merge Program has accurate information about the number of records to be sorted then it is able to make the most efficient use of both storage and intermediate storage.

If the dynamic allocation facility is being used then the FILSZ or SIZE parameter is also used to calculate the amount of DASD intermediate storage to be dynamically allocated. The more accurate the actual or estimated total number of records being sorted that is provided then the better the calculation of the amount of DASD intermediate storage needed for the sorting operation to complete successfully.

## Blocking Input and Output Records

Overall performance is improved if both the input and output records are blocked. Ideally, use the same blocking factor that the Sort/Merge Program uses internally which will be dependent on the characteristics of the device type selected for intermediate storage. Refer to the discussion below on assigning DASD intermediate storage for further details.

## Record Format

When the input data set consists of variable-length records, use the RECORD statement to supply maximum, minimum, and modal (most frequent) lengths to the Sort/ Merge Program. This information enables the Sort/Merge Program to calculate the optimum sort internal configuration.

The record length specifications on the RECORD statement are also used by the dynamic allocation facility. The values provided for variable-length records, particularly the modal record length (L5), will greatly impact the calculation of the amount of DASD intermediate storage needed for the sorting operation to complete successfully. In most cases the default value calculated by the Sort/Merge Program for the modal record length (L5) will not reflect the correct value and insufficient DASD intermediate storage could be allocated. Providing an appropriate value for the modal length, that better describes the characteristics of the input data, will result in a sufficient amount of DASD intermediate storage needed for the sorting operation to complete successfully.

## Intermediate Storage Assignment

Avoid assigning the minimum amount of intermediate storage for a given application. When a small amount of intermediate storage is assigned to the Sort/Merge Program, more intermediate merge phase passes are necessary because only a small number of record sequences can be merged at one time. Naturally, these extra passes increase sorting time.

Likewise, when the Sort/Merge Program has only a small amount of main storage to operate in, more intermediate merge phase passes are necessary because only a small number of records can be sorted internally and more sequences are produced.

## Assigning DASD Intermediate Storage

Sorting performance is improved if devices are used efficiently. Optimum performance is achieved if these recommendations are followed:

- From the pool of available DASD unit types, assign the DASD unit type with the largest track capacity. This will result in the Sort/Merge Program using large blocks to store the intermediate data significantly reducing the number of I/O operations needed to complete the sorting operation. As an example, by assigning 3390 DASD instead of 2314 DASD, the I/O count is generally reduced by a factor of four.
- The BALN sequencing technique requires at least three intermediate storage data sets with a maximum number of six intermediate storage data sets. For the CRCX technique, which is recommended for large sorting operations, at least seven intermediate storage data sets are required, with a maximum of 17 intermediate storage data sets. With both the BALN and CRCX sequencing techniques it is more efficient to use the minimum number of intermediate storage data sets for each technique, three for BALN and six for CRCX, as less storage is required for input/output buffers leaving more storage available for internal record storage. However, depending on the number of records being sorted, the length of the records being sorted and the capacity of a volume of the DASD unit type selected for intermediate storage it may not be possible to use the minimum number of data sets to provide the required amount of intermediate storage. In that case an increased number of intermediate storage data sets must be provided to ensure there is sufficient intermediate storage allocated to complete the sorting operation.
- Both the BALN and CRCX sequencing techniques benefit if the assigned intermediate storage data sets are all of the same size.

## Main Storage Usage

The maximum amount of main storage and the minimum amount of main storage that can be used by the Sort/Merge Program is specified at installation customization time. The default value for main storage usage is also set at installation customization time.

Performance usually improves as the Sort/Merge Program is allocated more main storage up to the maximum set at installation time. Refer to the Sort/Merge Program installation customization output to determine the values set for the three storage usage parameters and other parameters impacting storage utilization.

Using DASD unit types with a large track capacity and multiple intermediate storage data sets will require a significant allocation of main storage. Optimum performance occurs when each intermediate storage data set is double buffered. For 3390 DASD, where half-track blocking is used, each buffer requires approximately 28 KB of storage. Double buffering requires approximately 56 KB of storage for each data set. Using the CRCX sequencing technique with seven intermediate storage data sets will therefore require 392 KB of storage for optimal buffering. More storage will be required if additional intermediate storage data sets are used. Storage is also required for the internal Record Storage Area and the Sort/Merge Program load modules. The recommended storage allocation for such a sorting operation would be 512 KB.

## Changing the Main Storage Allocation

The default amount of main storage to be used, specified at installation customization, time can be overridden up to the maximum value that was set. The JCL PARM field passed to the Sort/Merge Program can be used to set the amount of storage to be used. Write the field as follows:

PARM='MAIN=*xxxxxs*' or PARM='MAIN=MAX'

Where: *xxxxxs* is the value multiplied by the optional *s* character of K or M to set the amount of main storage in bytes to be used for this operation. This value cannot be less than the minimum value set at installation customization time or greater than the maximum value set at installation customization time. Coding MAIN=MAX sets the storage available for use to the maximum value permitted.

Alternatively, and preferably, the amount of main storage to be used can be set using the MAINSIZE parameter on the OPTION control statement. Refer to the discussion of the OPTION control statement for details on how to code this parameter.

# Appendix A. Summary of How to Use the Sort/Merge Program

The following is a summary of the tasks needed in order to use the Sort/Merge Program to complete a sorting operation:

Determine the characteristics of the data to be sorted:

- Fixed or variable-length records, Undefined length records are not supported.

- Record length. If the records are variable-length records then determine the maximum, the minimum and the average length of the records to be sorted.

- Number of records to be sorted. If an accurate number is not available make an estimate that is not expected to be exceeded. An under estimate can result in a failed sort run.

- Identify the control fields in the records that will be used to sequence the records into the required order. If variable-length records are being sorted then confirm that all the control fields will be present in the minimum length records. It this is not the case then consider the use of exit routines to address the problem.

Decide if the dynamic allocation facility is going to be used to calculate the amount and then dynamically allocate the required DASD intermediate storage. If the dynamic allocation facility is not going to be used then calculate the required amount of DASD intermediate storage based on the previously determined input data characteristics.

Prepare the JCL statements required for the sort job. Chapter 3: Invoking the Sort/Merge Program with JCL Language describes the required JCL statements to run a sort job. The required DD statements will include a SORTIN DD statement and a SORTOUT DD statement unless user exits are used to provide or accept records from the Sort/Merge Program. If the dynamic allocation facility is not going to be used then SORTWKdd DD statements that allocate the required DASD intermediate storage will also have to be prepared.

Prepare the Sort/Merge control statements. Chapter 2: How to Use the Sort/Merge Program describes the syntax and format of the required control statements.

- A SORT control statement is always required to specify the control fields that will be used to sequence the records. Code the location in the record, the length of the control field, the format of the control field and desired sequence for each control field in the order of most significant control field to least significant control field.

- A RECORD control statement can be omitted if the input records are entirely sourced from a data set that in resident on DASD. A RECORD control statement is however always recommended if variable-length records are being sorted. The Sort/Merge Program calculated default values for the minimum and average record length may not accurately reflect the characteristics of the input data.

- An OPTION statement is generally not required however it is recommended that an OPTION statement be used to provide additional information to the Sort/Merge Program that can change on a run by run basis such as the FILSZ and the DYNALLOC parameters. If the data to be sorted is not entirely contained within a SORTIN data set which is resident on DASD then the FILSZ value is used by the dynamic allocation facility to calculate the amount of DASD intermediate storage required for the sort run. An estimated FILSZ value is highly recommended if the input records are going to be provided or supplemented by an E15 user exit routine. If a FILSZ value is not provided then the Sort/Merge Program will use the default value for DASD intermediate storage which may be insufficient to store and sort the number of records inserted by the E15 user exit.

Complete the preparation of the job stream and submit the job stream to the Operating System to schedule the running of the job.

The efficiency of the sorting job can sometimes be improved by providing additional information to the Sort/Merge Program. Refer to Chapter 6: Efficient Program Use for further information on improving the efficiency of the Sort/Merge Program.

The previous task list did not discuss the use of exit routines. The use of exit routines greatly enhances the flexibility of the Sort/Merge Program to satisfy a wide range of sorting requirements. Instead of the input records being solely sourced from the SORTIN data set, input records can be program generated and passed directly into the Sort/Merge Program. Instead of the records output from the sorting operation being written to the SORTOUT data set they can be accepted by a user exit routine for further processing.

User exits can also facilitate a sorting operation that would not be possible without the use of exits. An example being the sorting of variable-length records where some records are not of sufficient length to contain all of the specified control fields. The sorting job will fail when the Sort/Merge Program attempts to sequence a record that does not contain all of the control fields. An exit routine can be employed, on input to pad out the short records with user determined padding so that the short records can be sequenced. On output an exit routine can be used to remove the padding and restore the short records to their original length.

Careful reading of Chapter 5: Using Exit Routines is recommended prior to coding an exit routine. Most users will be writing exit routines in a high level language such as COBOL or PL/I that provide specific support for Sort/Merge user exits. Therefore, careful reading of the relevant language Programmers Guide is also recommended to understand the rules regarding Sort/Merge exits and the environment in which they are invoked.

# Appendix B. Sort/Merge Messages

The Sort/Merge Program generates two types of messages, those that result from critical error conditions and those that provide information regarding the operation of the Sort/Merge Program. The generated messages can be written to the SYSOUT message data set or they can be routed to the system console or to both output streams.

The message output options set at installation customization time can be overridden on a job by job basis by use of the MSG parameter in the job step PARM field or by use of the OPTION control statement or by a parameter passed to the Sort/Merge Program if it has been invoked by another program.

The Sort/Merge Program analyses control statements in two stages. Stage 1 analyses the general format of control statements. Stage 2 analyses the information contained in the control statements, in conjunction with any information that has been provided on JCL statements, to check for syntax and content errors. Each statement is scanned for errors. The first error detected stops the scan for that statement. An error message is generated and the scan continues on successive statements.

When the Sort/Merge Program encounters a critical error, a message is generated. Analysis of control information continues until the current stage is completed. The Sort/Merge Program then terminates. Thus, if a critical error is found in Stage 1 of the scan, the Sort/Merge Program terminates at the end of Stage 1; if the error is encountered in Stage 2 of the scan, the Sort/Merge Program terminates at the end of Stage 2. Depending on the installation customization settings the message number of the message identifying the critical error can also be the user abend code when the Sort/Merge Program terminates.

The messages are listed and explained in numerical order, from IER001 to IER076. The last character of each message identifier (A or I) indicates the severity of the message. A means that programmer action is required. I means that the message is an informational one and no action is required. The explanations of all critical messages begin with the word Critical.

**IER000I** 360S-SM-023 OS/360 Sort/Merge for MVS 3.8 Ver x.y – hh:mm:ss on dd mm yyyy

Explanation: Identifies the version and release of the Sort/Merge Program and the time and date of this run of the Sort/Merge Program.

System Action: None.

User Response: Informational only.

---

**IER001A** Invalid label or columns 1-15 on a continuation statement are not blank

Explanation:  Critical. Invalid label or columns 1-15 of a continuation statement are not blank.

System Action:  Termination.

User Response:  Check control statements for an invalid label, and on continuation statements for nonblank characters in columns 1 through 15.

---

**IER002A** More than 60 Control Stmts, Duplicate Stmt found or more than 1,100 chars received from ATTACHing program

Explanation:  Critical. This message is generated for one of three reasons:

- More than 60 control statements were supplied to the Sort/Merge program.
- A control statement type appears more than once. For example, there is more than one SORT statement.
- The control statements passed to the Sort/Merge program during an ATTACH, LINK, or XCTL operation contain more information than is allowed for the statements passed.

System Action:  Termination. Control statements above the 60 limit or duplicate type statements are not analyzed. If the sort was activated by an ATTACH, LINK, or XCTL, no information is processed.

User Response:  Check for excessive control statements, duplicate statement types, and, if the sort was activated by an ATTACH, LINK, or XCTL, more information than allowed.

---

**IER003A** No Continuation statement found

Explanation:  Critical. A continuation statement was indicated by a nonblank character in column 72 of the previous statement and no statement follows.

System Action:  Termination.

User Response:  Check for a coding error, an overflow of parameters into column 72, or a missing continuation statement.

**IER004A** Incorrect or invalid delimiter found

Explanation:  Critical. A control statement ends with a comma or other incorrect delimiter.

System Action:  Termination.

User Response:  Check for operands that are incorrectly split between control and continuation statements. Refer to the topic Control Statement Format in Chapter 2.

**IER005A** No control statement keyword found

Explanation:  Critical. A control statement that should contain an operation definer being SORT, MERGE, RECORD, MODS, OPTION, DEBUG or END. The control statement does not contain a valid one.

System Action:  Termination.

User Response:  Check all statements for incorrect, misplaced, or misspelled operation definers.

**IER006A** Operand does not start on the same record as the Operation keyword

Explanation:  Critical. The first operand of a control statement does not begin on the same statement as the operation definer.

System Action:  Termination.

User Response:  Check for statements that contain only the operation definer.

**IER007A** *xxxxxx* Statement syntax error

Explanation:  Critical. A control statement contains an error in syntax. *xxxxxx* is a 6 character code (SORT, MERGE, RECORD, MODS, OPTION or DEBUG) that indicates the control statement in which the error occurred.

System Action:  Termination.

User Response:  Check the control statements for syntax errors. Some of the more common syntax errors are:
Unbalanced parentheses
Missing commas or embedded blanks.

**IER008A** *xxxxxx* Parameter or value longer than 8 characters

Explanation:  Critical. A parameter greater than 8 characters has been specified. *xxxxxx* is a 6 character code SORT, MERGE, RECORD, MODS, OPTION or DEBUG that indicates the control statement in which the error occurred.

System Action:  Termination.

User Response:  Check Control Statements for parameters with more than eight characters.

**IER009I** Comment or excess information on *xxxxxx* Statement

Explanation:  More information than necessary appears in a control statement. This can be caused by a comment or a syntax error which cannot be diagnosed or is a genuine comment. *xxxxxx* is a 6 character code SORT, MERGE, RECORD, MODS, OPTION or DEBUG that indicates the control statement in which the comment or excess information occurred.

System Action:  The excess information is treated as a comment.

User Response:  Check the identified control statement to ensure it is a comment and not a syntax error.

**IER010A** No SORT or MERGE control statement found

Explanation:  Critical. All control statements have been processed and no SORT or MERGE control statement has been found.

System Action:  Termination.

User Response:  Supply a SORT or a MERGE control statement.

**IER011A** Too many Sort or Merge keywords

Explanation: Critical. More than the maximum of 5 keyword operands are defined on a SORT or a MERGE control statement.

System Action: Termination.

User Response: Check the SORT or the MERGE control statement keyword operands for too many parameters.

---

**IER012A** No FIELDS defined on SORT or MERGE control statement

Explanation: Critical. A SORT or a MERGE control statement does not contain a control field definition.

System Action: Termination.

User Response: Check the SORT or the MERGE control statement for lack of a control field definition (FIELD operand).

---

**IER013A** Invalid keyword on SORT or MERGE control statement

Explanation: Critical. An invalid keyword operand has been detected on a SORT or a MERGE control statement.

System Action: Termination.

User Response: Check the SORT or the MERGE control statement for an invalid keyword operand.

---

**IER014A** Duplicate Sort or Merge keyword

Explanation: Critical. A keyword operand is defined twice on a SORT or a MERGE control statement.

System Action: Termination.

User Response: Check the SORT or the MERGE control statement for a duplicate keyword operand.

---

**IER015A** Too many parameters on SORT or MERGE Control Statement

Explanation: Critical. Too many parameters appear on a keyword operand on a SORT or MERGE control statement.

System Action: Termination.

User Response: Check the SORT or the MERGE control statement for too many parameters

---

**IER016A** Invalid values in FIELDS operand

Explanation: Critical. An invalid number of values are specified with a FIELDS operand on a SORT or a MERGE control statement.

System Action: Termination.

User Response: Check values in control field definitions on SORT or MERGE control statement.

---

**IER017A** Error in Displacement or Length value on SORT or MERGE control statement

Explanation: Critical. An invalid length or displacement (position) value is specified in a control field definition on a SORT or a MERGE control statement.

System Action: Termination.

User Response: Check the length and position values specified in the FIELDS operand on a SORT or a MERGE control statement.

---

**IER018A** Control Field Type Error on SORT or MERGE control statement

Explanation: Critical. An error in specifying the type of control field defined on a SORT or a MERGE control statement has been detected.

System Action: Termination.

User Response: Check control field types on the SORT or the MERGE control statement for errors in specification.

---

**IER019A** Incorrect numeric value for SIZE, FILESZ, SKIPREC, STOPAFT or DYNALLOC parameter

Explanation: Critical. An error in specifying a correctly formatted numeric value for a SIZE, FILESZ, SKIPREC, STOPAFT or DYNALLOC parameter on either a SORT or MERGE control statement, has been detected.

System Action:  Termination.

User Response:  Check SORT or MERGE control statement for an invalidly formatted numeric operand.

---

**IER020A** Invalid RECORD keyword

Explanation:  Critical. An invalid keyword operand has been found in a RECORD control statement.

System Action:  Termination.

User Response:  Check the RECORD control statement for a coding error or other error in a keyword operand.

---

**IER021A** No TYPE definition found on RECORD statement

Explanation:  Critical. A RECORD control statement has been found without a TYPE operand.

System Action:  Termination.

User Response:  Check the RECORD control statement for lack of a TYPE operand.

---

**IER022A** Record TYPE not F or V on RECORD statement

Explanation:  Critical. An error in specifying the value associated with the TYPE operand of a RECORD control statement has been detected.

System Action:  Termination.

User Response:  Check the RECORD control statement for errors resulting in TYPE operand value being some character other than F (fixed-length records) or V (variable-length records).

---

**IER023A** No LENGTH operand found on RECORD Statement

Explanation:  Critical. The LENGTH operand of a RECORD control statement is not present.

System Action:  Termination.

User Response:  Check the RECORD control statement for lack of a LENGTH operand.

---

**IER024A** Incorrect value in LENGTH operand on RECORD statement

Explanation:  Critical. An incorrect value is associated with the LENGTH parameter of a RECORD control statement.

System Action:  Termination.

User Response:  Check the RECORD control statement for an incorrectly specified length value.

---

**IER025A** Record Length greater than maximum allowed

Explanation:  Critical. The record size specified on the RECORD control statement is greater than the maximum permitted.

System Action:  Termination.

User Response:  Check the RECORD control statement for an incorrectly specified record length.

---

**IER026A** L1 value not provided in LENGTH operand on RECORD statement

Explanation:  Critical. The LENGTH operand of a RECORD control statement lacks an L1 value.

System Action:  Termination.

User Response:  Check RECORD control statement for lack of an L1 value in LENGTH operand.

---

**IER027A** Control Field is defined or extends past the Minimum Record Length

Explanation:  Critical. A control field has been defined as extending beyond the minimum record length specified in the RECORD control statement.

System Action:  Termination.

User Response:  Check the SORT or MERGE control statement for incorrectly specified control field displacement. Check the RECORD control statement for incorrectly specified maximum record length value (L2).

---

**IER028A** More than 17 Exits specified

Explanation:  Critical. An attempt has been made to activate more than the maximum number of 17 program exits permitted.

System Action:  Termination.

User Response:  Reduce the number of exits specified on the MODS control statement.

**IER029A** Invalid Exit Number or invalid usage

Explanation:  Critical. This message is generated for one of two reasons.

- an exit other than the 17 permitted has been specified on a MODS control statement or
- an exit in the sort or intermediate merge phase has been activated during a merge application.

System Action:  Termination.

User Response:  Check the MODS control statement for errors resulting in specification of invalid program exit number. If a merge is being performed, check the MODS control statement for exit numbers which refer only to sort or intermediate merge phase exits.

**IER030A** Duplicate Exit definition

Explanation:  Critical. A program exit has been defined twice in the MODS control statement.

System Action:  Termination.

User Response:  Check the MODS control statement for an exit that has been defined more than once.

**IER031A** Invalid character in MODS control statement

Explanation:  Critical. An invalid character in a parameter of a MODS control statement has been found.

System Action:  Termination.

User Response:  Check the parameters of a MODS control statement for:

- a length field containing something other than numeric data or
- a source or name field beginning with something other than an alphabetic character or
- a source or length field containing a special character other than $, @, or #.

**IER032A** Exit E61 required but not specified

Explanation:  Critical. A SORT or a MERGE control statement defines a control field calling for a user written routine, by specifying E for the control field sequence indicator, and exit E61 is not activated by a MODS control statement.

System Action:  Termination.

User Response:  Check the SORT or the MERGE control statement for errors resulting in the specification of an E type parameter. Check the MODS control statement, for lack of an E61 specification.

**IER033A** No E Control Field specified with E61 active

Explanation:  Critical. Program Exit E61 is activated and no control fields have been specified for user modification with an E control field sequence parameter on the SORT or the MERGE control statement.

System Action:  Termination.

User Response:  Check the MODS, and the SORT or the MERGE control statement for errors resulting in the activation of exit E61 and the lack of an E type parameter on the SORT or MERGE control statement.

**IER034A** Parameter Error on MODS control statement

Explanation:  Critical. An incorrect number of parameters follow an operand definer on a MODS control statement, or SYSIN is specified on the MODS statement as the source for user written routines, and no SORTMODS DD statement is present.

System Action:  Termination.

User Response:  Check MODS control statement for parameter specification error.

**IER035A** Duplicate Exit module name specified

Explanation:  Critical. The same user-written routine is being used for more than one exit in a phase, or two or more routines have the same name.

System Action:  Termination.

User Response:  Check the MODS control statement for improper use of duplicate names. Duplicate names within a phase can be used only when the user written routines are to be link edited together, and they are in one load module.

**IER036I** Blocking = *xxxxxx*

Explanation: *xxxxxx* is the blocking factor used by the sort for intermediate storage records for fixed-length records. For variable-length records, the value represents size of the buffer area.

System Action:  None.

User Response:  None.

**IER037I** Records in RSA = *xxxxxx*

Explanation:  *xxxxxx* is the number of records that can fit into the program's Record Storage Area at one time during a sort.

System Action:  None.

User Response:  None.

**IER038I** Estimated maximum records = *xxxxxxxxxx*

Explanation:  *xxxxxxx* represents an estimate of the maximum number of records that can be sorted using the intermediate storage and main storage available to the sort for the current application.

System Action:  None.

User Response:  None.

**IER039A** Insufficient Storage

Explanation:  Critical. There is not enough main storage available to continue program operation.

System Action:  Termination.

User Response:  For any given sorting operation, the minimum amount required depends upon the number of intermediate storage data sets, the record length, and the block size. Reducing the number of intermediate storage data sets reduces the amount of main storage required for buffer areas. If the number of intermediate storage data sets is at the minimum permitted for the application, reducing the block size can also reduce the amount of main storage needed for buffer areas. If such corrective action is not possible, the user specified maximum must be increased using the MAIN parameter in the PARM field of the EXEC statement or using the MAINSIZE parameter on an OPTION control statement.

**IER040A** Insufficient SortWork Units

Explanation:  Critical. There is not enough intermediate storage available to the sort to continue program operation.

System Action:  Termination.

User Response:  Check the SORTWKdd DD statements for errors. Check to see if less than three intermediate storage units were assigned. Assign more intermediate storage to the SORTWKdd data sets. Check that at least three areas of at least five tracks each are specified on the DD statements.

**IER041A** SIZE parameter is greater than Estimated Maximum Records

Explanation:  Critical. The number of records specified in the SIZE operand of a SORT control statement is greater than the maximum sort capacity calculated by the program.

System Action:  The program terminates unless the data set size was estimated or not given; then the sort continues.

User Response: Check the SIZE operand of SORT control statement for error. If the SIZE operand is correct, check the DD statements for an error in assigning intermediate storage. If the DD statements are correct, assign more intermediate storage to the program and rerun.

---

**IER042A** SORTWORK Unit Error –*rrrrrr*
where *rrrrrr* provides an explanation of the error:

- Duplicate DD Name
- Unsupported Unit Type
- More than one Unit Type

Explanation:  Critical. Different unit types for SORTWORK storage devices, or an invalid combination of input, work, and output devices has been assigned or duplicate SORTWKdd DD statements have been specified.

System Action:  Termination.

User Response: Assign SORTWORK storage so that all units are of the same type, i.e., all are either the same DASD unit type or tape units. Check the DD statements for duplication.

---

**IER043A** Data set attributes not specified or conflict

Explanation:  Critical. DD statements that define the input and output data sets conflict with each other or lack any of the following information: Input or output blocking factor (BLKSIZE) or Record format (RECFM) or Record length (LRECL).

System Action:  Termination.

User Response:  Correct the statements in error and rerun the job

---

**IER044I** EXIT E*xx* invalid option

Explanation:  An invalid DCB field specification was passed to the program at exit E18, E19, E28, E29, E38, or E39. The *xx* value in the above message text is replaced by the number of the exit where the error occurred.

System Action:  The invalid option is ignored.

User Response:  Check the parameter list passed by the user-written routine against the documented acceptable parameters for the exit routine.

---

**IER045I** End of Sort Phase

Explanation:  The Sort/Merge Program's sort phase has been successfully run.

System Action:  None.

User Response:  None.

---

**IER046A** Sort capacity exceeded

Explanation:  Critical. The Sort/Merge Program has used all available intermediate storage and the input data set has not been exhausted.

System Action:  The program terminates.

User Response. Assign more DASD space for intermediate SORTWKdd data sets and rerun the job.

---

**IER047A** Record Count Error, In - *xxxxxxxxxx*, Out - *yyyyyyyyyy*

Explanation:  Critical. The number of records entering and leaving a program phase are not equal. These numbers do not include records inserted or deleted by user written routines. If an actual data set size was specified in the SIZE parameter of the SORT control statement, it is placed in the IN field of this message. This message can appear in Phase 1 or Phase 2. In Phase 3 the message is Record Count Error and message IER054I contains the count. The numbers replace the values specified as *xxxxxxxxxx* and *yyyyyyyyyy* in the text of the message as shown above.

System Action:  The program terminates.

User Response:  Check for a valid SIZE value. If correct, rerun the job.

---

**IER048I** Estimated Maximum Record (Nmax) capacity exceeded

Explanation:  The calculated sort capacity (Nmax) has been exceeded while processing the input data set, and exit E16 is specified.

System Action:  The user-written routine at exit E16 is entered. See Chapter 5: Using Exit Routines for further information.

User Response:  None. The number of records sorted is equal to the value calculated by the sort. See message IER038I.

**IER049I** Skip Merge Phase

Explanation: It is not necessary to run the intermediate merge phase to complete a sorting application because the number of sequences created by the sort phase is less than the merge order.

System Action: Control is passed directly from the sort phase to the final merge phase.

User Response: None.

**IER050I** End of Merge Phase

Explanation: The intermediate merge phase has been successfully run.

System Action: None.

User Response: None.

**IER051A** Unending Merge

Explanation: Critical. There is not enough intermediate storage assigned to successfully complete the intermediate merge phase.

System Action: Termination.

User Response: Rerun the job after assigning more intermediate storage to the Sort/Merge Program.

**IER052I** End of Sort

Explanation: The final merge phase has been successfully completed.

System Action: Return is made to the Operating System or to an invoking program for a normal end of task.

User Response: None.

**IER053A** Out of Sequence Error

**IER053A** Out Buffer *xxxxxxxxx*

**IER053A** From Merge *xxxxxxxxx*

Explanation: Critical. The current record leaving the final merge phase is not in collating sequence with the last record blocked for output. A maximum of 20 bytes are displayed in hexadecimal of the major sequence field in the record in the output buffer and the record out of the merge process

System Action: Termination.

User Response: If a user written routine was modifying the records leaving the final merge phase at the time this message was generated, check the routine thoroughly. If not, rerun the job.

**IER054I** Records In - *xxxxxxxxxx*, Out - *yyyyyyyyyy*

Explanation: The In and Out counts list the number of records accepted as input and the number of records in the output data set. If there were no records in the input data set, this field will be blank. In a merging application, the RECORDS In field is blank unless an actual data set size was specified in the SIZE parameter of the MERGE control card. When an actual size is specified, it is inserted in the Records In field of the message.

System Action: None.

User Response: None.

**IER055I** Records Inserted - *xxxxxxxxxx*, Records Deleted - *yyyyyyyyyy*

Explanation: The number of records inserted and/or deleted during the Sort/Merge Program's run replace the *xxxxxxxxxx* and *yyyyyyyyyy* values respectively.

System Action: None

User Response: None.

**IER056A** SORTIN or SORTOUT Data set not found

Explanation: Critical. SORTIN and/or SORTOUT do not appear as DD names in DD statements supplied to the Sort/Merge Program. This message can also appear when DD statements are supplied for a merge, and a SORT control statement is given instead of a MERGE statement.

System Action: Termination.

User Response: Check the DD statements for error.

**IER057A** SORTWK01 not assigned to the same Drive as SORTIN

Explanation: Critical. An intermediate storage data set other than SORTWK01 was assigned to the same tape drive as SORTIN.

System Action: Termination.

User Response: Check the DD statements for error.

**IER058A** SORTOUT assigned to the same Drive as a SORTWORK Drive

Explanation: Critical. SORTOUT was specified on the same tape drive unit as an intermediate storage data set.

System Action: Termination.

User Response: Check the DD statements for error.

**IER059A** Record Length invalid for SORTWORK device - Reason Code *x*

Explanation: Critical. The record length in the input data set(s) is either less than 18 bytes, or is too large for the assigned intermediate storage devices. A record which cannot be contained on one DASD track is too large. Reason code *x* identifies the specific incorrect value that is in error:

| Reason Code | Explanation |
|---|---|
| 1 | L1 value less than 18 |
| 2 | L2 value less than 18 |
| 3 | L3 value less than 18 |
| 4 | L4 value less than 18 |
| 5 | L5 value less than 18 |
| 6 | Record length greater than DASD track capacity |

System Action: Termination.

User Response: If the record length is too large, assign a different type of intermediate storage device with a larger track capacity. If the length is too small, redefine the sort with a record length of at least 18 bytes.

**IER060A** *dddddddd* DSCB not found on Volume *vvvvvv*

Explanation: Critical. DD statement *dddddddd* used to define a DASD intermediate storage data set on Volume *vvvvvv* is incorrect.

System Action: Termination.

User Response: Check the DD statement for an error.

**IER061A** I/O Error *xxxxx….xxxxx*

Explanation: Critical. A permanent error occurred during an I/O operation where *xxxxx….xxxxx* is the standard SYNAD message identifying the details of the error.

System Action: If no user options are specified, the program terminates.

User Response: Examine the SYNAD error message and determine the cause of the error. Change the job to avoid using the failing device.

**IER062A** Link Edit Error

Explanation: Critical. The Linkage Editor found a serious error. Running the Sort/Merge Program is not possible.

System Action: Termination.

User Response: Check user written modification routines for a problem that would cause the Linkage Editor to fail.

**IER063A** OPEN Error - *xxxxxxxx*

Explanation: Critical. An error occurred during the operation of the OPEN routine. xxxxxxxx represents the DD name of the data set being opened.

System Action: Termination.

User Response: Check for a missing or incomplete DD statement.

**IER064A** DELETE Error

Explanation: Critical. The Sort/ Merge Program was unable to delete either itself or a user written modification routine. This message should appear only when modification routines are used.

System Action: Termination.

User Response: Check any modification routines. If no modification routines are used rerun the job.

**IER065A** SYSIN Deck structure Error

Explanation: Critical. The end of file indication was received on the SYSIN data set before all needed user exit modules were read.

System Action: Termination.

User Response: Check the SYSIN data set contains all exit routine object decks that the MODS statement specifies will be present. Check for misplaced JCL statements, especially a /* preceding a user exit object deck on SYSIN.

**IER066A** Approximate Record Count - *xxxxxxxxxx*

Explanation: Critical. Sort capacity has been reached. The count *xxxxxxxxx* is an approximation of the number of records the Sort/Merge Program can successfully sequence with the assigned intermediate storage.

System Action: Termination.

User Response: Rerun application with more intermediate storage assigned to the sorting operation.

**IER067A** Invalid EXEC or ATTACH Parameter - *rrrrrrrrrrrrrrrrr*

Explanation: An error defined by *rrrrrrrrrrrrrrr* was found in the PARM field parameters of the EXEC statement, or in the optional parameters of the parameter list passed to a sort initiated by ATTACH, LINK, or XCTL. Invalid parameters are ignored. If a parameter is entered more than once, the last entry is used (if valid).

System Action: Processing continues. Invalid parameters are ignored.

User Response: None. For succeeding runs, check the validity of optional parameters.

**IER068A** Out of Sequence in SORTIN*nn*

Explanation: Critical. During a merge only run, a data set was found to be out of sequence. The nn is replaced by the data set identification (01 to 16).

System Action: Termination.

User Response: If a user written routine was modifying the records, check it thoroughly. If not, check the sequence of the data set that is in error.

**IER069A** MODULE *xxxxxxxx* Failed to LOAD

Explanation: Critical. A LOAD request for module *xxxxxxxx* returned a nonzero return code.

System Action: Termination.

User Response: Check that the Sort/Merge Program was installed correctly. If the failing module was IERAM1 check that the Installation Customization assembly and Link Edit job completed successfully.

**IER070I** Control Stmts Input from DD Name *dddddddd*

**IER070I** Control Stmts from Program Parameter List

**IER070I** Control Stmts *ccccccccccccccccccccccccccccccccccccccccccccc*

Explanation: The first message format identifies the source of the following control statements. Subsequent IER070I messages list the control statements input to the Sort/Merge Program. These message will only be generated if the List option is specified.

System Action: None.

User Response: None.

**IER071A** OPTION Statement Error – *rrrrrrrrrr* where rrrrrrrrrr provides an explanation of the error detected and *kkkkkkkk* identifies the specific field in error

No Fields found

*kkkkkkkk* Invalid Field

*kkkkkkkk* Duplicate Field

*kkkkkkkk* Too many parameters

*kkkkkkkk* Insufficient parameters

*kkkkkkkk* Invalid data

Explanation: Critical. An error was detected on the OPTION Statement provided to the Sort/Merge Program.

System Action: Termination.

User Response: Check the format and syntax of the provided OPTION Statement for the identified error.

---

**IER072A** DEBUG Statement Error – *rrrrrrrrrr*

Where *rrrrrrrrrr* provides an explanation of the error detected and *kkkkkkkk* identifies the specific field in error

No Fields found

*kkkkkkkk* Invalid Field

*kkkkkkkk* Duplicate Field

*kkkkkkkk* Too many parameters

*kkkkkkkk* Insufficient parameters

*kkkkkkkk* Invalid data

Explanation: Critical. An error was detected on the DEBUG Statement provided to the Sort/Merge Program.

System Action: Termination.

User Response: Check the format and syntax of the provided DEBUG Statement for the identified error.

---

**IER073A** Dynamic Allocation Error, Free/Alloc, *ddname*, RC=*xxxx*,S99ERROR=*xxxx*,*rrrrrrrrrrrrr* where *rrrrrrrrrrrrr* provides an explanation of the more common S99ERROR codes

- Invalid Device name
- Requested allocation failed attempting to allocate *ttttt* tracks

Explanation: Critical. An attempt to dynamically allocate a SORTWKdd data set failed.

System Action: Termination.

User Response: Determine the cause of the dynamic allocation failure and correct the error.

---

**IER074A** Storage Size requested not available

Explanation: Critical. The Sort/Merge Program was not able to GETMAIN the amount of storage requested by the MAINSIZE parameter or the installation customization default value or the minimum amount of storage required by the Sort/Merge Program was not available.

System Action: Termination.

User Response: Increase the region size specified for the Sort/Merge Program so that the minimal main storage requirement is met or reduce the amount of storage requested by the MAINSIZE parameter.

---

**IER075A** DYNALLOC Error. Less than 3 or more than 17 data sets have been specified for intermediate storage

Explanation: Critical. At least 3 DASD intermediate storage data sets must be specified and no more than 17 DASD intermediate storage data sets can be specified for dynamic allocation.

System Action: Termination.

User Response: Check the coding of the DYNALLOC parameter on the provided SORT or OPTION control statement to ensure the number of intermediate datasets specified is within the set limits.

---

**IER076I** Dynamically allocating *nn* SORTWORK data sets, *ttttt* tracks per data set

Explanation: The dynamic allocation facility will attempt to allocate *nn* SORTWORK data sets. Each SORTWORK data set will be *ttttt* tracks in size.

System Action: None.

User Response: None. If the SORTWORK data sets are not allocated successfully then message IER073A will identify the reason for the failure.

---

# Appendix C. Syntax

This document uses two kinds of description for the syntax of control statements and parameters. The descriptions are:
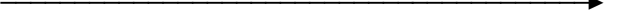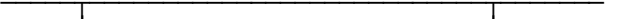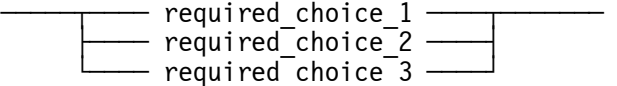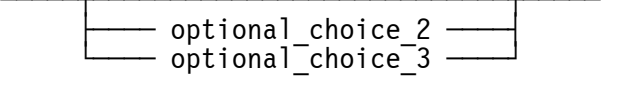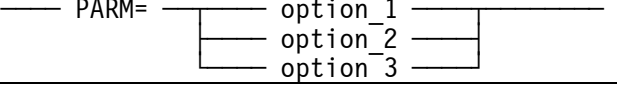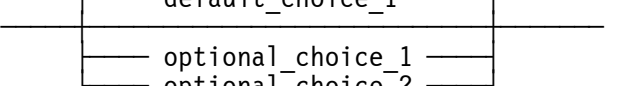
- Syntax descriptions
- Syntax diagrams

## Reading Syntax Descriptions

Table 9 Reading Syntax Descriptions

| Syntax Element | Description |
|---|---|
| KEYWORDS | Keywords are denoted with upper case letters. Obey the spelling. In the actual statements or commands they can be coded in upper case or lower case letters |
| *variables* | All user-defined values are denoted with lower case italic letters. In the actual statements or commands they can be coded in upper case or lower case letters. |
| {   } | Signifies that all, or some portion, of the code elements between the braces are required elements. Note that the braces are not part of the statements and must be not coded. |
| [   ] | Signifies that all, or some portion of the code elements between the square brackets can optionally appear but are not required elements. Note that the square brackets are not part of the statements and must be not coded. |
| \| | The OR symbol signifies that you can use only one of the code elements or values from the possible choices. Note that the OR symbol is not part of the statements and must be not coded. |
| xxx,... | Signifies that there can be more than one value in a comma delimited list. Note that the dots are not part of the statements and must be not coded. |
| xxx ... | Signifies that there can be more than one value in a blank space delimited list. Note that the dots are not part of the statements and must be not coded. |

## Reading Syntax Diagrams

| Symbol | Description |
|--------|-------------|
| ►►─────────────────────────────── | This symbol indicates the beginning of a syntax diagram. |
| ───────────────────────────►◄ | This symbol indicates the end of a syntax diagram. |
| ───────────────────────────► | This symbol indicates that the syntax diagram is continued on the next line. |
| ►──────────────────────────── | This symbol indicates that the syntax diagram is a continuation from the previous line. |
| ──────── required_element ──────── | A required element (keyword or variable) appears on the main path of the horizontal line. This element must be specified |
| ─────────────────────────── <br> └───── optional_choice ─────┘ | An optional element (keyword or variable) appears below the main path of the horizontal line. This element may or may not be specified. |
| ───┬──── required_choice_1 ────┬─── <br> ├──── required_choice_2 ────┤ <br> └──── required_choice_3 ────┘ | A required choice (keyword or variable) appears vertically stacked in the main path of the horizontal line. One of the available options must be chosen |
| ─────────────────────────── <br> ├──── optional_choice_2 ────┤ <br> └──── optional_choice_3 ────┘ | An optional choice (keyword or variable) appears vertically stacked below the main path of the horizontal line. One of the available options can be chosen. |
| ──── PARM= ───┬──── option_1 ────┬──── <br> ├──── option_2 ────┤ <br> └──── option_3 ────┘ | A keyword with options. Only one of the available options can be specified. |
| ──┬──── default_choice_1 ────┬── <br> ├──── optional_choice_1 ────┤ <br> └──── optional_choice_2 ────┘ | An optional choice (keyword or variable) with default appears vertically stacked with the default value above the main path of the horizontal line and the remaining optional elements below the main path of the horizontal line. Only one of the available options can be specified. If none of these elements is explicitly specified, the default above the main line is taken. |

| Symbol | Description |
|---|---|
|  optional_choice <br> optional_choice | An arrow returning to the left of an element below the main path of the horizontal line indicates an optional repeatable item. A character within the arrow path means that repeated items have to be separated by that character. If there is no character within the arrow path then the items are separated by a blank. |
| ──────┤ SEGMENT ├────── | This symbol is a reference to a syntax segment, which is described separately below the main syntax diagram. Complex syntax diagrams are occasionally broken into separated simpler segments. |
| ├── SEGMENT= ── value_1 <br> value_2 ──┤ | This symbol indicates a syntax segment which is referenced from a main syntax diagram that is shown above the syntax segment. |
| KEYWORDS | Keywords are denoted with upper case letters. Obey the spelling. Lower case letters are optional and can be omitted (for example DISable). In the actual statements or commands the keywords can be coded in upper case or lower case letters. |
| variables | All user-defined values are denoted with lower case italic letters. They represent user names or values. In the actual statements or commands they can be coded in upper case or lower case letters. |
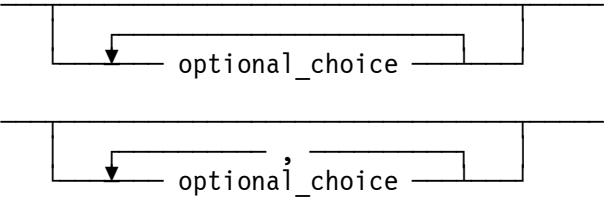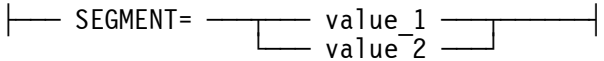
Figure 33 Reading Syntax Diagrams

**Sample Syntax Diagram**

**Syntax**

►►── CMDNAME ─── *required_argument* ──────────────────────────►

                               *optional_argument*

►─── *required_choice_1* ──────────────────────────────────────►
     *required_choice_2*        *optional_choice_1*
     *required_choice_3*        *optional_choice_2*

►─── REQUIRED KEYWORD=*variable* ──────────────────────────────►
                        •── OPTIONAL_KEYWORD=*variable*

►─── DEFAULT_KEYWORD ──── KEYWORD=*default_choice* ───────────►◄
        KEYWORD_1            KEYWORD= ──── *choice_1*
        KEYWORD_2                          *choice_2*

►─── *repeatable_item n* ─── , ─── *repeatable_item_n* ────────►

►─┤ fragment_name ├─── ( ── *variable_1* ── , ── *variable_2* ── ) ──►

**fragment_name**

├─── DEFAULT_KEYWORD ──── KEYWORD= ──── *choice_1* ───────────┤
        KEYWORD_1                       *choice_2*
        KEYWORD_2                       *choice_3*
        KEYWORD_3                       choice_4
        KEYWORD_4

Figure 34 Sample Syntax Diagram

# Index