OS/360 Sort/Merge for MVS 3.8

Installation, Customization And Diagnosis 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 installation, customization, diagnostic facilities and performance considerations for the OS/360 Sort/Merge Program for MVS 3.8.

It discusses:

The steps required to install the program and the optional material from the distribution tape.

Customizing the program to suit the installation's requirements.

Running the provided Installation Verification Programs to validate the successful installation of the Sort/Merge Program.

Building the Sort/Merge Program from the provided optional materials.

Using the built-in diagnostic facilities and DEBUG control statement to diagnose problems with the Sort/Merge Program.

Performance Considerations.

Diagnostic Sort/Merge Program messages.

Who should read this document

This document is intended for users who are responsible for the installation, customization and support of the OS/360 Sort/Merge Program for MVS 3.8 at an MVS 3.8 installation. This document does not describe the control statements and JCL Language needed to run the Sort/Merge Program to sequence data. That information is provided in the related document OS/360 Sort/Merge for MVS 3.8 Application Programming Guide.

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 responsible for the installation, customization and support of the System/360 Operating System Sort/Merge Program for MVS 3.8. It contains a step by step guide on how to install the Sort/Merge Program 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 program at the installed site. Optional source material is provided for those users who wish to assemble and link the Sort/Merge Program in their own installation. A guide is provided to assist this process. The DEBUG control statement is described together with other related diagnostic information that can assist in diagnosing problems with the Sort/Merge Program.

Chapter 1: Installation

This chapter details the pre-installation requirements and then describes the step by step process needed to install the program from the distribution tape.

Chapter 2: Customization

This chapter describes the customization options and their impact on the operation of the Sort/Merge program. The process to update or change the customization options is described.

Chapter 3: Installation Verification Programs

Five Installation Verification Program jobs are provided to confirm the successful installation and operation of the Sort/Merge Program. This chapter describes their operation and the functions verified by the IVPs. The IVP job streams can be used as examples for the preparation of job streams for user sorting operations.

Chapter 4: Building the Sort/Merge Program

This chapter describes how to build the Sort/Merge Program from the source code libraries provided as part of the optional material. The job streams provided for the assemblies and link edit tasks are described together with the expected results from each step.

Chapter 5: Diagnostic Facilities

This chapter describes the format and use of the DEBUG control statement to provide diagnostic output on the operation of the Sort/Merge Program. Use of the SORTDIAG JCL DD statement and the output written to this data set are also described in detail.

Chapter 6: Performance Considerations

This chapter describes the factors that impact the performance of the Sort/Merge Program and makes recommendations for optimizing sorting performance.

Appendix A

Contains a directory of the Sort/Merge Program’s diagnostic messages.

Appendix B

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 Application Programming Guide, describes the required control statements, JCL Language and optional programming interfaces needed to use the Sort/Merge Program to sequence or merge data. Some information in that document can be of assistance to the installer to better understand the implications of certain installation customization options upon the operation of the Sort/Merge Program.

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

The 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 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. Installation

This chapter details the pre-installation requirements and then describes the step by step process needed to install the Sort/Merge Program. The installation steps and the provided job stream make the assumption that the target environment is a MVS 3.8 system running in a TK5 configuration. If this is not the case then the installer will need to make changes to the provided job streams, both in the installation steps, and running the IVPs.

1.1 Pre-installation requirements

Before commencing the installation process gather or confirm the availability of the following resources:

- The OS/360 Sort/Merge for MVS 3.8 PC distribution file INSTALL.SORT.XMI.
- The Master Catalog password. It will be needed for a number of the installation steps for changes to the Master Catalog.
- Use of a TSO user-id with a sufficient access rights to update the SYS2.LINKLIB data set and delete and allocate the SYS1.SORTLIB data set.

1.2 Installation Steps

1. Upload the INSTALL.SORT.XMI PC File.

Using IND$FILE PC File Transfer, or any other appropriate file transfer method, upload the INSTALLSORT.XMI PC file to the target TK5 environment ensuring the uploaded MVS data set has DCB parameters of RECFM=FB,LRECL=80,BLKSIZE=27920

2. RECEIVE the Installation PDS.

Issue the following command at a TSO READY prompt:

```
RECEIVE INDS('INSTALL.SORT.XMI') NOPROMPT
DSN('INSTALL.SORT') VOL(TK5002)
```

3. Install the Sort/Merge Program
The Sort/Merge libraries can now be extracted from the installation PDS using the TSO RECEIVE command running in TSO batch mode.

Member $INSTALL in the data set INSTALLSORT contains the installation JCL. Customize this to conform to site standards and submit the job.

This job will copy the Sort/Merge Program definition load modules to the SYS2.LINKLIB data set. The data set SYS1.SORTLIB will be deleted and then reallocated. The Sort/Merge Program runtime load modules will then be copied to the newly allocated SYS1.SORTLIB data set. The cataloged procedures, SORT and SORTD, optimized for the MVS 3.8 environment, are copied to the SYS2.PROCLIB data set. The cataloged procedure ASMPROJ is also copied to the SYS2.PROCLIB data set for use, optionally, in building the Sort/Merge Program from the provided source data sets.

2. Customization

This chapter describes the customization options and their impact on the operation of the Sort/Merge Program. Each customization option is discussed and the default values provided with the Sort/Merge Program are identified. The process to update or change the customization options is described. The customization process can be rerun at any time after the Sort/Merge Program has been installed.

The customization options are provided to the Sort/Merge Program as parameters to the SORTMERG macro. The macro, with its parameters, is then assembled and link edited to produce the load module IERAM1 which is placed in the SYS2.LINKLIB data set where it is accessed by the Sort/Merge Program's definition phase load modules.

2.1 SORTMERG Macro Parameters

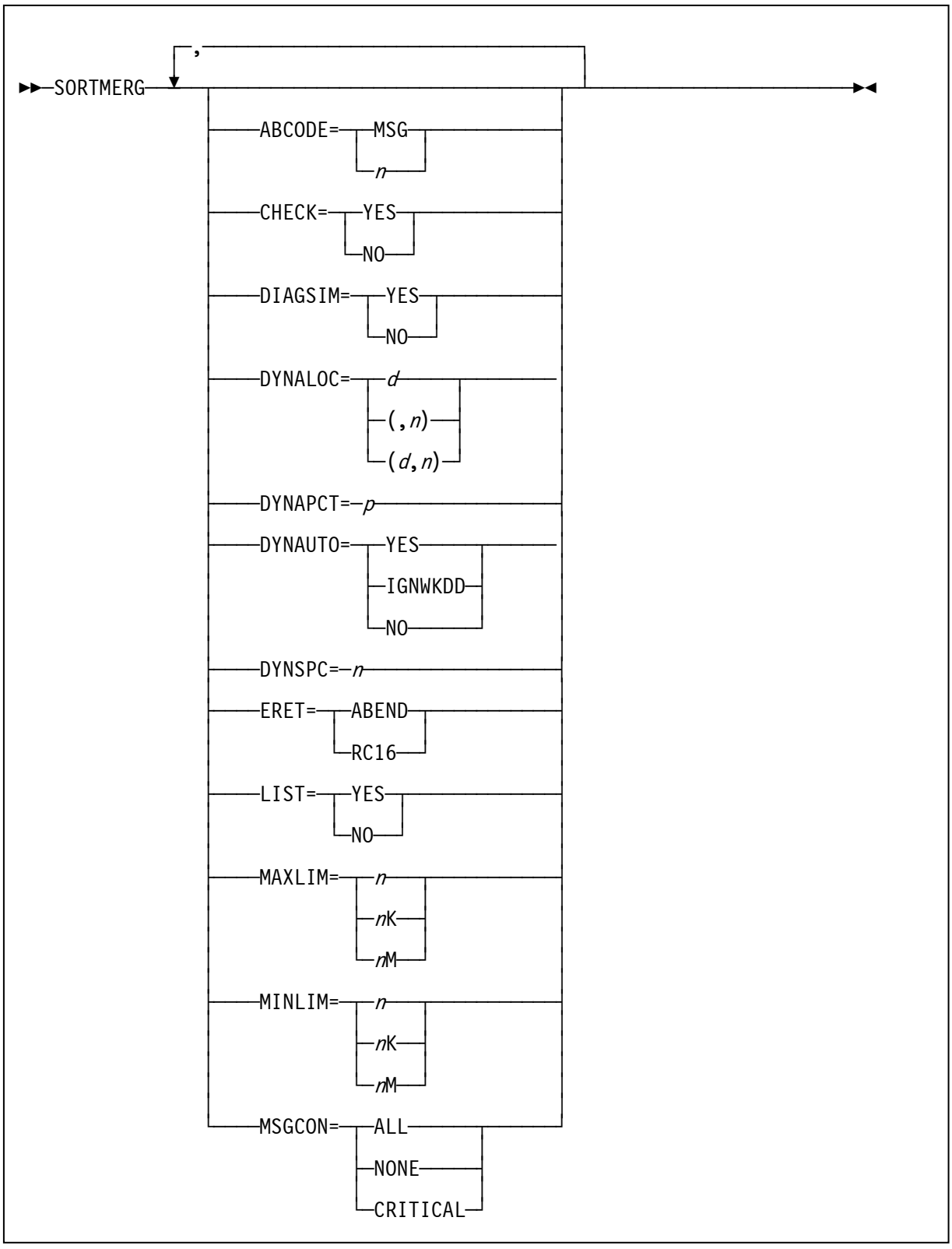


Figure 1 SORTMERGE Macro Definition

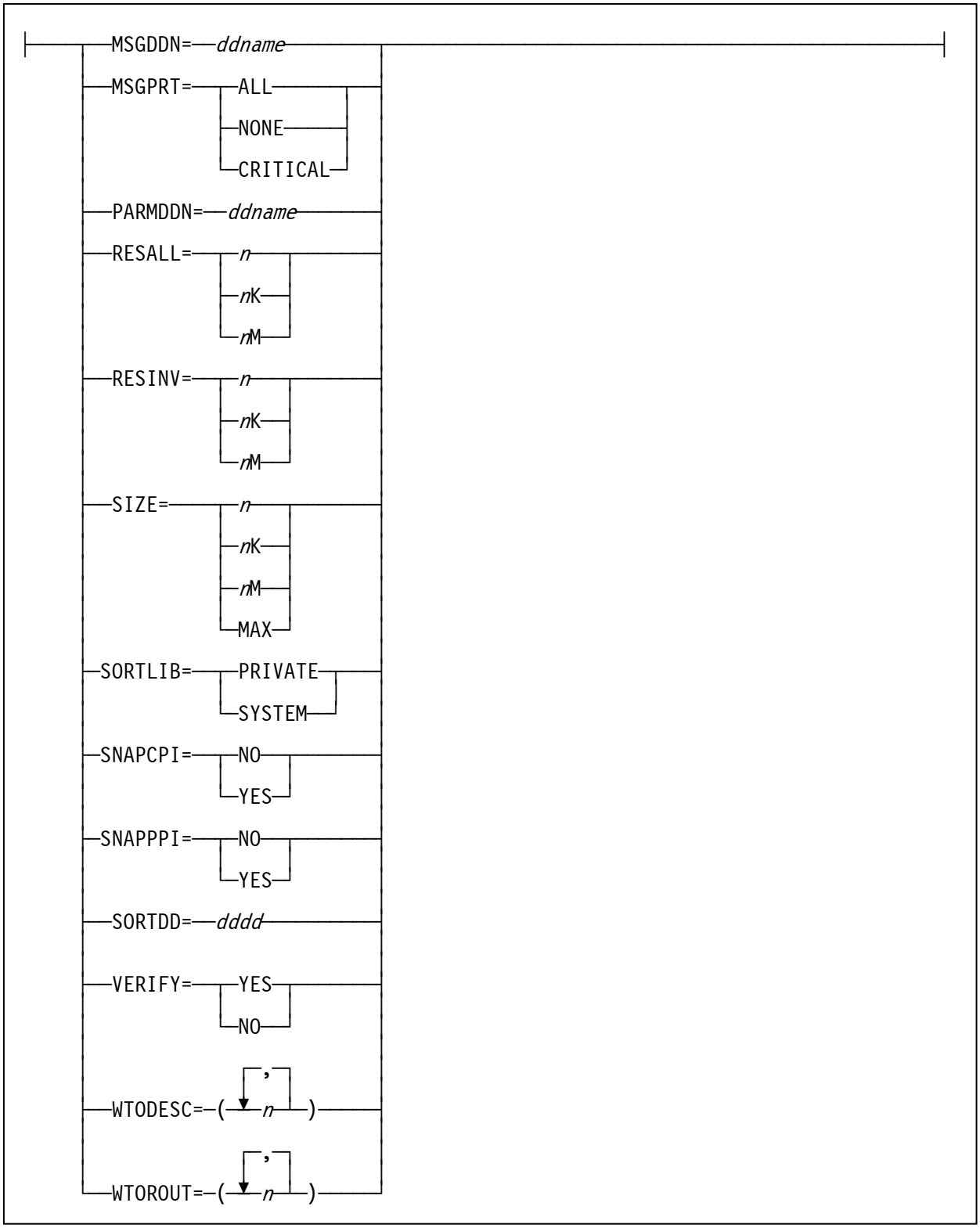
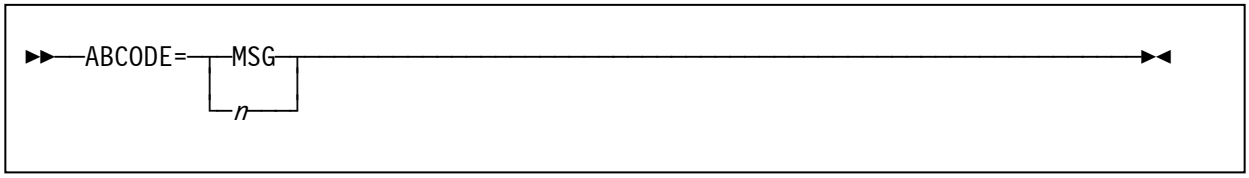


Figure 2 SORTMERG Macro Definition (Cont)

ABCODE



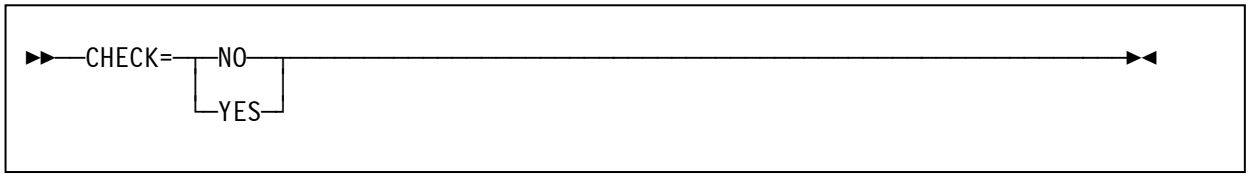
For situations where the Sort/Merge Program has detected a critical error and must terminate a user determined value can be used as the user abend code or, alternatively, the user abend code can be set to the message number of the message identifying the reason for the termination.

MSG Abend with the user abend code set to the message number identifying the reason for the abend.

n Abend with a user abend code between 1 and 99.

Default: MSG

CHECK



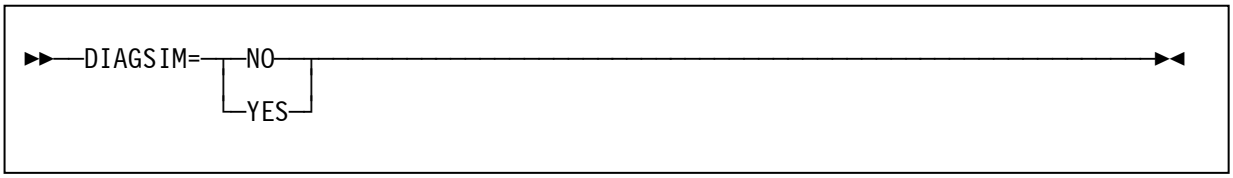
The CHECK option is used to specify that the record count check will apply for sorting operations that only use the E35 exit to process records without a SORTOUT data set. NOCHECK bypasses the record count check.

YES The record count will be checked

NO The record count will not be checked

Default: YES

DIAGSIM



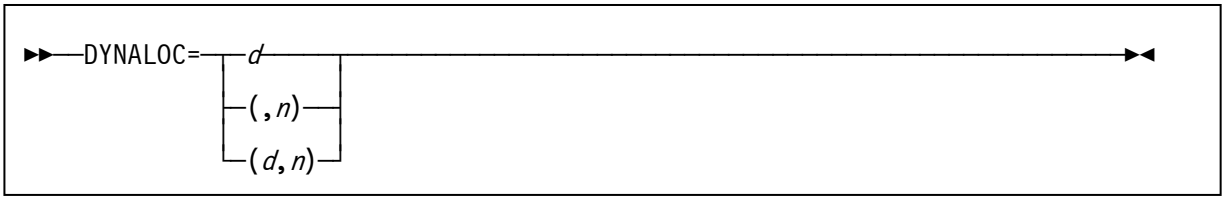
The DIAGSIM=YES parameter simulates the presence of a SORTDIAG DD statement in the sort job step JCL stream. The diagnostic mode of the Sort/Merge Program is activated. All diagnostic message output is written to the SYSOUT data set.

YES Diagnostic message output is written to the SYSOUT data set

NO Diagnostic mode is not activated therefore no diagnostics messages are written to the SYSOUT data set

Default: NO

DYNALOC



When DASD is selected for intermediate storage then using the dynamic allocation facility avoids the need for the user to calculate the amount of intermediate storage required for the sorting operation and to provide JCL DD statements 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 used by the Sort/Merge Program to dynamically 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 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.

n

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

Default: The default for *d* is the DASD unit type of 3390. The default for *n* is 6.

Note

The DASD unit type selected will impact sorting operations where the user accepts the default value for intermediate storage DASD unit type. The largest record length that can be sorted is approximately equal to the largest record that can fit on the selected DASD track 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 record size able to be sorted is reduced to half the track capacity minus the internal block overhead. Table 1 shows the approximate maximum record size that the Sort/Merge Program will accept for a given DASD unit type when either fixed or variable length records are used.

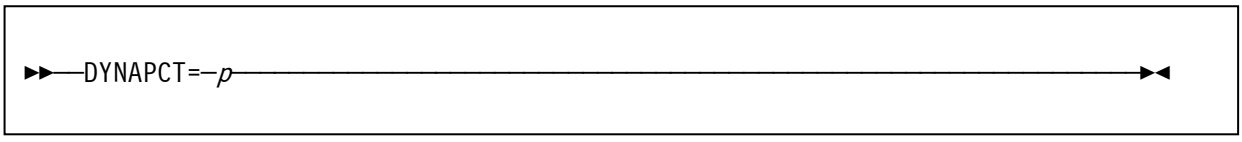
Table 1 DASD Unit Type – Maximum Record Length

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

For optimum sorting performance set the default device name to a suitable DASD unit type or user assigned group name with the largest track capacity and sufficient available space. This will result in the Sort/Merge Program using large blocks to store the intermediate data with a significant reduction in 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 will be reduced by a factor of four together with a substantial reduction in processor usage.

The number of work data sets allocated will determine the sequencing technique selected by the Sort/Merge Program. 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.

DYNAPCT

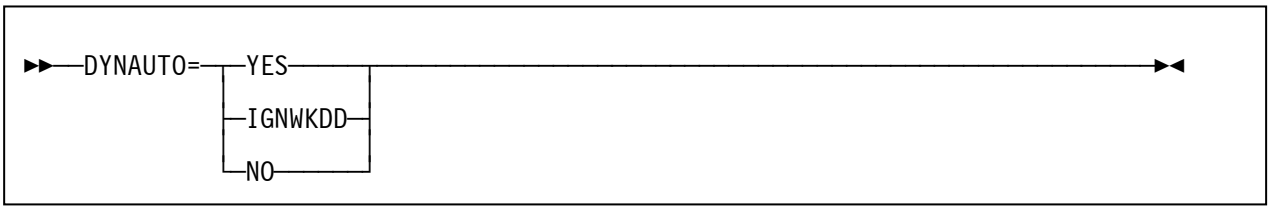


The DYNAPCT parameter sets a default for the percentage uplift, to the value the Sort/Merge Program calculated, for the amount of intermediate storage to be allocated for sorting operations. This parameter is particularly useful when variable length records are being sorted. The calculated median record length value is often 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 enables the successful completion of the sort operation.

p specifies the percentage uplift.

Default: 10

DYNAUTO



The DYNAUTO parameter controls the use of the dynamic allocation facility for intermediate DASD storage.

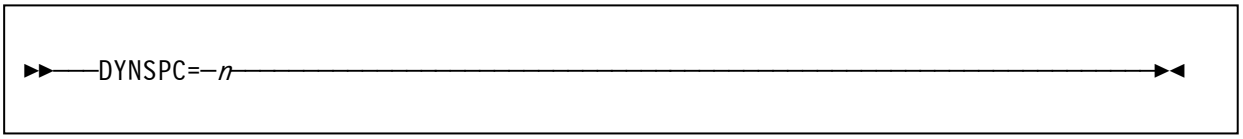
YES Specifies that the dynamic allocation facility will be used if no SORTWKdd data sets are found in the job step JCL stream.

IGNWKDD Specifies that the dynamic allocation facility will always be used. Any SORTWKdd data sets found in the job step input stream will be de-allocated and new SORTWKdd data sets will be allocated by the Sort/Merge Program using the dynamic allocation facility.

NO The dynamic allocation facility will not be used. Job steps invoking the Sort/Merge Program must provide suitable SORTWKdd JCL DD statements in the input stream.

Default: YES

DYNSPC

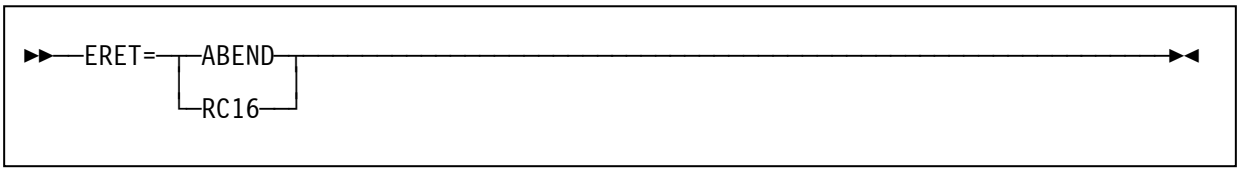


When the dynamic allocation feature is used this parameter specifies, in megabytes, the total amount of intermediate storage to be allocated for work data sets. This parameter is only used when the input file record count is not provided to the Sort/Merge Program and the input data set is not DASD resident. This situation is most likely to occur when an E15 user exit is used to provide all the input records to the Sort/Merge Program.

n specifies the total space to be allocated in megabytes.

Default: 10 megabytes

ERET



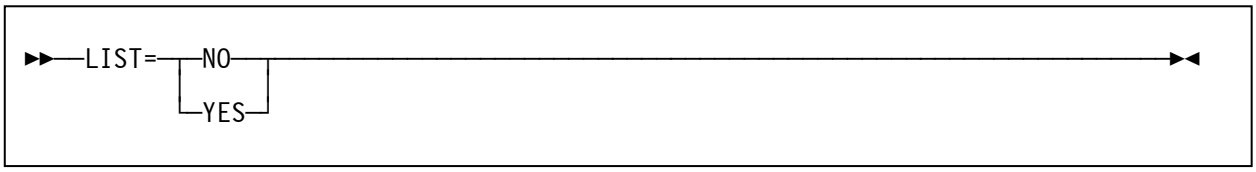
The ERET parameter determines the action the Sort/Merge Program will take when it encounters a critical error and must terminate the sorting operation.

ABEND The Sort/Merge Program will ABEND. Depending on the setting for ABCODE parameter either the user determined ABEND code value will be used or the number of the message identifying the reason for the ABEND will be used as the user ABEND code.

RC16 The Sort/Merge Program will terminate with a return code of 16.

Default: ABEND

LIST



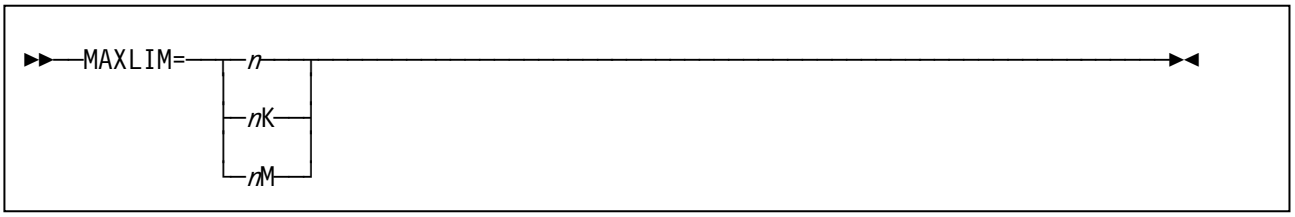
The LIST parameter controls the listing of all input control statements, including control statements passed to program invoked sorts, on the selected output message stream.

YES All control statements will be listed.

NO No control statements will be listed.

Default: YES

MAXLIM



MAXLIM specifies the maximum amount of storage, in bytes, that the Sort/Merge Program can use during a sorting operation. Any user specified storage value provided to the Sort/Merge Program by a JCL EXEC PARM parameter, OPTION statement parameter or in a ATTACH, LINK, XCTL parameter list for an invoked sort cannot exceed this value.

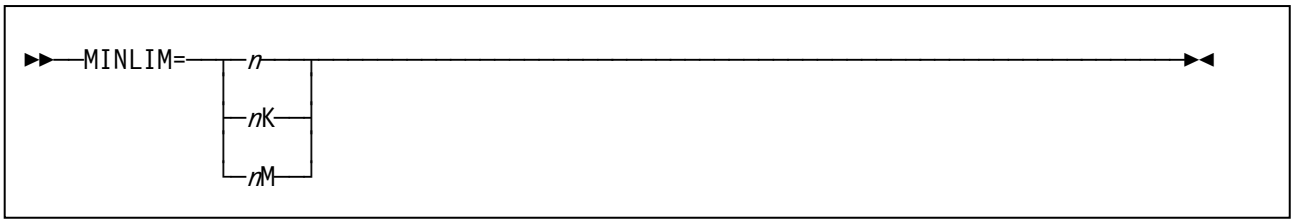
n Maximum value expressed in bytes.

*n*K Maximum value expressed in the number of K Bytes.

*n*M Maximum value expressed in the number of M Bytes.

Default: 2048K

MINLIM



MINLIM specifies the minimum storage requirement, in bytes, for the Sort/Merge Program. If the Sort/Merge Program is not able to obtain the specified minimum amount of storage then the sorting or merging operation is terminated. The storage requirements for the Sort/Merge Program have increased considerably compared to the requirements of the previous release. This is primarily due to the increased buffer sizes needed for optimal usage of large track capacity DASD unit types.

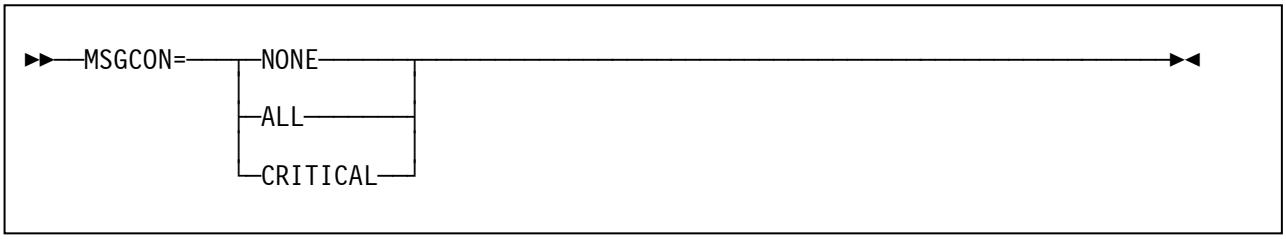
n Minimum storage value expressed in bytes

*n*K Minimum storage value expressed in the number of K bytes

*n*M Minimum storage value expressed in the number of M bytes

Default: **256K**

MSGCON



The MSGCON parameter sets the default filter for message flow to the console. The routing used for all messages to specific consoles is controlled by the WTOROUT and WTODESC parameters.

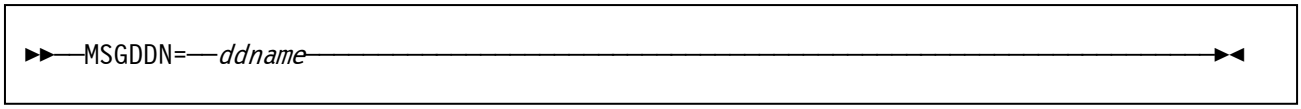
NONE No messages will be routed to the console

ALL All messages will be routed to the console

CRITICAL Only critical messages, resulting in the termination of the Sort/Merge Program, will be routed to the console

Default: NONE

MSGDDN

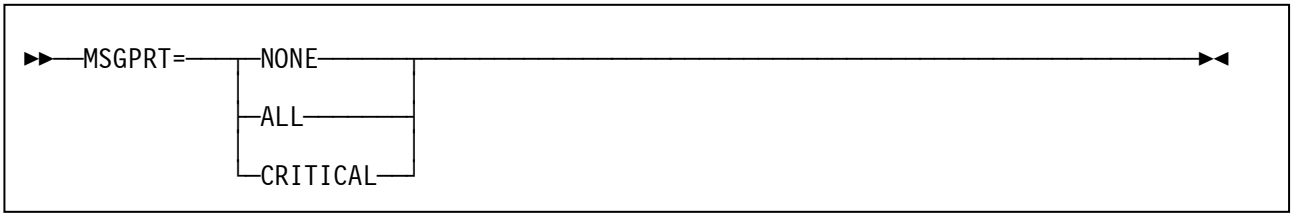


The MSGDDN parameter sets the default DD name for the Sort/Merge Program message data set. The characters must conform to the specifications for valid JCL DD names.

ddname The default DD Name for the message data set

Default: SYSOUT

MSGPRT



The MSGPRT parameter sets the default filter for message flow to the SYSOUT message data set.

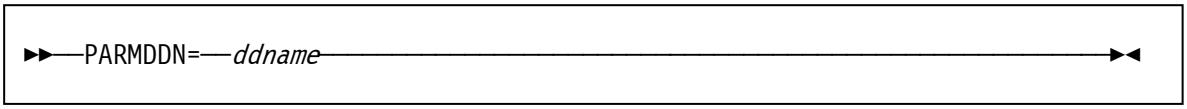
NONE No messages will be routed to the SYSOUT message data set

ALL All messages will be routed to the SYSOUT data set

CRITICAL Only critical messages, resulting in the termination of the Sort/Merge Program, will be routed to the SYSOUT data set

Default: ALL

PARMDDN

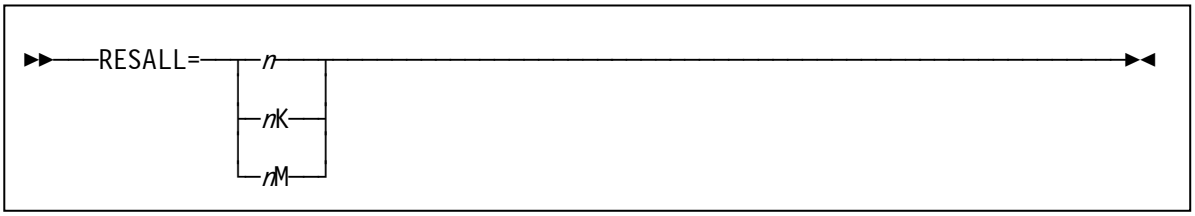


The PARMDDN parameter sets the default DD name for the IERPARM control statement input data set. If this data set is provided in the job step JCL stream then it can be used to provide control statements that override all previous sources of control statement input for a sort or a merge operation. The characters must conform to the specifications for valid JCL DD names.

ddname the default DD name for the IERPARM control statement input data set.

Default: IERPARM

RESALL



The RESALL parameter is only in effect when:

- SIZE=MAX has been specified or the user has set MAINSIZE=MAX and
- The Sort/Merge Program has been invoked by JCL statements

The value set by RESALL is subtracted from the amount of storage that the Sort/Merge Program determined was the maximum available for its use. Storage can be required for system use or for exit routines after the Sort/Merge Program's definition phase has determined the maximum amount of storage available. The RESALL parameter ensures that sufficient storage is available for later use in the job step.

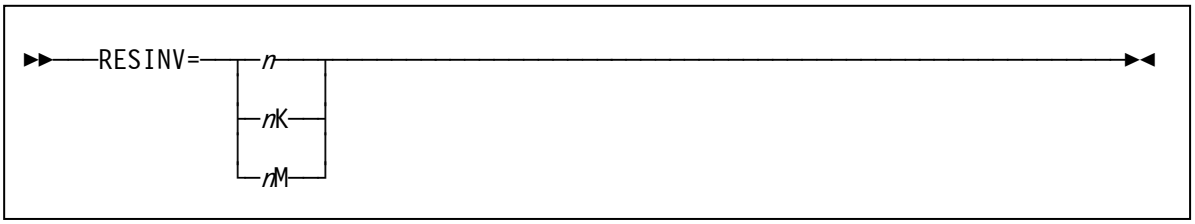
n Reserved storage value expressed in bytes

*n*K Reserved storage value expressed in the number of K bytes

*n*M Reserved storage value expressed in the number of M bytes

Default: 64K

RESINV



The RESINV parameter is only in effect when:

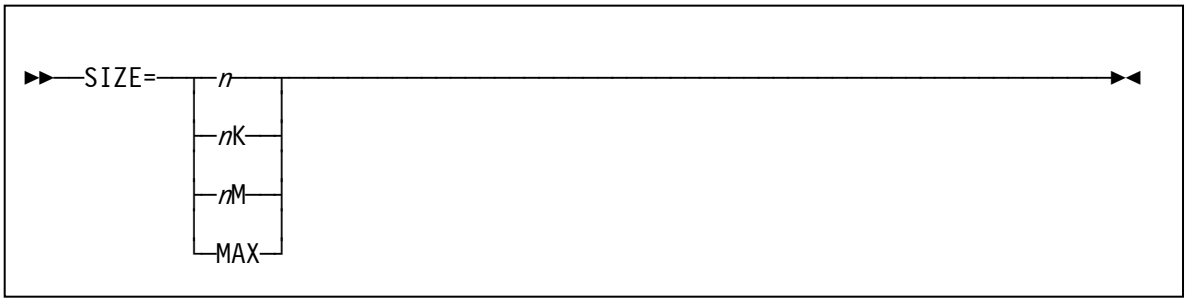
- SIZE=MAX has been specified or the user has set MAINSIZE=MAX and
- The Sort/Merge Program has been invoked by another program using the operating system's ATTACH, LINK or XCTL services

The value set by RESINV is subtracted from the amount of storage that the Sort/Merge Program determined was the maximum available for its use. Storage can be required for system use or for the invoking program after the Sort/Merge Program's definition phase has determined the maximum amount of storage available. The RESINV parameter ensures that sufficient storage is available for later use in the job step.

n	Reserved storage value expressed in bytes
*n*K	Reserved storage value expressed in the number of K bytes
*n*M	Reserved storage value expressed in the number of M bytes

Default: 96K

SIZE



The SIZE parameter sets the default limit on the amount of storage the Sort/Merge Program can use for its operation. The value set by the SIZE parameter cannot exceed the value set with the MAXLIM parameter and it cannot be less than the value set for the MINLIM parameter. If SIZE=MAX is specified then the Sort/Merge Program will attempt to use all available storage up to the limit set by the MAXLIM parameter minus the value of the RESALL parameter or the RESINV parameter depending on how the Sort/Merge Program was invoked.

n Storage value expressed in bytes

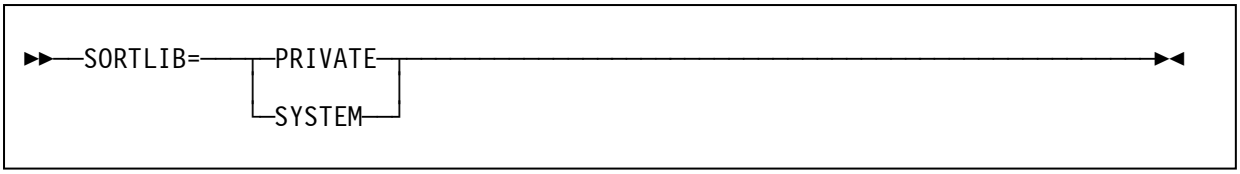
*n*K Storage value expressed in the number of K bytes

*n*M Storage value expressed in the number of M bytes

MAX Obtain the maximum storage available up to the limit set by the MAXLIM parameter

Default: 512K

SORTLIB



The SORTLIB parameter determines if the Sort/Merge Program will use the SORTLIB JCL DD statement to locate and load the assignment and run time modules during a sorting or merging operation or if it will use the services of the operating system to locate and load the required modules from either a STEPLIB JCL DD statement or from a data set placed on the LINKLST.

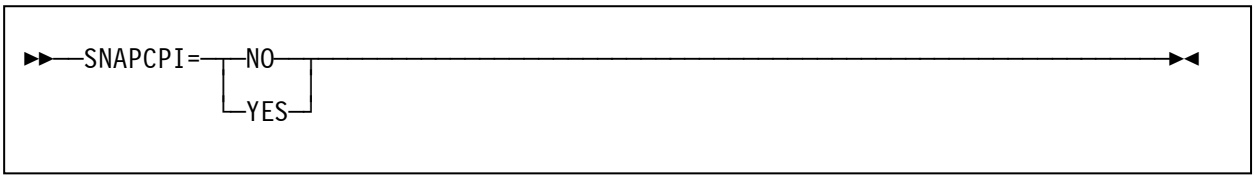
Use of the SORTLIB=SYSTEM parameter avoids the requirement for every sorting or merging operation to provide a SORTLIB DD statement in the JCL input job stream. When the SORTLIB=SYSTEM parameter is used then the load modules, usually resident in the SYS1.SORTLIB data set, can be provided by either using a STEPLIB JCL DD statement or from a data set placed on the LINKLST. The SORTLIB=SYSTEM option is not recommended as the Sort/Merge Program loads a large number of modules as it progresses through the phases of a sorting or merging operation. This would result in considerable LINKLST search activity with a possible detrimental effect on overall system performance.

PRIVATE The required load modules will be loaded from the SORTLIB DD statement

SYSTEM Either the STEPLIB DD statement or a library on the LINKLST will be used to locate and load the required load modules

Default: PRIVATE

SNAPCPI



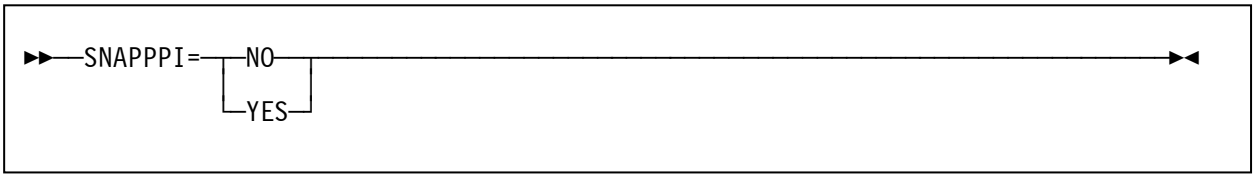
The SNAPCPI parameter is a debugging only option that is not required for general use. If the Sort/Merge Program is running in its diagnostic mode and SNAPCPI is active then, during the sort definition phase, a print dump of the CPI will be generated after control returns from each of the definition phase modules. This topic is discussed further in Chapter 5: Diagnostic Facilities.

NO No print dump of the CPI will be generated

YES A print dump of the CPI will be generated upon the exit of each definition phase module

Default: NO

SNAPPPI



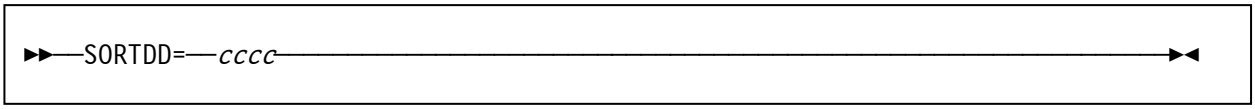
THE SNAPPPI parameter is a debugging only option that is not required for general use. If the Sort/Merge Program is running in its diagnostic mode and SNAPPPI is active then, during the sort definition phase, a print dump of the PPI will be generated after control returns from each of the definition phase modules. This topic is discussed further in Chapter 5: Diagnostic Facilities.

NO No print dump of the PPI will be generated

YES A print dump of the PPI will be generated upon the exit of each definition phase module

Default: NO

SORTDD

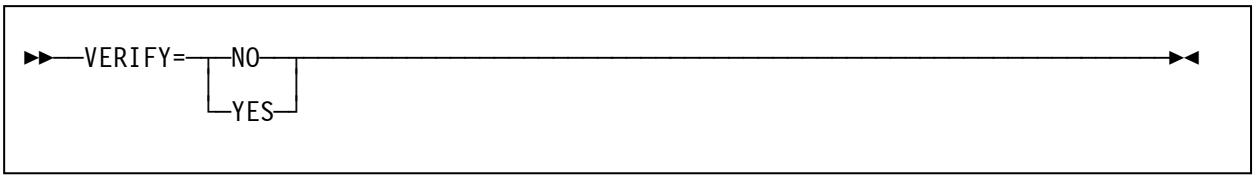


The SORTDD parameter specifies the four character prefix for ddnames used by the Sort/Merge Program. The four characters replace the first four characters in the following ddnames: SORTIN, SORTOUT, SORTINnn, SORTWKdd and SORTCNTL. This parameter does not apply to the ddname used for the Sort/Merge Program message stream. The ddname for the message data set is determined by the MSGDDN parameter. The four characters must conform to the specifications for valid JCL DD names.

cccc four character prefix for the Sort/Merge Program DD names

Default: SORT

VERIFY



The VERIFY parameter determines if the Sort/Merge Program is to perform a sequence check on the final output of records from the sorting operation to confirm the validity of the sorting operation.

NO Sequence checking will not be performed

YES Sequence checking will be performed

Default: YES

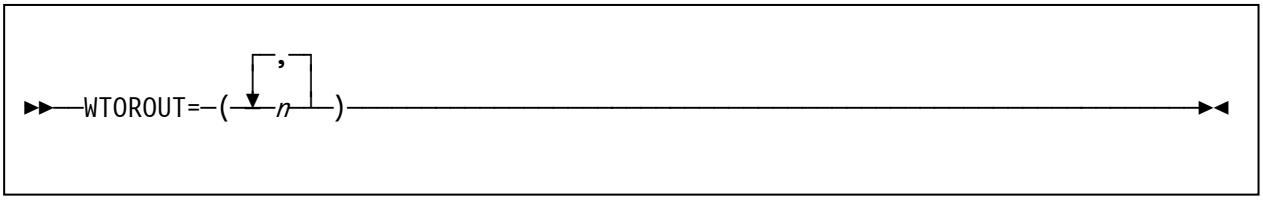
WTODESC

The WTODESC and WTOROUT parameters, together, provide the routing and descriptor codes for all WTO messages issued by the Sort/Merge Program. The parameter values and their effect on the routing of messages to specific consoles is described in the document MVS Supervisor Services and Macro Instructions. The default values route all WTO messages to the Job Log.

n values in the range 1 to 10

Default: 7, Application Program message

WTOROUT



The WTOROUT and WTODESC parameters, together, provide the routing and descriptor codes for all WTO messages issued by the Sort/Merge Program. The parameter values and their effect on the routing of messages to specific consoles is described in the document MVS Supervisor Services and Macro Instructions. The default values route all WTO messages to the Job Log.

n values in the range 1 to 15

Default: 11, Programmer information

2.2 Updating the Customization Settings

Member CUSTOMIZ in the SORT.MVS38.CNTL data set contains the job stream to change or update the customization settings.

```
REVEDIT  SORT.MVS38.CNTL(CUSTOMIZ) - 1.00                COLUMNS 00001 00072
COMMAND ===>                                                 SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 //T1CSM   JOB  111,'CUSTOMIZE S/M',    <-- CUSTOMIZE FOR INSTALLATION
000002 //             CLASS=S,MSGCLASS=C      <-- CUSTOMIZE FOR INSTALLATION
000003 //*
000004 //*********************************************************************
000005 //*
000006 //*       ASSEMBLE AND LINKEDIT THE SORT/MERGE
000007 //*       CONFIGURATION OPTIONS MODULE IERAM1
000008 //*
000009 //*********************************************************************
000010 //*
000011 //ASMOPT  EXEC PGM=IFOX00,PARM='OBJ,LINECNT=96',REGION=512K
000012 //SYSLIB   DD  DSN=SORT.MVS38.CNTL,DISP=SHR
000013 //SYSUT1   DD  UNIT=VIO,SPACE=(TRK,(30,30))
000014 //SYSUT2   DD  UNIT=VIO,SPACE=(TRK,(30,30))
000015 //SYSUT3   DD  UNIT=VIO,SPACE=(TRK,(30,30))
000016 //SYSPRINT DD  SYSOUT=*
000017 //SYSPUNCH DD  DUMMY
000018 //SYSGO    DD  DSN=&&OBJECT,UNIT=VIO,SPACE=(TRK,(30)),
000019 //             DISP=(MOD,PASS),
000020 //             DCB=(RECFM=FB,BLKSIZE=800,LRECL=80)
000021 //SYSIN    DD  *
000022          TITLE 'OS/360 SORT/MERGE FOR MVS 3.8 CUSTOMIZATION OPTIONS'
000023 *
000024 *        REFER TO OS/360 SORT/MERGE FOR MVS 3.8
000025 *        INSTALLATION, CUSTOMIZATION AND DIAGNOSIS DOCUMENT
000026 *        FOR AN EXPLANATION OF THE PARAMETERS
000027 *
000028          SORTMERG  ABCODE=MSG,                                        X
000029                CHECK=YES,                                             X
000030                DIAGSIM=NO,                                            X
000031                DYNALOC=(3390,6),                                      X
000032                DYNAPCT=10,                                            X
000033                DYNAUTO=YES,                                           X
000034                DYNSPC=10,                                             X
000035                ERET=ABEND,                                            X
000036                LIST=YES,                                              X
000037                MAXLIM=2048K,                                          X
000038                MINLIM=256K,                                           X
000039                MSGCON=NONE,                                           X
000040                MSGDDN=SYSOUT,                                         X
000041                MSGPRT=ALL,                                            X
000042                PARMDDN=IERPARM,                                       X
000043                RESALL=64K,                                            X
000044                RESINV=96K,                                            X
000045                SIZE=512K,                                             X
000046                SORTLIB=PRIVATE,                                       X
000047                SNAPCPI=NO,                                            X
000048                SNAPPPI=NO,                                            X
000049                SORTDD=SORT,                                           X
000050                VERIFY=YES,                                            X
000051                WTODESC=(7),                                           X
000052                WTOROUT=(11)
000053          END
000054 /*
```

Figure 3 Sort/Merge Program Customization

The SORTMERG macro parameters can be changed to reflect the required changes to the customization options.

Update the JOB statement to conform to the installation standards and submit the job. The job will assemble the SORTMERGE macro and link edit the IERAM1 load module into the SYS2.LINKLIB data set. Note that the TSO user-id used to submit the job will require the access rights to update the SYS2.LINKLIB data set.

If no changes are required to any of the customization options then this job can be omitted.

3. Installation Verification Programs

Five installation verification jobs are provided to verify the successful installation of the Sort/Merge Program. Members IVP1, IVP2, IVP3, IVP4 and IVP5 in the SORT.MVS38.CNTL data set contain the five IVP job streams.

3.1 IVP1

IVP1 demonstrates use of the E15 and the E35 user exits with a program that invokes the Sort/Merge Program using a LINK request to the operating system. The invoking program has been developed so that it can be used to test different record counts, record lengths, record formats, DASD unit types and sequencing techniques.

```
REVEDIT  SORT.MVS38.CNTL(IVP1) - 1.34                    COLUMNS 00001 00072
COMMAND ===>                                                 SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 //T1IVP1  JOB  111,'S/M IVP1',        <-- CUSTOMIZE FOR INSTALLATION
000002 //             CLASS=S,MSGCLASS=C      <-- CUSTOMIZE FOR INSTALLATION
000003 //*
000004 //*********************************************************************
000005 //*
000006 //*       OS/360 SORT/MERGE FOR MVS 3.8
000007 //*
000008 //*       ASSEMBLE, LINKEDIT AND RUN IVP1
000009 //*
000010 //*********************************************************************
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 49 LINE(S) EXCLUDED
000060 *********************************************************************
000061 *
000062 *        CONFIGURATIONAL SETTINGS
000063 *
000064 *        CHANGE THESE SETTINGS TO TEST DIFFERENT SORT
000065 *        CONFIGURATIONS
000066 *
000067 *********************************************************************
000068 *
000069 &NUMRECS SETA  50000             <--   NUMBER OF RECORDS TO BE
000070 *                                      SEQUENCED
000071 *
000072 &RECFM   SETC  'F'               <--   RECORD FORMAT F | V
000073 *
000074 &LRECL   SETA  250               <--   REQUIRED FOR F AND V RECORDS
000075 *
000076 *                                      TO PROVIDE A RANGE OF VARIABLE
000077 *                                      RECORD LENGTHS SET THESE
000078 *                                      PARAMETERS BELOW.
000079 &LRECLA  SETA  400               <--   REQUIRED FOR V RECORDS
000080 *
000081 &LRECLB  SETA  300               <--   REQUIRED FOR V RECORDS
000082 *
000083 &LRECLC  SETA  200               <--   REQUIRED FOR V RECORDS
000084 *
000085 &LRECLD  SETA  100               <--   REQUIRED FOR V RECORDS
000086 *
000087 &DASD    SETC  '3390'            <--   DASD TYPE FOR DYNALLOC
000088 *                                      ALL MVS 3.8 DASD TYPES SUPPORTED
000089 *                                      INCLUDING VIO
000090 *
000091 &NUMDASD SETA  6                 <--   NUMBER OF SORTWKdd DATA SETS
000092 *                                      3- 6 DATA SETS - BALN ALGORITHM
000093 *                                      7-17 DATA SETS - CRCX ALGORITHM
```

Figure 4 IVP1 Configuration Settings

Line	Explanation
000069	50,000 records will be generated and passed to the Sort/Merge Program by the E15 user exit. This value can be increased or decreased to verify the operation of the Sort/Merge Program with different numbers of input records.
000072	The program will generate fixed length records. The &RECFM variable can be changed to V to generate variable length records.
000074	The record length is set to 250. This can be changed to the minimum record length of 18 up to the maximum record length that is supported by the DASD unit type selected for intermediate storage.
000079-000086	If variable length records have been selected by setting the &RECFM variable to V then a range of record lengths can be generated. Set variables &LRECLA through to &LRECLD to generate different record lengths. The program will cycle through the four values to generate different length variable records.
000087	Set the variable &DASD to the selected DASD unit type for intermediate storage. All DASD unit types supported by MVS 3.8 can be used including VIO. The SORTWKdd data sets will be dynamically allocated by the Sort/Merge Program to the specified DASD unit type.
000091	The variable &NUMDASD controls the number of SORTWKdd data sets that will be dynamically allocated by the Sort/Merge Program. Depending on the number of data sets specified then the Sort/Merge Program will select either the BALN or CRCX sequencing technique.

Update the JOB statement to conform to the installation standards and submit the job.

A checksum process is implemented in the invoking program user exits to verify that records sequenced by the Sort/Merge Program have not been corrupted by the sorting operation. For each record generated in the E15 user exit a checksum is generated and placed in the record before it is passed to the sort. When each sorted record is received by the E35 user exit the checksum is regenerated and compared to the checksum placed in the record by the E15 exit. The invoking program is terminated if the two checksums do not match.

Check that the three job steps all ended with a condition code of zero. The Sort/Merge Program will write the following output to the SYSOUT message data set.

```
 SYS16346.T145314.RA000.T1.JOB06781 ------------------------------------------------------------------------------- Line 758 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER000I 360S-SM-023 OS/360 Sort/Merge for MVS 3.8 Version 1.01 - 14:53:07 on 11 Dec 2016
IER070I Control Stmts from Program Parameter List
IER070I Control Stmts SORT FIELDS=(5,10,CH,D),SIZE=E50000,DYNALLOC=(3390,6)
IER070I Control Stmts RECORD LENGTH=250,TYPE=F
IER036I Blocking =   111
IER037I Records in RSA =  1698
IER038I Estimated maximum records =   85026
IER050I End of Merge Phase
IER055I Records Inserted   50000, Records Deleted   50000
IER054I Records In        , Records Out
IER052I End of Sort
```

Figure 5 Example IVP1 message output

3.2 IVP2

IVP2 is the IVP program provided as part of the SAMPLIB examples and programs distributed with OS/360 Release 21. It is a simple sorting operation sequencing records of length 80 bytes with a single control field.

Update the JOB statement to conform to the installation standards and submit the job.

Check that the job step has ended with a condition code of zero. The Sort/Merge Program will write the following output to the SYSOUT message data set.

```
 SYS16346.T150230.RA000.T1.JOB06773 ------------------------------------------------------------------------------- Line 85 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER000I 360S-SM-023 OS/360 Sort/Merge for MVS 3.8 Version 1.01 - 18:36:04 on 07 Dec 2016
IER070I Control Stmts Input from DDName SYSIN
IER070I Control Stmts  SORT FIELDS=(40,10,CH,A)
IER070I Control Stmts  RECORD TYPE=F,LENGTH=(80)
IER070I Control Stmts  END
IER036I Blocking =    90
IER037I Records in RSA =  5569
IER038I Estimated maximum records =   44820
IER045I End of Sort Phase
IER049I Skip Merge Phase
IER054I Records In     500, Records Out     500
IER052I End of Sort
```

Figure 6 Example IVP2 message output

The correctly sequenced records will be written to the SORTOUT data set for printing.

3.3 IVP3

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 and its supporting utility program IFASMFDP. All sort control fields, used to sequence records, must be present in every variable-length record processed by the Sort/Merge Program. An attempt to sort short records without all the control fields present in every record will result in an unsuccessful sorting operation. This problem can be addressed by implementing user exits to filter records too short for sorting and restoring the short records to the output data set after the selected records have been sequenced by the Sort/Merge Program.

IVP3 demonstrates the use of user exits to remove SMF records that are of insufficient length to be sorted and then restore the records that were not sorted back into the output data set. The first two steps of the IVP3 job assemble and link edit the E15 and E35 exits. Input SMF records for sorting are read from the SMF daily dump data set and then sorted. Sample records from the generated output data sets are then listed using the IDCAMS utility program.

Refer to the document OS/VS2 MVS SPL: System Management Facility for a more complete description regarding sorting SMF records.

Update the JOB statement to conform to the installation standards. In addition, confirm the data set name of the SMF daily dump data set at the installation is correct before submitting the job.

.

```
 SYS16346.T150857.RA000.T1.JOB06774 ------------------------------------------------------------------------------ Line 1051 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER000I 360S-SM-023 OS/360 Sort/Merge for MVS 3.8 Version 1.01 - 18:36:36 on 07 Dec 2016
IER070I Control Stmts Input from DDName SYSIN
IER070I Control Stmts     SORT  FIELDS=(19,16,A,11,4,A,7,4,A),FORMAT=BI,SIZE=E4000
IER070I Control Stmts     MODS  E15=(E15,60000,EXITLIB,N),E35=(E35,70000,EXITLIB,N)
IER070I Control Stmts     OPTION NOVERIFY
IER070I Control Stmts     END
IER036I Blocking = 27968
IER037I Records in RSA =   464
IER038I Estimated maximum records =  113346
IER050I End of Merge Phase
IER055I Records Inserted   10535, Records Deleted   10535
IER054I Records In   16935, Records Out   16935
IER052I End of Sort
```

Figure 7 Example IVP3 message output

Check that all eight job steps ended with a condition code of zero. The Sort/Merge Program will write output similar to that listed in Figure 7 to the SYSOUT message data set. The record numbers in message IER055I and message IER054I will vary according to the number of SMF records present in the SMF.DAILY.DATA generation zero data set.

Additional listings generated by the IDCAMS utility program follow the SYSOUT message data set output.

3.4 IVP4

The IVP4 job uses the IEBDG utility program to generate 36,000 records in ascending order containing multiple control fields. The first run of the Sort/Merge Program is used to sequence the generated records into descending order. The second run of the Sort/Merge Program re-sequences the records back into ascending order. The IEBCOMPR utility program is then used to compare the original data set generated by the IEBDG utility program with the data set output from the second run of the Sort/Merge Program.

Update the JOB statement to conform to installation standards and submit the job.

```
 SYS16346.T152055.RA000.T1.JOB06775 ------------------------------------------------------------------------------ Line 224 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER000I 360S-SM-023 OS/360 Sort/Merge for MVS 3.8 Version 1.01 - 18:37:38 on 07 Dec 2016
IER070I Control Stmts Input from DDName SYSIN
IER070I Control Stmts     SORT  FIELDS=(10,10,CH,D,20,20,CH,D),SIZE=36000
IER070I Control Stmts *   RECORD TYPE=F,LENGTH=400     RECORD STATEMENT NOT REQUIRED
IER070I Control Stmts     OPTION DYNALLOC=(3350,3)
IER036I Blocking =    47
IER037I Records in RSA =  1111
IER038I Estimated maximum records =   53204
IER045I End of Sort Phase
IER050I End of Merge Phase
IER054I Records In   36000, Records Out   36000
IER052I End of Sort
IER000I 360S-SM-023 OS/360 Sort/Merge for MVS 3.8 Version V1.0 - 18:37:41 on 07 Dec 2016
IER070I Control Stmts Input from DDName SYSIN
IER070I Control Stmts     SORT  FIELDS=(10,10,CH,A,20,20,CH,A),SIZE=36000
IER070I Control Stmts *   RECORD TYPE=F,LENGTH=400     RECORD STATEMENT NOT REQUIRED
IER070I Control Stmts     OPTION DYNALLOC=(3390,7)
IER036I Blocking =    69
IER037I Records in RSA =   999
IER038I Estimated maximum records =   61686
IER050I End of Merge Phase
IER054I Records In   36000, Records Out   36000
IER052I End of Sort
                                        COMPARE UTILITY                                    PAGE 0001
    END OF JOB-TOTAL NUMBER OF RECORDS COMPARED = 00036000
```

Figure 8 Example IVP4 message output

Check that all four job steps ended with a condition code of zero. The Sort/Merge Program will write the output listed in Figure 8 to the SYSOUT message data set. The record numbers listed in the IER054I messages should be the same for both sorting job steps and also match the number of records that were generated by the IEBDG utility program and successfully compared by the IEBCOMPR utility.

The IVP4 job can also be used as a bench marking tool. It can be used to generate a significant I/O workload by setting the relevant parameters to appropriate values.

3.5 IVP5

IVP5 has been provided specifically for those users who wish to carry out performance testing or bench marking operations using the Sort/Merge Program. The program used for IVP5 is a modified version of the program used for IVP1. The same parameters provided for IVP1 to test different input record counts, record length, record format, DASD unit types and sequencing techniques are available for use in IVP5. Refer to the description of IVP1 for information on changing parameters in order to performance test or bench mark different sorting configurations using IVP5.

Processing overhead within the two exit routines, E15 and E35, has been kept to a minimum by no longer using the checksum mechanism implemented in IVP1 to confirm the integrity of the records being sorted.

The sort control field for IVP5 is generated by the use of a pseudo random number generator. Each sorting run will therefore be sorting the same sequence of random numbers. This will enable comparisons between other sorting runs where configuration parameters have been changed to observe their impact on performance and resource usage. The seed value for the random number generator can be changed by use of the &SEED parameter to generate different series of random numbers for sorting. Alternatively, a unique series of random numbers can be generated for each sorting run. Further information on the random number generation process is provided in the comments contained within the IVP5 source code.

The output from a successful run of IVP5 is similar to the output produced by IVP1.

```
 SYS17068.T131924.RA000.T1.JOB08026 ------------------------------------------------------------------------------- Line 1162 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER000I 360S-SM-023 OS/360 Sort/Merge for MVS 3.8 Version 1.01 - 13:17:51 on 09 Mar 2017
IER070I Control Stmts from Program Parameter List
IER070I Control Stmts SORT FIELDS=(5,10,CH,D),FILSZ=250000,DYNALLOC=(3390,6)
IER070I Control Stmts RECORD LENGTH=2000,TYPE=F
IER076I Dynamically allocating   7 SORTWORK data sets,  1512 tracks per data set
IER036I Blocking =    13
IER037I Records in RSA =   221
IER038I Estimated maximum records =    252694
IER050I End of Merge Phase
IER055I Records Inserted  250000, Records Deleted  250000
IER054I Records In        , Records Out
IER052I End of Sort
```

Figure 9 Example IVP5 message output

Additional timing messages will also be written to the IVP5 Job Log. These messages are generated by the IVP5 program at various key points in the sorting operation. The timing data is obtained from the MVS 3.8 operating system timing services. The times produced are guidelines only due to the PC, the PC operating system and Hercules timer implementations. The results of each run will show some variance depending on other activities occurring on the PC and within the PC operating system at the time of the IVP5 run.

```
 SYS17068.T142343.RA000.T1.JOB08027 ------------------------------------------------------------------------------- Line 8 Col 2 133
 Command ===>                                                 Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
14.22.07 JOB 8027  +IVP5 Linking to the Sort       Elapsed time =     0.00 secs, CPU time =     0.00 secs
14.22.07 JOB 8027  +IVP5 Begin record insertion    Elapsed time =     0.26 secs, CPU time =     0.20 secs
14.23.20 JOB 8027  +IVP5 End record insertion      Elapsed time =    72.24 secs, CPU time =    51.67 secs
14.23.25 JOB 8027  +IVP5 Begin receiving records   Elapsed time =     5.18 secs, CPU time =     3.24 secs
14.23.33 JOB 8027  +IVP5 End of receiving records  Elapsed time =     7.90 secs, CPU time =     6.59 secs
14.23.33 JOB 8027  +IVP5 End of Sort               Elapsed time =     0.02 secs, CPU time =     0.01 secs
14.23.33 JOB 8027  +IVP5 Totals                    Elapsed time =    85.60 secs, CPU time =    61.74 secs
```

Figure 10 Example IVP5 timing messages

Table 2 Explanation of IVP5 timing messages

Message	Explanation
Linking to the Sort	This message is issued just prior to the IVP5 program issuing a LINK request to the operating system to invoke the Sort/Merge Program.
Begin record insertion	This message is issued when the E15 exit routine is invoked for the first time. The times shown reflect the overhead of the Sort/Merge Program running its definition phase, optionally dynamically allocating the SORTWKdd data sets, opening data sets and establishing the sort phase.
End record insertion	This message is issued prior to the last return from the E15 exit routine. The times shown represent the time taken for the sort phase minus the short final completion of the sort phase.
Begin receiving records	This message is issued when the E35 exit routine is invoked for the first time. The times shown reflect the time taken for the intermediate merge phase of the sort and the establishment of the final merge phase.
End receiving records	This message is issued prior to the last return from the E35 exit routine. The times reflect the time taken for the final merge phase.
End of Sort	This message is issued when the Sort/Merge Program terminates its processing and returns to the invoking program. The time taken for this is generally minimal as the SORTWKdd data sets are closed and storage freed prior to the return.
Totals	The totals of all the Sort/Merge Program times.

4. Building the Sort/Merge Program

If the optional material was installed as part of the installation process then the Sort/Merge Program can be built from the installed source libraries. Member ASMALL in the SORT.MVS38.CNTL data set contains the jobs required to assemble all the Sort/Merge Program source modules into an object library.

Ensure that all the JOB statements in the job stream conform to the installation standards and submit the job. To ensure the assembly listing of the modules are in alphabetically ascending order either a single initiator should be used for all assemblies or the jobs assigned a Job Class only serviced by one initiator.

```
REVEDIT  SORT.MVS38.CNTL(ASMALL) - 1.00                  COLUMNS 00001 00072
COMMAND ===>                                                 SCROLL ===> CS
  64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 //T1AS    JOB  111,'ASM SORT/MERGE',   <-- CUSTOMIZE FOR INSTALLATION
000002 //             CLASS=S,MSGCLASS=Z      <-- CUSTOMIZE FOR INSTALLATION
000003 //*
000004 //*************************************************************
000005 //*
000006 //*        SUBMIT JOBSTREAM TO ASSEMBLE
000007 //*        360S-SM-023 OS/360 SORT/MERGE FOR MVS 3.8
000008 //*
000009 //*************************************************************
000010 //*
000011 //       EXEC PGM=IDCAMS
000012 //SYSPRINT DD SYSOUT=*
000013 //SYSIN    DD  *
000014    DELETE SORT.MVS38.OBJ
000015    SET LASTCC = 0
000016 /*
000017 //ALLOC   EXEC PGM=IEFBR14
000018 //OBJLIB   DD  DSN=SORT.MVS38.OBJ,
000019 //             UNIT=3390,VOL=SER=TK5002,DISP=(,CATLG,DELETE),
000020 //             DCB=(DSORG=PO,BLKSIZE=3120,LRECL=80,RECFM=FB),
000021 //             SPACE=(TRK,(60,30,36))
000022 //*
000023 //T1A01   JOB  111,'ASM SORT/MERGE',   <-- CUSTOMIZE FOR INSTALLATION
000024 //             CLASS=S,MSGCLASS=C      <-- CUSTOMIZE FOR INSTALLATION
000025 //*
000026 //*************************************************************
000027 //*
000028 //*        ASSEMBLE MODULES FOR
000029 //*        360S-SM-023 OS/360 SORT/MERGE FOR MVS 3.8
000030 //*
000031 //*************************************************************
000032 //*
000033 //IERABA   EXEC ASMPROJ,HLQ=SORT,PROJECT=MVS38,M=IERABA,SOUT='A'
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -   345 LINE(S) EXCLUDED
000379 //T1A13   JOB  111,'ASM SORT/MERGE',   <-- CUSTOMIZE FOR INSTALLATION
000380 //             CLASS=S,MSGCLASS=C      <-- CUSTOMIZE FOR INSTALLATION
000381 //*
000382 //*************************************************************
000383 //*
000384 //*        ASSEMBLE MODULES FOR
000385 //*        360S-SM-023 OS/360 SORT/MERGE FOR MVS 3.8
000386 //*
000387 //*************************************************************
000388 //*
000389 //IER8ON   EXEC ASMPROJ,HLQ=SORT,PROJECT=MVS38,M=IER8ON,SOUT='A'
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  9 LINE(S) EXCLUDED
000399 //IER9PA   EXEC ASMPROJ,HLQ=SORT,PROJECT=MVS38,M=IER9PA,SOUT='A'
000400 //
```

Figure 11 ASMALL Assembly Job Stream

The assembly process has been divided into 13 separate jobs to avoid spool depletion because of the large volume of print output produced by the assembler steps. Each of the 13 jobs will assemble approximately 20 modules. The assembly jobs invoke the cataloged procedure ASMPROJ. The ASMPROJ cataloged procedure was copied to the SYS2.PROCLIB data set as part of the installation process.

The ASMPROJ cataloged procedure invokes the INITOBJ program to process the object deck before it is stored in the object library. If the INITOBJ program is not already installed then it can be installed by running the job stream provided in member INSTINOB in the SORT.MVS38.CNTL data set. The INITOBJ program is already installed in the TK4- environment.

After the completion of the assembly jobs two load module libraries are created by a job to link edit the object decks into load modules. Member LINKSM in the SORT.MVS38.CNTL data set contains the link edit job stream.

```
REVEDIT  SORT.MVS38.CNTL(LINKSM) - 1.00                  COLUMNS 00001 00072
COMMAND ===>                                                SCROLL ===> CS
 128KB ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000001 //T1SLK   JOB  111,'LINK SORT/MERGE',  <-- CUSTOMIZE FOR INSTALLATION
000002 //             CLASS=S,MSGCLASS=C      <-- CUSTOMIZE FOR INSTALLATION
000003 //*
000004 //*********************************************************************
000005 //*
000006 //*        LINK EDIT
000007 //*
000008 //*        360S-SM-023 OS/360 SORT/MERGE FOR MVS 3.8
000009 //*
000010 //*        DELETE AND ALLOCATE TARGET LIBRARIES
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 36 LINE(S) EXCLUDED
000047 //*        LINK EDIT DEFINITION PHASE MODULES
000048 //*        TARGET -> LOADLIB
000049 //*
000050 //*********************************************************************
000051 //*
000052 //SMLINK  EXEC PGM=IEWL,REGION=1024K,
000053 //             PARM='NCAL,MAP,LIST,XREF'
000054 //SYSUT1   DD  UNIT=VIO,SPACE=(TRK,(40,20))
000055 //SYSPRINT DD  SYSOUT=*
000056 //SYSLMOD  DD  DSN=SORT.MVS38.LOADLIB,
000057 //             DISP=SHR
000058 //SMOBJECT DD  DSN=SORT.MVS38.OBJ,DISP=SHR
000059 //SYSLIN   DD  *
000060  INCLUDE SMOBJECT(IERRCM)
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 74 LINE(S) EXCLUDED
000136 //*        LINK EDIT ASSIGNMENT AND RUN TIME MODULES
000137 //*        TARGET -> SORTLIB
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  3 LINE(S) EXCLUDED
000141 //SMLIB   EXEC PGM=IEWL,REGION=1024K,
000142 //             PARM='NCAL,MAP,LIST'
000143 //SYSUT1   DD  UNIT=VIO,SPACE=(TRK,(40,20))
000144 //SYSPRINT DD  SYSOUT=*
000145 //SYSLMOD  DD  DSN=SORT.MVS38.SORTLIB,DISP=SHR
000146 //SMOBJECT DD  DSN=SORT.MVS38.OBJ,DISP=SHR
000147 //SYSLIN   DD  *
000148  ENTRY IER8BN
000149  INCLUDE SMOBJECT(IER8BN)
000150  IDENTIFY IER8BN('360SSM023 OS/360 SORT/MERGE FOR MVS 3.8')
000151  NAME IER8BN(R)
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -   814 LINE(S) EXCLUDED
```

Figure 12 LINKSM Link Edit Job Stream

Update the JOB statement to conform to the installation standards and submit the job.

As part of the two link edit steps every Sort/Merge Program CSECT is provided with an IDENTIFY statement so that load modules for this release of the Sort/Merge Program can be identified compared to load modules from the previous release that do not have IDENTIFY statements.

A REVIEW Browse of any of the Sort/Merge Program load modules will show the presence or absence of the IDENTIFY text to confirm the origins of the load module.

```
 Browse substituted ------------------------------------------- Line 1 Col 1 80
 Command ===>                                                   Scroll ===> CS
1       10        20        30        40        50        60        70        80
+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
.......................IERRCO  ........SORT    .....-..
Ø|.....................................................................................
Ø..5752SC104 ....|.Êà.                   (BIND on 16-11-19 at 17:24:43   V03 M08)
Ø..Ø..5741SC103 ....|                    (TRAN on 16-11-19 by 5741SC103  V02 M01)
Ø.h....|.360SSM023 OS/360 SORT/MERGE FOR MVS 3.8
.............. .......
å00..IERRCO 11/19/16 17.23O+}..+ ..Q..á.Ø.........&.}.&}...JY.}çØR..á0Øî...&....
.......................ç..._...È...y
******EOF-TTR=00060C************ BOTTOM OF DATA ***************718-BYTES********
```

Figure 13 REVIEW Browse of IERRCO00

Figure 13 shows a REVIEW browse of the load module IERRCO00 and its alias of SORT. The CSECT IERRCO is identified as belonging to 360SSM023 OS/360 SORT/MERGE FOR MVS 3.8.

Assembling individual modules

If individual modules are being changed and assembled then care must be taken to avoid module mismatches that will result in the Sort/Merge Program failing with random error conditions. The Sort/Merge Program has three different types of modules:

1. Definition phase modules
2. Assignment phase modules
3. Run phase modules.

Definition phase modules can be changed and assembled individually without mismatch concerns unless changes are being made to the major control blocks being the CPI and PPI. The definition phase modules are those modules that are link edited into the SYS2.LINKLIB data set.

Assignment and run time modules reside in the SYS1.SORTLIB data set. They have a unique relationship with each other. Each run time module has a corresponding assignment phase module that is run prior to the run time module receiving control to configure the module for the specific sorting operation. The assignment module is responsible for initializing variables and making changes to the code in the run time module for such things as changed offsets due to sorting variable-length records. To ensure the assignment module updates the correct location in the run time module the source of the run time module is included into the assembly of the assignment module as a DSECT. Therefore, any changes to and assembly of a run time module must also include the assembly of its corresponding assignment module. Comment statements at the beginning of the source code for each run time module identify the corresponding assignment phase module.

5. Diagnostic Facilities

The OS/360 Sort Merge Program for MVS 3.8 has extensive build-in diagnostic facilities.

The diagnostic mode of the Sort/Merge Program can be activated by the presence of a SORTDIAG JCL DD statement in the step job stream or by coding the DIAGSIM parameter on a DEBUG control statement. The JCL EXEC PARM parameter DIAG or coding DIAG in an ATTACH/LINK/XCTL parameter list is ignored without any error being noted.

If a SORTDIAG DD statement placed in the job stream is used to activate the diagnostic mode of the Sort/Merge Program then the diagnostic messages will be written to the SORTDIAG message data set.

The DCB parameters for the SORTDIAG message data set are set to RECFM=FBA and LRECL=121. Any BLKSIZE which is a multiple of the LRECL can be provided. The SORTDIAG output is usually directed to the output class set by the JOB statement MSGCLASS parameter. The example SORTDIAG DD statement below shows a typical SORTDIAG DD statement.

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

If the DIAGSIM DEBUG control statement parameter is used to activate the diagnostic mode of the Sort/Merge Program then the diagnostic messages are written to the SYSOUT message data set. As the name DIAGSIM implies it simulates the presence of a SORTDIAG JCL DD statement.

Diagnostic mode output messages are written after the initial heading message IER900I. Figure 14 shows an example of the output produced when diagnostic mode has been activated and no additional diagnostic options selected.

```
SYS16341.T132940.RA000.T1.JOB06723 ------------------------------------------------------------------------------ Line 754 Col 2 133
 Command ===>                                                  Scroll ===>  C
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER900I Initial Diagnostic Options -
IER070I Control Stmts from Program Parameter List
IER070I Control Stmts SORT FIELDS=(5,10,CH,D),SIZE=E36000,DYNALLOC=(3390,3)
IER070I Control Stmts RECORD LENGTH=400,TYPE=F
IER036I Blocking =    69
IER037I Records in RSA =  1177
IER961I Sort Technique - BALN
IER962I Phase 1,Number of Buffers =   3,Buffer Size = 27616
IER962I Phase 2,Number of Buffers =  18,Buffer Size = 27616
IER962I Phase 3,Number of Buffers =  20,Buffer Size = 27616
IER963I Storage =  524288
IER964I Phase 1 Storage =  551010
IER964I Phase 2 Storage =  511280
IER964I Phase 3 Storage =  456209
IER965I Merge Order =  16
IER981I SORTWK01,190,3390,WORK03,  002 00 -   014 13,Tracks =   194
IER981I SORTWK02,190,3390,WORK03,  014 14 -   027 12,Tracks =   194
IER981I SORTWK03,190,3390,WORK03,  027 13 -   040 11,Tracks =   194
IER038I Estimated maximum records =   53406
IER911I Getmain - Out Buffer,L= 006BE0,A= 0A9420
IER911I Getmain - In Buffer ,L= 000198,A= 0A4C88
IER911I Getmain - Gen Area  ,L= 0020B8,A= 0B0F48
IER910I Generated Storage End Addr - 0B3000
IER903I RSA Table Addr - 0B2F90
IER901I Input Buffer Table Addr - 0B2F88
IER904I TREE Addr from 0B10E8 to 0B2F88
IER905I MOVE Routine Addr - 0B10DA
IER906I DCB Table Addr - 0B0FBC
IER902I Output Buffer Addr - 0A9428, 000000
IER907I Output CCW Addr - 0B0FA0
IER908I Output IOB Addr - 0B0FE0
IER045I End of Sort Phase
IER049I Skip Merge Phase
IER940I Generated Storage End Addr - 0AA000
IER941I Input Buffer Table Addr - 0A9EA0
IER944I DCB Table Addr - 0A9A74
IER943I MOVE Routine Addr - 0A9A66
IER942I Output Buffer Addr - 0A4C90, 0AA298
IER945I Input CCW Addr - 0A9760
IER055I Records Inserted   36000, Records Deleted   36000
IER054I Records In        , Records Out
IER052I End of Sort
```

Figure 14 Example Diagnostic message output

Additional diagnostic facilities are activated by providing a DEBUG Control Statement to the Sort/Merge Program as part of the control statement input stream.

The rules for coding the DEBUG control statement are the same as all the other Sort/Merge Program control statements. The general rules are fully described in the related document OS/360 Sort/Merge for MVS 3.8 Application Programming Guide, Chapter 2.2 Control Statement Format.

5.1 DEBUG Control Statement

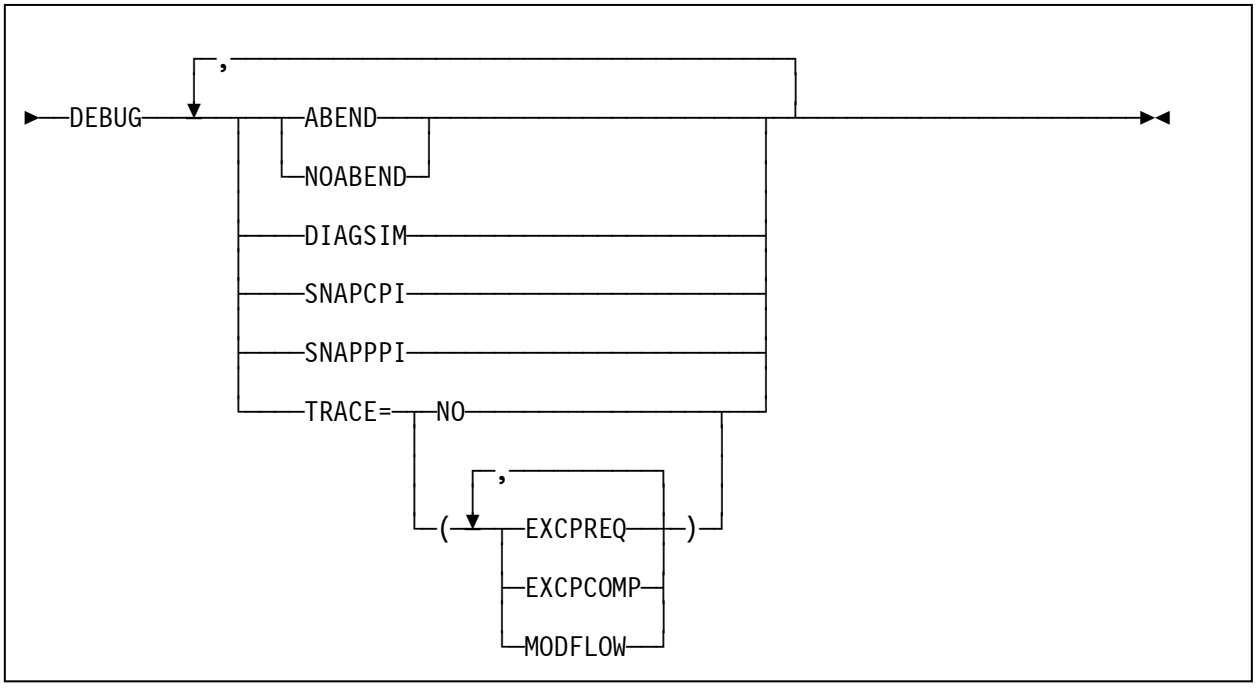
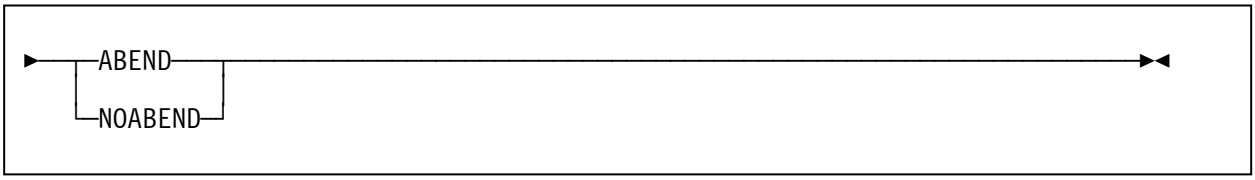


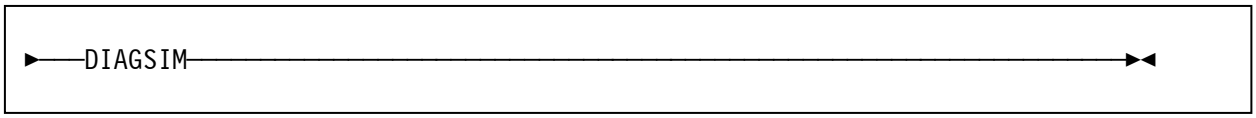
Figure 15 DEBUG Control Statement1617

ABEND or NOABEND



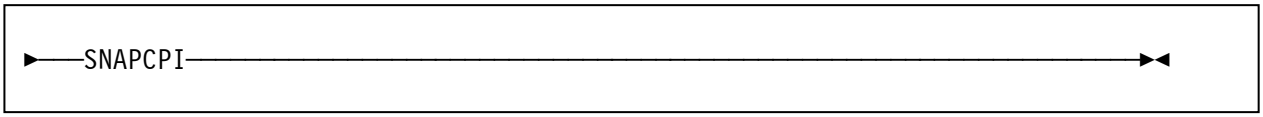
This operand provides an ability to override the installation customization parameter ERET. When the Sort/Merge Program detects a critical error then, depending on the setting of ABEND or NOABEND, the Sort/Merge Program will abend or terminate with a return code of 16. Use of the ABEND parameter can assist diagnosis of a critical error by producing a dump.

DIAGSIM



The diagnostic mode of the Sort/Merge Program can be activated either by the presence of a SORTDIAG JCL DD statement in the job step input stream or by coding the DIAGSIM parameter on a DEBUG statement. When the DIAGSIM parameter is used then all diagnostic messages are written to the message SYSOUT data set. If it is not possible to change the JCL running the sorting or merging job to include a SORTDIAG JCL DD statement then the DIAGSIM parameter can be used to activate the diagnostic mode of the Sort/Merge Program.

SNAPCPI



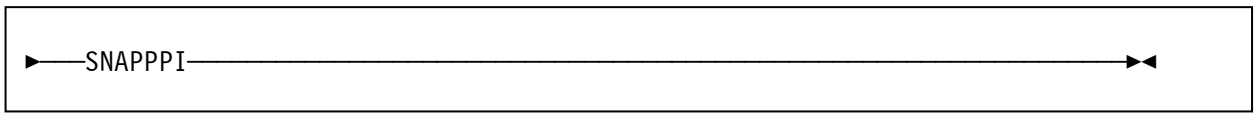
The SNAPCPI operand causes the Sort/Merge Program to print dump the CPI control block and the control statement analysis and reduction area after each definition phase module has completed processing. The output is written to the SORTDIAG message data set or to the SYSOUT message data set depending on how the diagnostic mode of the Sort/Merge Program was activated. The message IER982 identifies the CPI and the module that just completed processing. The message IER986 identifies the related control statement analysis and reduction area. All options and parameters gathered during the definition phase of the Sort/Merge Program are stored in the CPI. An examination of the CPI print dumps provide a tool for determining processing errors during the definition phase processing

Figure 16 shows an example of the output produced when SNAPCPI is active. The print dump will be generated for every module invoked during the definition phase.

```
 SYS16341.T180809.RA000.T1.JOB06724 ------------------------------------------------------------------------------ Line 764 Col 2 133
 Command ===>                                                Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
JOB T1IVP1           STEP EXECIVP1        TIME 180344   DATE 16341    ID = 001    CPUID = FD0001273033       PAGE 0001
-STORAGE
IER982I CPI post IERRCY Processing
0A4C40                    00000000 000A5DC0    00000000 500A92EC 12C8C0FC 500A92EC   *        ~~~~~~~){~~~~&~k~~H{~&~k~*
0A4C60   00C68010 00000000 000A9A8C 000A9A8C    0000003B 000A7DD8 009C00AC 000A91F2   *~F~~~~~~~~~~~~~~~~~~~~~~~~'Q~~~~~~j2*
0A4C80   00000046 00C68010 500A92EC 00C68010    C3D7C9C1 40404040 C2C5E3C1 00000000   *~~~~~F~~&~k~~F~~CPIA    BETA~~~~*
0A4CA0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
      LINE 0A4CC0 SAME AS ABOVE
0A4CE0   00000000 00000000 00045000 00004982    27864020 00000000 00000000 00000000   *~~~~~~~~~~~~&~~~~b~f ~~~~~~~~~~~~~~~~*
0A4D00   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4D20   0000000A 5C4A0A5C A4000000 00000000    00000000 00000000 00000000 00000000   *~~~~~*¢~*u~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4D40   00000000 00000000 00000000 00000000    00000000 E2D6D9E3 F3F3F9F0 40404040   *~~~~~~~~~~~~~~~~~~~~~~~~SORT3390    *
0A4D60   00070032 0000000A 00000000 00200000    00020000 00010000 00018000 00080000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4D80   000A7BA8 E2E8E2D6 E4E34040 C9C5D9D7    C1D9D440 006D0011 00000000 00000000   *~~~#ySYSOUT  IERPARM ~ ~~~~~~~~~~*
0A4DA0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4DC0   00000000 0000179C 000B2860 000B28F1    000B2860 000B2895 000B2896 000B28AE   *~~~~~~~~~~~~ -~~~1~~~-~~~n~~~0~~~~~*
0A4DE0   00000000 00000000 00000000 00000000    000B28AF 000B28F0 00000000 000B3860   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~0~~~~~~~-~*
0A4E00   000B405F 00000000 02000020 E3F1C9E5    D7F14040 6BC5E7C5 C3C9E5D7 F1400000   *~~ ¬~~~~~~~~T1IVP1  ,EXECIVP1 ~~*

JOB T1IVP1           STEP EXECIVP1        TIME 180344   DATE 16341    ID = 001 .                          PAGE 0002

                  DUMP INDEX
                  ----------
 DATA AREAS                          PAGE NUMBER
-----------                          -----------
 STORAGE AREAS.......................  0001
END OF DUMP
JOB T1IVP1           STEP EXECIVP1        TIME 180344   DATE 16341    ID = 002    CPUID = FD0001273033       PAGE 0001
-STORAGE
IER986I Cntl Stmt area post IERRCY processing
0B2860   E2D6D9E3 40C6C9C5 D3C4E27E 4DF56BF1    F06BC3C8 6BC45D6B E2C9E9C5 7EC5F3F6   *SORT FIELDS=(5,10,CH,D),SIZE=E36*
0B2880   F0F0F06B C4E8D5C1 D3D3D6C3 7E4DF3F3    F9F06BF3 5D40D9C5 C3D6D9C4 40D3C5D5   *000,DYNALLOC=(3390,3) RECORD LEN*
0B28A0   C7E3C87E F4F0F06B E3E8D7C5 7EC640C4    C5C2E4C7 40E2D5C1 D7C3D7C9 6BE2D5C1   *GTH=400,TYPE=F DEBUG SNAPCPI,SNA*
0B28C0   D7D7D7C9 40404040 40404040 40404040    40404040 40404040 40404040 40404040   *PPPI                            *
0B28E0   40404040 40404040 40404040 40404040    40000000 00000000 00000000 00000000   *                    ~~~~~~~~~~~~~~~~*
0B2900   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
      LINES 0B2920-0B3840 SAME AS ABOVE
0B3860   02E2D5C1 D7C3D7C9 FF00E2D5 C1D7D7D7    C9FF0000 00000000 00000000 00000000   *~SNAPCPI~~SNAPPPI~~~~~~~~~~~~~~~~*
0B3880   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
      LINES 0B38A0-0B3FC0 SAME AS ABOVE
0B3FE0   00000000 00000000 00000000 00000000    00000000 00000000 00000000            *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~    *

JOB T1IVP1           STEP EXECIVP1        TIME 180344   DATE 16341    ID = 002 .                          PAGE 0002
```

Figure 18 Example print dump of the CPI19

SNAPPPI



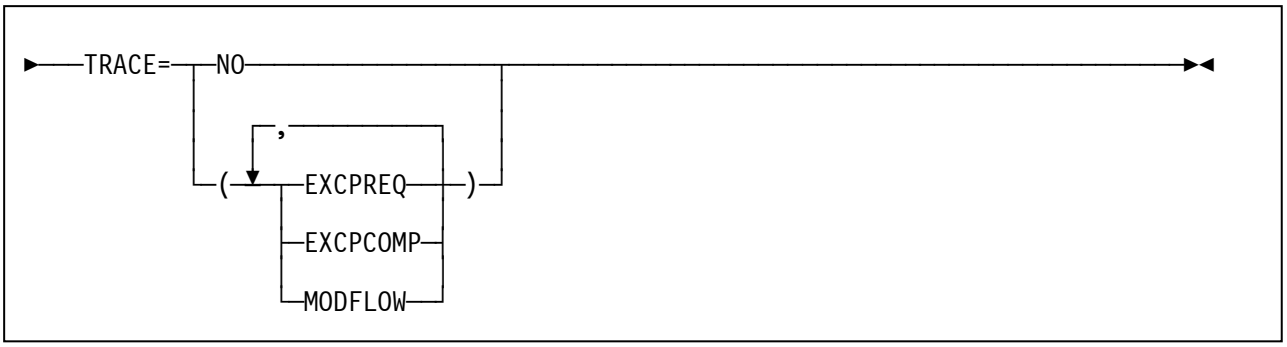
The SNAPPPI operand causes the Sort/Merge Program to print dump the PPI control block when it has been initialized in the definition phase and then after each definition phase module has completed processing. The print dump output is written to the SORTDIAG message data set or to the SYSOUT message data set depending on how the diagnostic mode of the Sort/Merge Program was activated. The message IER987 identifies the PPI and the module that just completed processing. The settings in the PPI control and configure the run time phase of the Sort/Merge Program. An examination of PPI print dumps provide a tool for determining processing errors during the final processing of the sort definition phase prior to the commencement of the run time phase.

Figure 17 shows an example of the output produced when SNAPPPI is active.

```
 SYS16341.T180809.RA000.T1.JOB06724 ---------------------------------------------------------------------------- Line 1338 Col 2 133
 Command ===>                                                Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
JOB T1IVP1           STEP EXECIVP1        TIME 180344   DATE 16341    ID = 001    CPUID = FD0001273033       PAGE 0001
-STORAGE
IER987I PPI - AFTER IER    PROCESSING
0A5680                     00000000 000A5DC0    00000000 400A6D64 00FB903A 000A7DE4   *       ^^^^^^^){^^^^ ^ ^^^^^^^'U*
0A56A0   000A6D30 00000081 00050DB4 000A7040    0000001B 00000042 000A54E8 000A6CC8   *^^ ^^^^a^^^^^^^^ ^^^^^^^^^^Y^^%H*
0A56C0   000A6C20 000A7DE4 00000043 000A6FEC    000020B8 00000000 00000000 000A5C4A   *^^%^^^'U^^^^^^^?^^^^^^^^^^^^^^^*¢*
0A56E0   00000001 00000003 0000001E 00000012    0000001E 00000000 00010004 00090004   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0A5700   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0A5720   019000C2 019000C2 019000C2 00000000    00000000 00000000 00000000 00000000   *^^^B^^^B^^^B^^^^^^^^^^^^^^^^^^^^*
0A5740   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
      LINES 0A5760-0A5780 SAME AS ABOVE
0A57A0   00000000 00000000 01000000 00000001    02000000 00000001 03000000 00000001   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0A57C0   04000000 00000001 05000000 00000001    06000000 00000001 07000000 00000001   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0A57E0   08000000 00000001 09000000 00000001    0A000000 00000001 0B000000 00000001   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0A5800   0C000000 00000001 0D000000 00000001    0E000000 00000001 0F000000 00000001   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0A5820   10000000 00000001 11000000 00000001    00000000 0000C101 00000000 0000C101   *^^^^^^^^^^^^^^^^^^^^^^^^A^^^^^^^A^*
0A5840   00000000 0000C101 00000000 00000001    00000000 00000001 00000000 00000001   *^^^^^^^A^^^^^^^^^^^^^^^^^^^^^^^^*
0A5860   00000000 00000001 00000000 00000001    00000000 00000001 00000000 00000001   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
     LINE 0A5880 SAME AS ABOVE
0A58A0   00000000 00000001 00000000 00000001    00000000 00000001 A8B14036 52120002   *^^^^^^^^^^^^^^^^^^^^^^^^y^ ^^^^^*
0A58C0   49822786 40200000 C2C5E3C1 00000000    00000000 00000000 00000000 00000000   *^^b^f ^^^^BETA^^^^^^^^^^^^^^^^^^^*
0A58E0   00000000 00000000 00000000 00050DB4    00008CA0 00000194 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^m^^^^^^^*
0A5900   00000499 00450001 12140000 00000000    00000000 00000000 00000000 01900190   *^^^r^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0A5920   01900000 00000010 00100000 00040009    00000000 00000000 00000000 00030000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0A5940   00000190 00006BE0 02040190 00000AC0    00000AB8 00000000 000A5CA4 0007D4AE   *^^^^^^^,\^^^^^^^{^^^^^^^^^*u^^M^^*
0A5960   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
      LINE 0A5980 SAME AS ABOVE
0A59A0   00000000 00000000 00000000 00000000    000A55A8 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^y^^^^^^^^^^^*
0A59C0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
      LINE 0A59E0 SAME AS ABOVE
0A5A00   00000000 00000000 00000000 00000000    00000000 00000000 000A5150 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^&^^^^*
0A5A20   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
      LINES 0A5A40-0A5A80 SAME AS ABOVE
0A5AA0   00000000 00000000 00000000 00000000    E2D6D9E3 00000000 000000F0 F0F00000   *^^^^^^^^^^^^^^^^^^SORT^^^^^^^000^^*
0A5AC0   0000F3F3 F9F04040 40400003 0000000A    E2E8E2D6 E4E34040 C9C5D9D7 C1D9D440   *^^3390    ^^^^^^SYSOUT  IERPARM *
0A5AE0   006D0011 02000020 E3F1C9E5 D7F14040    6BC5E7C5 C3C9E5D7 F1400000 000007F1   *^ ^^^^^^T1IVP1  ,EXECIVP1 ^^^^^1*
0A5B00   070058F0 D48C07FF 07F10700 131158F0    D48C07FF 00000000 00000000 00000000   *^^^0M^^^^1^^^^0M^^^^^^^^^^^^^^^^*
0A5B20   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
      LINE 0A5B40 SAME AS ABOVE
0A5B60   00000000 6D480002 48000003 35600004    2A180005 22B80006 1D680007 19600008   *^^^^^ ^^^^^^^^_^^^^^^^^^^^^^^^_^^*
0A5B80   16480009 13B8000A 0000000B 004E0000    00017B66 00017AE0                     *^^^^^^^^^^^^^+^^^^#^^^:\         *

JOB T1IVP1           STEP EXECIVP1        TIME 180344   DATE 16341    ID = 001 .                          PAGE 0002
```

Figure 20 Example print dump of the PPI

TRACE



The TRACE operand is used to activate the Sort/Merge Program's most powerful diagnostic tools.

NO

deactivates all tracing options.

EXCPREQ

The EXCPREQ operand activates the tracing of all EXCP I/O requests made to the intermediate storage SORTWKdd data sets. For channel programs that write data, the first 256 bytes of the I/O area are provided in the trace together with a fully formatted IOB. The CCW chain used for the I/O operation is also formatted. No data is formatted for channel programs that read data. The message IER983I identifies the Sort/Merge Program module and the offset within that module that issued the EXCP request.

Note that for a sorting operation with a large number of records, resulting in many I/O operations to the intermediate storage SORTWKdd data sets, a considerable number of output lines of trace data will be generated.

Figure 18 shows an example of the output generated for a sorting operation using the BALN technique when EXCPREQ is active.

```
 SYS16342.T094229.RA000.T1.JOB06726 ------------------------------------------------------------------------------- Line 1503 Col 2 133
 Command ===>                              SYS16342.T094229.RA000. Scroll ===> --
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER983I EXCP Issued by IERRPB+0190
IOB 0B3FE0
    -08 IERECB   00000000  -04 IERALTCW   000B3F88
    +00 IOBFLAG1 40 IOBCMDCH             +01 IOBFLAG2 00 +02 IOBSENS0 00 +03 IOBSENS1 00
    +04 IOBECBCC 00 +05 IOBECBPB 0B3FD8  +08 IOBFLAG3 00 +09 IOBCMDA  000000
    +12 IOBUSTAT 00                      +13 IOBCSTAT 00                   +14 RESIDUAL      0
    +16 IOBSIOCC 00 +17 IOBSTRTB 0B3FA0  +20 IOBFLAG4 00 +21 IOBDCBPB 0B4068    DDNAME SORTWK01
    +24 IOBRESTR 00000000
    +32 IOBSEEK  +36 IOBCC   002  +38 IOBHH 00     +40 IOBR  00
CCW 0B3FA0  +00 31 SEARIDEQ      +01 DATA  0B4003 +04 FLAGS 60 CC+SLI         +06 COUNT     5
                      CC   002         HH 00           R  00
CCW 0B3FA8  +00 08 TIC           +01 DATA  0B3FA0 +04 FLAGS 60 CC+SLI         +06 COUNT     0
CCW 0B3FB0  +00 1D WRITECKD      +01 DATA  0AC420 +04 FLAGS 20 SLI           +06 COUNT 27616
                      CC   002         HH 00           R  01          KL   00  DL    27608
    0AC428 C8C8C8C8 2B64FCC5 E95A7C7B 5B6C5F5C 4D5D6D4F 617A5EC5 C6C7C8C9 D1D2D3D4 *HHHH~~~EZ!@#$%¬*()_|/:;EFGHIJKLM*
    0AC448 D5D6D7D8 D9E2E3E4 E5E6E7E8 E95A7C7B 5B6C5F5C 4D5D6D4F 617A5EC1 C2C3C4C5 *NOPQRSTUVWXYZ!@#$%¬*()_|/:;ABCDE*
    0AC468 C6C7C8C9 D1D2D3D4 D5D6D7D8 D9E2E3E4 E5E6E7E8 E95A7C7B 5B6C5F5C 4D5D6D4F *FGHIJKLMNOPQRSTUVWXYZ!@#$%¬*()_|*
    0AC488 617A5EC1 C2C3C4C5 C6C7C8C9 D1D2D3D4 D5D6D7D8 D9E2E3E4 E5E6E7E8 E95A7C7B */:;ABCDEFGHIJKLMNOPQRSTUVWXYZ!@#*
    0AC4A8 5B6C5F5C 4D5D6D4F 617A5EC1 C2C3C4C5 C6C7C8C9 D1D2D3D4 D5D6D7D8 D9E2E3E4 *$%¬*()_|/:;ABCDEFGHIJKLMNOPQRSTU*
    0AC4C8 E5E6E7E8 E95A7C7B 5B6C5F5C 4D5D6D4F 617A5EC1 C2C3C4C5 C6C7C8C9 D1D2D3D4 *VWXYZ!@#$%¬*()_|/:;ABCDEFGHIJKLM*
    0AC4E8 D5D6D7D8 D9E2E3E4 E5E6E7E8 E95A7C7B 5B6C5F5C 4D5D6D4F 617A5EC1 C2C3C4C5 *NOPQRSTUVWXYZ!@#$%¬*()_|/:;ABCDE*
    0AC508 C6C7C8C9 D1D2D3D4 D5D6D7D8 D9E2E3E4 E5E6E7E8 E95A7C7B 5B6C5F5C 4D5D6D4F *FGHIJKLMNOPQRSTUVWXYZ!@#$%¬*()_|*
```

Figure 21 Example EXCPREQ data

EXCPCOMP

The EXCPCOMP operand activates the tracing of all WAIT requests for I/O completion to the intermediate storage SORTWKdd data sets. The trace is entered after the WAIT for the I/O event has been posted complete. For channel programs that read data, the first 256 bytes of the I/O area are provided in the trace together with a fully formatted IOB. The CCW chain used for the I/O operation is also formatted. No data is formatted for channel programs that write data. The message IER984I identifies the Sort/Merge Program module and the offset within that module that issued the WAIT request.

Note that for a sorting operation with a large number of records, resulting in many I/O operations to the intermediate storage SORTWKdd data sets, a considerable number of output lines of trace data will be generated.

Figure 19 shows an example of the output generated for a sorting operation using the BALN technique when EXCPCOMP is active.

```
 SYS16342.T095806.RA000.T1.JOB06727 ------------------------------------------------------------------------------- Line 3645 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER984I WAIT on EXCP Issued by IERRGE+00C8
IOB 0A9DA8
    -08 IERECB   7F000000  -04 IERALTCW   020A9D20
    +00 IOBFLAG1 40 IOBCMDCH              +01 IOBFLAG2 00 +02 IOBSENS0 00 +03 IOBSENS1 00
    +04 IOBECBCC 7F +05 IOBECBPB 0A9DA0   +08 IOBFLAG3 00 +09 IOBCMDA  0A9D20
    +12 IOBUSTAT 0C CE+DE                 +13 IOBCSTAT 00                    +14 RESIDUAL      0
    +16 IOBSIOCC 40 +17 IOBSTRTB 0A9D08   +20 IOBFLAG4 00 +21 IOBDCBPB 0A9E68    DDNAME SORTWK02
    +24 IOBRESTR 00000000
    +32 IOBSEEK  +36 IOBCC   003  +38 IOBHH 05     +40 IOBR  01
CCW 0A9D08  +00 31 SEARIDEQ       +01 DATA  0A9DCB +04 FLAGS 60 CC+SLI         +06 COUNT     5
                      CC   003         HH 05           R  01
CCW 0A9D10  +00 08 TIC            +01 DATA  0A9D08 +04 FLAGS 60 CC+SLI         +06 COUNT     0
CCW 0A9D18  +00 06 READDATA       +01 DATA  11C420 +04 FLAGS 20 SLI            +06 COUNT 27608
    11C420 C8C8C8C8 C52B64FC D8D9E2E3 E4E5E6E7 E8E95A7C 7B5B6C5F 5C4D5D6D 4F617A5E *HHHHE~~~QRSTUVWXYZ!@#$%¬*()_|/:;*
    11C440 C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E95A7C 7B5B6C5F *ABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%¬*
    11C460 5C4D5D6D 4F617A5E C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E95A7C **()_|/:;EFGHIJKLMNOPQRSTUVWXYZ!@*
    11C480 7B5B6C5F 5C4D5D6D 4F617A5E C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3 *#$%¬*()_|/:;ABCDEFGHIJKLMNOPQRST*
    11C4A0 E4E5E6E7 E8E95A7C 7B5B6C5F 5C4D5D6D 4F617A5E C1C2C3C4 C5C6C7C8 C9D1D2D3 *UVWXYZ!@#$%¬*()_|/:;ABCDEFGHIJKL*
    11C4C0 D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E95A7C 7B5B6C5F 5C4D5D6D 4F617A5E C1C2C3C4 *MNOPQRSTUVWXYZ!@#$%¬*()_|/:;ABCD*
    11C4E0 C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E95A7C 7B5B6C5F 5C4D5D6D *EFGHIJKLMNOPQRSTUVWXYZ!@#$%¬*()_*
    11C500 4F617A5E C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E95A7C *|/:;ABCDEFGHIJKLMNOPQRSTUVWXYZ!@*
```

Figure 22 Example EXCPCOMP data

MODFLOW

The Sort/Merge Program consists of a large number of load modules that are loaded and deleted during the running of a sorting or merging operation. The MODFLOW operand provides a trace of the modules as they are loaded, called and deleted by the controlling assignment module for the selected sorting or merging sequencing technique. The Sort/Merge Program does not implement standard operating system conventions in the way control is passed from module to module during the running phase of the sorting or merging operation. Therefore, the flow of control between the modules during the run time phase of the Sort/Merge Program is not traced because control is passed directly from module to module at numerous branch entry points.

The message IER980I identifies the name of a module being called, having been previously loaded or resident in another load module. The message IER988I identifies modules being loaded. This message has two formats. The first format identifies the normal loading of a module. The second format identifies the loading of a module that will form part of the group of modules that will implement a phase of the selected sort or merge sequencing technique. The function provided by the loaded module is listed after the storage address of the module. The message IER989I identifies the name of a module being deleted.

Figure 20 shows an example of the output generated for a phase of a BALN technique sorting operation when MODFLOW is active.

```
 SYS16342.T101732.RA000.T1.JOB06729 ------------------------------------------------------------------------------- Line 791 Col 2 133
 Command ===>                                                Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER980I Calling IERAO1
IER988I IERRC6 Loaded at 0A60C0
IER989I Deleting IERRC6
IER988I IERROK Loaded at 0A6478 - ALG
IER988I IERRDD Loaded at 0A6238 - DEB
IER988I IERROB Loaded at 0A7640 - NET
IER988I IERRBC Loaded at 0A5070 - BLK
IER988I IERRPB Loaded at 0A70F8 - WRT
IER988I IERRGA Loaded at 0A7060 - EOF
IER988I IERRMA Loaded at 0A6120 - RMA
IER988I IERAMA Loaded at 0A8D98 - AMA
IER988I IERAP1 Loaded at 0A7008 - OPEN
IER988I IERAPG Loaded at 0A8608
IER980I Calling IERAPG
```

Figure 23 Example MODFLOW generated data

6. Performance Considerations

There are many factors that impact the performance of the Sort/Merge Program running in the MVS 3.8 operating system environment. This chapter identifies the major factors and their impact on the performance of the Sort/Merge Program. The topics include:

Selection of DASD Unit Type for intermediate storage

Storage allocation for the Sort/Merge Program

Selection of the Sort/Merge Program sequencing technique

Length of records being sorted

Use of compressed or non-compressed Hercules DASD

PC Configuration.

Each topic is discussed and benchmarking results provided where appropriate. Recommendations are make for each topic on obtaining optimum performance for sorting operations.

Selection of DASD Unit Type for intermediate storage

The DASD Unit Type selected for intermediate storage has a major impact on the performance of the Sort/Merge Program. The amount of data that will be transferred to and from the intermediate storage data sets for any given sorting operation will be the same for all DASD unit types. The performance improvement is a result of the reduced number of I/O operations needed to transfer the data to and from the intermediate storage data sets. With larger I/O buffers the Sort/Merge Program will not have to schedule as many I/O requests to refresh and empty the I/O buffers compared to using smaller I/O buffers. This will result in less EXCP requests made to the operating system where EXCP processing has a significant instruction path length. An EXCP request that has to PGFIX and then PGFREE eight pages, when using 3390 DASD will take slightly longer than a EXCP request than has to PGFIX and PGFREE two pages for 2314 DASD. However, most of the processing overhead of an EXCP request is independent of the amount of data that will be transferred by the I/O operation.

Note that the use of 3390 DASD will require a significant increase in the amount of storage needed to run a sorting operation due to the increased I/O buffer sizes. Further information on storage requirements are provided in the section discussing the allocation of storage for the Sort/Merge Program.

Figure 21 compares the resource usage of an identical sorting operation, using the same MVS 3.8 environment, Hercules configuration and PC hardware. The sorting operations were run firstly using 2314 DASD and then the sorting operation was re-run using 3390 DASD.

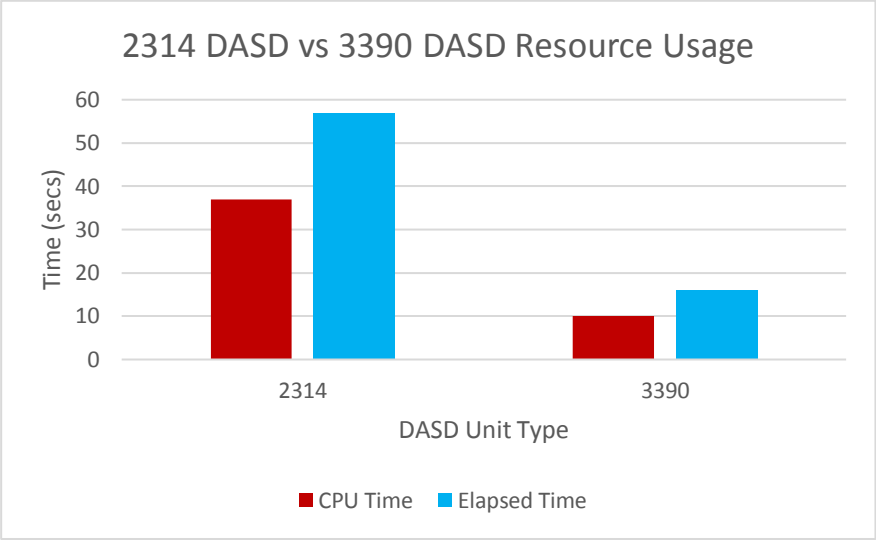


Figure 24 Comparison of 2314 DASD vs 3390 DASD Resource Usage

The reduction in both CPU time and elapsed time using 3390 DASD for intermediate storage is due to the reduction in the number of EXCP requests that were required to complete the sorting operation. The sort using 2314 DASD issued EXCP 68,739 requests while the sort using 3390 DASD issued 17,965 EXCP requests.

The performance improvement gained by using 3390 DASD compared to 2314 DASD has been consistently observed across a wide range of different Hercules configurations and different PC hardware. Using 3390 DASD for intermediate storage is recommended for improving sorting performance.

Storage allocation for the Sort/Merge Program

As mentioned in the section on selecting a DASD Unit Type for intermediate storage, the storage requirements for this version of the Sort/Merge Program have increased greatly from the previous version. This is primarily due to the support and use of DASD Unit Types with a large track capacity. For optimum overlap of the processor and I/O requests the Sort/Merge Program allocates two buffers for each intermediate storage data set provided sufficient storage is available. If sufficient storage is not available then a reduced number of buffers will be allocated resulting in a less than optimum sorting operation.

For 3390 DASD, where half-track blocking is used, each buffer requires approximately 28KB of storage. Double buffering requires approximately 56 KB for each intermediate storage data set used for the sorting operation. Using the BALN sequencing technique with six intermediate storage data sets will require 336 KB of storage for buffers. Additional 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. If a large number of records are going to be sorted that will require more than six intermediate storage data sets and 3390 DASD is going to be used then the storage allocated should be increased by at least 56 KB for each additional intermediate storage data set.

The results of bench marking the Sort/Merge Program with larger allocations of storage have shown that once sufficient storage has been provided for double buffering for each intermediate storage data set and there is adequate storage for the internal Record Storage Area then there is little improvement in sorting performance. The bench marking results are shown in Figure 22.

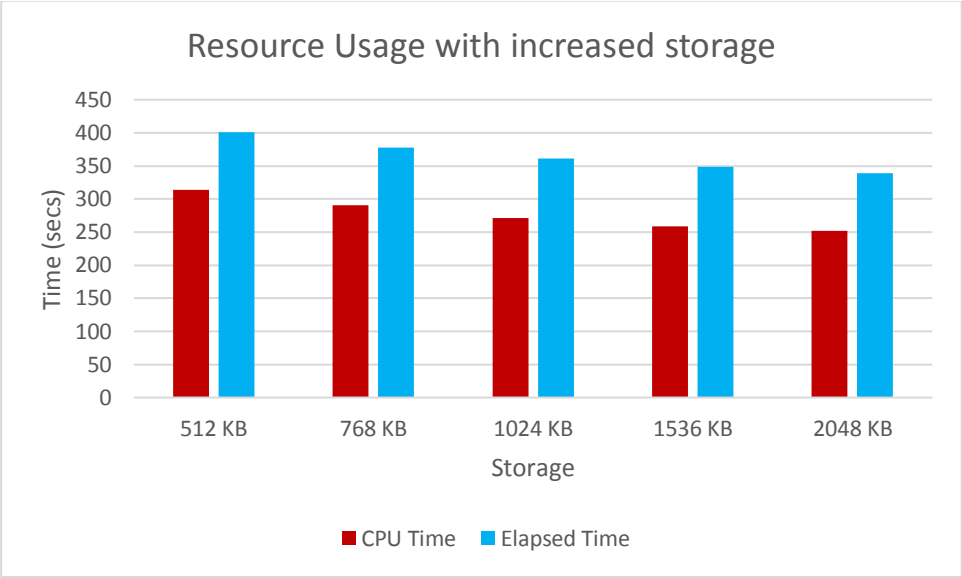


Figure 25 Comparison of Resource Usage with increased storage

The Sort/Merge Program has poor locality of reference when running in a virtual storage system. This is because the Sort/Merge Program functions by moving records from input buffers to the Record Storage Area and then selects, using multiple compare instructions, the next record to be moved out to one of the output buffers. The impact of this logic is that the contents of the RSA and all the I/O buffers are referenced constantly during a sorting operation. Allocating 512 KB of storage to the Sort/Merge Program will result in the need for a working set of at least 128 pages to avoid paging overhead. Allocating additional storage to the Sort/Merge Program above what is needed for optimum performance will result in an increased number of pages required for the working set. In a single user system this will not impact the total system performance but in a system with a workload competing for resources then this will adversely impact the performance of the system.

For the best overall system performance avoid allocating the Sort/Merge Program storage above that needed for effective operation.

Selection of the Sort/Merge Program sequencing technique

When DASD is selected for intermediate storage the Sort/Merge Program will use either the BALN or the CRCX sequencing technique.

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 at least six intermediate storage data sets are required, with a maximum of 17 intermediate storage data sets. For 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. This can result in a change of sequencing technique.

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.

For sorting operations that do not require the larger capacity available with the CRCX sequencing technique then there is the option of using either the BALN technique or the CRCX technique. Either technique can be selected by specifying or providing the number of intermediate work data sets that will force the selection of the required technique, being less than seven for BALN and seven or more for CRCX.

Figure 23 compares the resource usage of a number of sorting operations with different record lengths, using the same MVS 3.8, Hercules configuration and PC hardware. Six intermediate work data sets were used for each sorting operation. For comparison purposes the Sort/Merge Program was forced to use the CRCX sequencing technique as only 6 intermediate work data sets were provided.

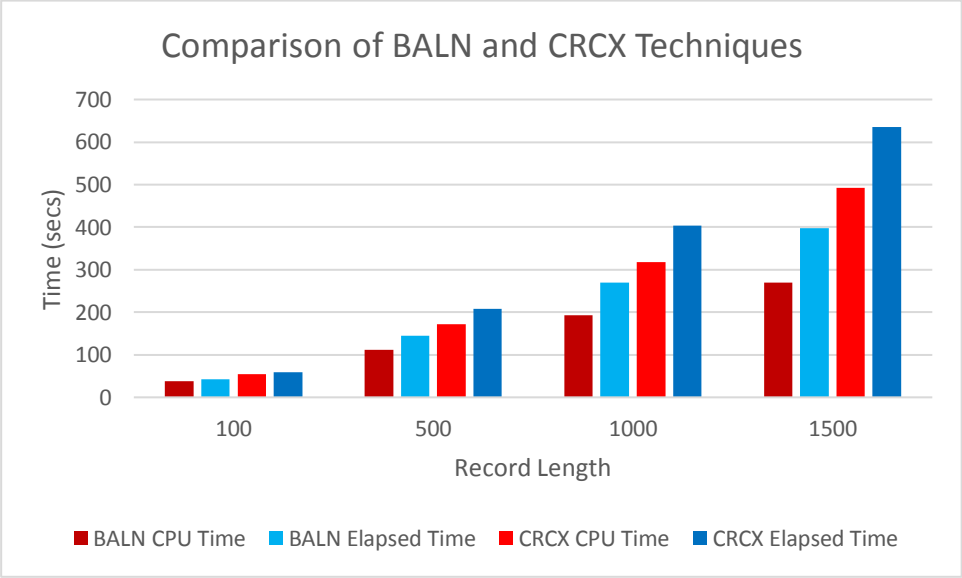


Figure 26 Comparison of BALN and CRCX Sequencing Techniques

Benchmarking has shown that while the CRCX sequencing technique has the capacity to sort a considerably larger number of records as it can use up to 17 intermediate work data sets it is not as efficient as the BALN technique when either sequencing technique can be used.

Figure 24 shows a comparison of the EXCP counts used for the sorting operations shown in Figure 23.

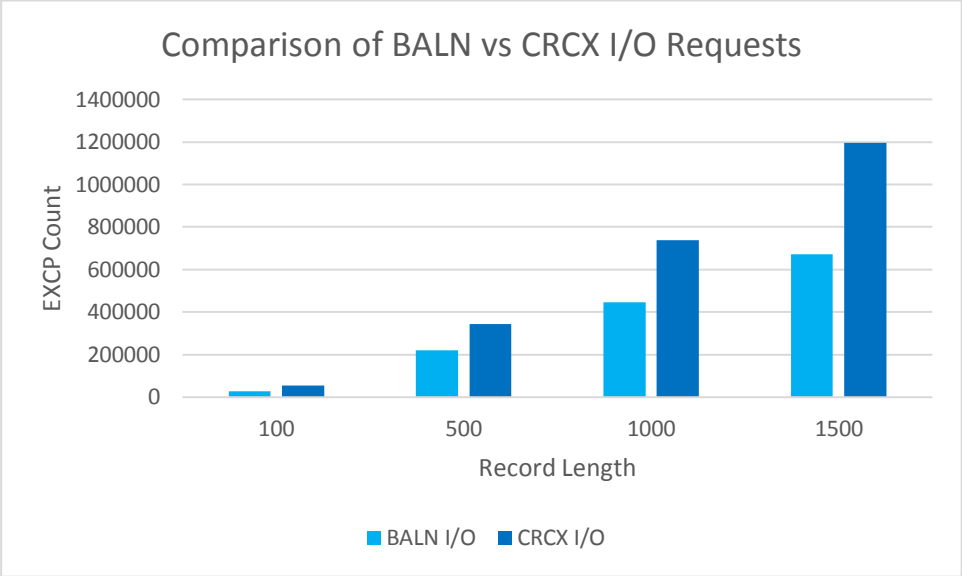


Figure 27 Comparison of BALN vs CRCX I/O Requests

The additional EXCP requests used by the CRCX sequencing technique to implement its algorithm result in a less efficient sort compared to the BALN technique for sorting the same number of records.

Wherever possible, the BALN sequencing technique should be used unless the increased capacity of the CRCX sequencing technique requires its use.

Length of records being sorted

When DASD is selected for intermediate storage the Sort/Merge Program will use either the BALN or the CRCX sequencing technique. Both of these sequencing techniques demonstrate a linear relationship between the CPU time used, the elapsed time and the length of the records being sorted. No “elbow” effect was observed, as shown in Figure 25.

Length of records being sorted

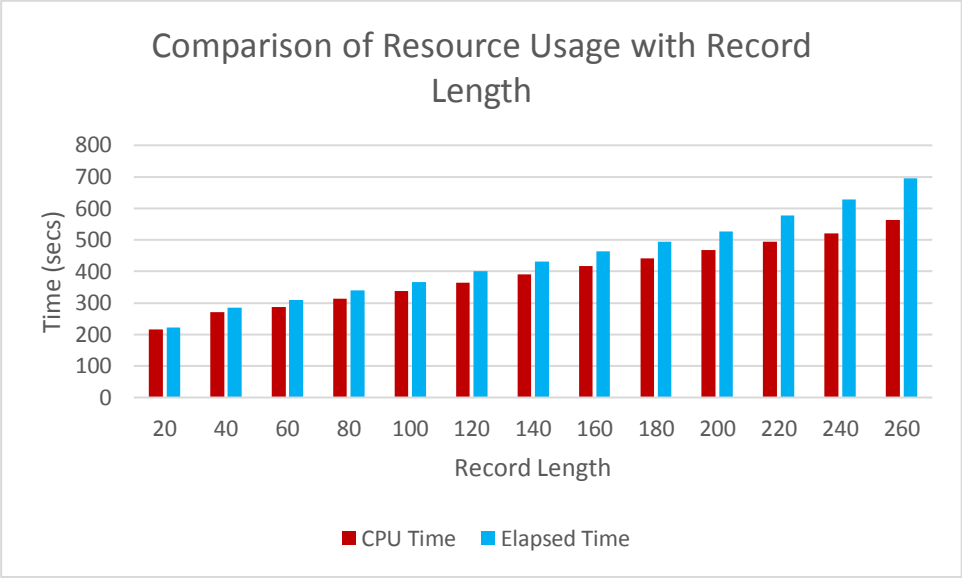


Figure 28 Comparison of Resource Usage with Record Length

Benchmark runs with longer length records were also performed. Again, the linear relationship between resource usage and record length was maintained. This can be seen in Figure 23.

Both the BALN and CRCX sequencing techniques demonstrate predicable resource usage with increasing record length and are not susceptible to non-linear increases in resource usage as the record length increases.

Use of compressed or non-compressed Hercules DASD

With 3390 DASD being the recommended DASD unit type for Sort/Merge Program intermediate storage then consideration has to be given on how to provide a number of intermediate storage 3390 DASD volumes for sorting purposes. Previous versions of the Sort/Merge Program were restricted to the use of 2314 DASD. Providing six volumes of uncompressed 2314 DASD occupied approximately 180 MB of PC disk space, a relatively small amount given the capacity of PC disk drives. An uncompressed 3390 DASD volume requires approximately 1 GB of PC disk drive space. As the Sort/Merge Program can now use up the 17 3390 DASD volumes for intermediate storage for large sorts this would require the allocation of approximately 17 GB of PC disk space. This would considerable increase the space used and time taken to back up the PC files used for Hercules emulated DASD. Consideration therefore has to be given to the use of Hercules compressed DASD for the 3390 DASD volumes used for intermediate storage.

A number of benchmarks have been run using compressed DASD volumes and uncompressed DASD volumes. Sorts using record lengths ranging from 18 bytes up to 27,900 bytes in length have been run. The number of records being sorted have ranged from requiring less than one volume of 3390 DASD up to the maximum of requiring 17 volumes of 3390 DASD.

The benchmark results have shown that use of compressed DASD compared to non-compressed DASD has little or no effect on sorting performance. The caveat to this statement is that the PC processor used for running the Hercules Emulator must be a multi core processor with a processor speed of at least 2 GHz.

There are two reasons why the overhead of compressing and decompressing blocks of data being transferred to and from compressed DASD does not impact the performance of the Sort/Merge Program. The first reason is because of the DASD I/O emulation design in the Hercules Emulator. Each I/O request is scheduled with a thread separate from the thread running the code responsible for emulating the 370 processor instructions. With a multi core PC processor the PC operating system dispatches one of the other available cores to run the compression/decompression process and carry out the physical I/O operation while the emulation of 370 instructions proceeds without being impacted. The second reason is that the design of Sort/Merge Program provides for a high degree of overlap between I/O operations and CPU processing. This is achieved by double buffering each of the intermediate storage data sets. While one buffer assigned to an intermediate storage data set is being used for an I/O transfer the other buffer is available for use in processing the records it contains or moving records into the buffer. The combination of the Hercules Emulator design and the design of the Sort/Merge Program together provide for the use of compressed DASD with little or no effect on performance.

Use of compressed DASD will place an additional load upon the PC processor as multiple threads, using the available cores, will be active running the Hercules Emulation thread and the threads for the I/O operations it schedules. Other applications running on the same PC can be impacted depending on the number of cores available and their processor requirements.

The use of compressed DASD is recommended due to the significantly reduced requirements for PC disk drive space while having little or no effect on sorting performance.

PC Configuration

As the Sort/Merge Program uses a significant amount of CPU processing and issues a large number of I/O requests to write and read intermediate storage data sets the performance and configuration of the PC hosting the Hercules Emulation has a major impact on performance.

Figure 26 compares the resource usage of a series of sorting operations run on an Intel Dual Core E4300 processor system and again run on an AMD FX-8320 processor system. The same number of records were sorted each time with a range of record lengths. Both systems used a HDD PC disk drive.

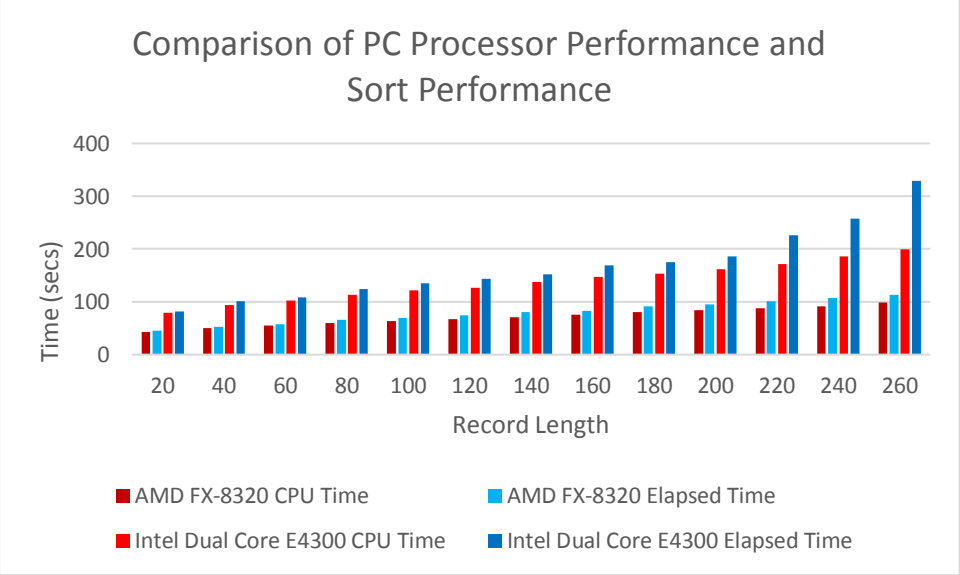


Figure 29 Comparison of PC Processor Performance and Sort Performance

The benchmark results from the Sort/Merge Program match published performance data resulting from bench marking the performance of the respective processors in other environments. A faster PC processor will result in improved sorting performance.

Figure 26 also shows the high degree of overlap between I/O operations and CPU processing. This is due, as discussed in the section on compressed or non-compressed DASD, to the design of the Hercules Emulator and the design of the Sort/Merge Program together providing maximum overlap of I/O operations and CPU processing.

The implication of the data shown in Figure 26 is that using an SSD instead of a HDD for Sort/Merge intermediate data storage will not provide a significant improvement in performance unless the PC processor has limited processing speed and is unable to drive I/O processing at optimum speed. The elapsed time using an SSD would, in all probability, be very close to or match the CPU time taken by a sorting operation. However due the large number of I/O requests with large data blocks issued by the Sort/Merge Program the life span of an SSD would be impacted if used regularly for sorting.

Appendix A. Sort/Merge Diagnostic Messages

Diagnostic messages are only generated by the Sort/Merge Program when it is running in diagnostic mode. Diagnostic mode can be activated by placing a SORTDIAG DD statement in the input job stream for the job step. Alternatively, diagnostic mode can be activated by coding the DIAGSIM parameter on a DEBUG control statement. In this case all diagnostic messages will be written to the SYSOUT message data set.

The same message can be generated by a number of different modules. Every module that can generate a particular message is listed. The selection of the type of intermediate storage, sequencing technique, number of control fields, record format and use of exits can impact the Sort/Merge Program's selection of the appropriate module to perform a specific function. When multiple modules are listed then one of the modules would be selected to perform the required function for each sorting or merging operation depending on the configuration.

The messages are listed and explained in numerical order, from IER073 to IER989. All the messages are informational except for message IER073 which is issued prior to the termination of a sorting operation.

IER073A DYNALLOC Error,ALLOC,SORTWK01,RC=0004,S99ERROR=0218,S99INFO=0000, Request to allocate 57144 tracks failed

Module	IERRCI
Explanation	Critical. The dynamic allocation SVC 99 returned with a non-zero return code. The S99ERROR and the S99INFO fields are formatted in the message. When diagnostic mode is active then the message will be followed by message IER073A DYNALLOC Error – SVC 99 PARAMETER LIST AREA. This message is the heading for a print dump of the dynamic allocation SVC 99 parameter list area. Figure 27 shows an example of a print dump of the SVC 99 parameter list.

```
 SYS17069.T124842.RA000.T1.JOB08029 ------------------------------------------------------------------------------ Line 759 Col 2 133
 Command ===>                                                     Scroll ===> CS
       10        20        30        40        50        60        70        80        90       100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER073A DYNALLOC Error,ALLOC,SORTWK01,RC=0004,S99ERROR=0218,S99INFO=0000,Request to allocate 57144 tracks failed
JOB T1IVPT           STEP EXECIVP1        TIME 124832   DATE 17069    ID = 001    CPUID = FD0001273033       PAGE 0001
-STORAGE
IER073A DYNALLOC ERROR – SVC 99 PARAMETER LIST AREA
0B46A0                                                800B46B8 14012000 02180000   *                    ~~~~~~~~~~~~*
0B46C0   000B46CC 00000000 00000000 000B46FC    000B470A 000B471A 000B4724 000B472E   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0B46E0   000B4737 000B473E 80000000 00000000    00000000 00000000 00000000 00010001   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0B4700   0008E2D6 D9E3E6D2 F0F10002 0001000A    5050E2D6 D9E3E6D2 F0F10015 00010004   *~~SORTWK01~~~~~~&&SORTWK01~~~~~~*
0B4720   F3F3F9F0 00070000 000A0001 0003000A    00010003 0265D800 04000100 01040005   *3390~~~~~~~~~~~~~~~~~~Q~~~~~~~~~*
0B4740   00010001 04000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0B4760   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
      LINE 0B4780 SAME AS ABOVE
0B47A0   00000000 00000000 00000000 00000000    00000000                              *~~~~~~~~~~~~~~~~~~~~            *
```

Figure 30 Example print dump of DYNALLOC SVC 99 Parameter List area

IER900I Initial Diagnostic Options –xxxxxxxxxxxxxxx

Module	IERRCM
Explanation	Informational. This message is generated as the heading for the SORTDIAG message data set. Usually there is nothing listed after the heading of Initial Diagnostic Options. However, if the installation configuration options were changed and regenerated to activate some of the diagnostic options then they would be listed here.

IER901I Input Buffer Table Addr –xxxxxx

Module	IERAPG, IERAPL
Explanation	Informational. The address provided, from the PPILAB02 field, is the address of the input buffer table which may have one or two entries depending on storage availability.

IER902I Output Buffer Addr –xxxxxx, xxxxxx

Module	IERAPA, IERAPB, IERAPN, IER9PA
Explanation	Informational. The addresses provided, from the PPILAB04 and PPILAB05 fields, are the addresses of the output buffers used in the final merge phase.

IER903I RSA Table Addr –xxxxxx

Module	IERAPG, IERAPL
Explanation	Informational. The Record Storage Area table address from the PPILAB08 field.

IER904I Tree Addr from xxxxxx to xxxxxx

Module	IERAOA, IERAOB, IERAOC, IERAOD, IERAOE, IERAOF, IERAOG, IERAOH
Explanation	Informational. The start and end addresses of the storage used for the Tree area.

IER905I Move Routine Addr –xxxxxx

Module	IERABF, IERABS
Explanation	Informational. The address of the routine used to move records internally within the sort during Phase 1.

IER906I DCB Table Addr –xxxxxx

Module	IERAGA, IERAGI, IERAGN
Explanation	Informational. The address of the table containing the DCB addresses from the field PPISTDCB.

IER907I Output CCW Addr –xxxxxx

Module	IERAPA, IERAPB, IERAPN
Explanation	Informational. The address of the CCW string used for writing records to intermediate storage.

IER908I Output IOB Addr –xxxxxx

Module	IERAPA, IERAPB, IERAPN, IER9PA
Explanation	Informational. The address of the IOB used for writing records to intermediate storage.

IER909I OPEN List Addr –xxxxxx

Module	IERAPA, IER9PA
Explanation	Informational. The address of a list of DCBs that are to be opened for this phase.

IER910I Generated Storage End Addr –xxxxxx

Module	IERAPG, IERAPL
Explanation	Informational. End address or high order address of working storage for Phase 1.

IER911I Getmain – Out Buffer L=xxxxxx, A=xxxxxx
In Buffer
Gen Area

Module	IERAPG
Explanation	Informational. Storage addresses for the input buffer, output buffer and working storage.

IER920I Generated Storage End Addr – xxxxxx

Module	IERAPH
Explanation	Informational. End address or high order address of working storage for Phase 2.

IER921I Sort Buffer Table Addr –xxxxxx

Module	IERAPH, IERAPL
Explanation	Informational. Address of table containing the addresses of the I/O buffers used for Phase 3.

IER922I Output Buffer Addr –xxxxxx

Module	IERAPD, IERAPE, IERAPO
Explanation	Informational. Address of the output buffer from field PPILAB04.

IER923I Move Routine Addr –xxxxxx

Module	IERABR
Explanation	Informational. The address of the routine used to move records internally within the sort during Phase 2.

IER924I DCB Table Addr –xxxxxx

Module	IERAGG, IERAGJ
Explanation	Informational. The address of the table containing the DCB addresses from the field PPISTDCB.

IER925I Output CCW Addr –xxxxxx

Module	IERAPD, IERAPE, IERAPO
Explanation	Informational. The address of the CCW string used for writing records to intermediate storage during Phase 2.

IER926I IOB Table Addr –xxxxxx

Module	IERAPD, IERAPE, IERAPO
Explanation	Informational. The address of the IOB table containing the IOB addresses used for writing records to intermediate storage during Phase 2.

IER927I Input CCW Addr –xxxxxx

Module	IERAGB, IERAGC, IERAGL, IERAGO, IER9GB
Explanation	Informational. The address of the CCW string used for reading records during the intermediate phase.

IER930 REQ/REL TRK xxxxxxxx

Module	IER8ON
Explanation	Informational. When using the CRCX sequencing technique the Sort/Merge Program manages the intermediate storage DASD space by means of track groups. Track groups are requested, used for the storage of records, and then released when the records have been merged into longer sequence strings. The first two bytes of the data are the offset (in hex) into the DCB table to identify the relevant DCB of the SORTWKdd data set. The last six bytes are either the TTR (in hex) of the DASD address provided in response to a request for a track group from the pool of free track groups or the TTR of a track group that has been released back to the pool of track groups. Due to the large number of requests and releases of track groups that occur during a sizeable CRCX sorting operation message IER930 must be enabled by zapping the TRACKT subroutine located in the IER8ON load module.

IER931I EOF ON SORTIN

Module	IERRGA
Explanation	Informational. The end of input routine for the initial sort phase has been entered. This can be as a result of encountering the end of the file of the SORTIN input data set or by the limitation on records input to the sort by the STOPAFT parameter.

IER940I Generated Storage End Addr –xxxxxx

Module	IERAPI
Explanation	Informational. End address or high order address of working storage for Phase 3.

IER941I Input Buffer Table Addr –xxxxxx

Module	IERAPI
Explanation	Informational. The address provided, from the PPILAB02 field, is the address of the input buffer table. This message is only issued for merge only operations.

IER942I Output Buffer Addr –xxxxxx

Module	IERABL, IERABM, IERABN, IERABO, IERABP, IER9BN, IER9BO
Explanation	Informational. Address of the output buffer, from field PPILAB04, during Phase 3.

IER943I Move Routine Addr –xxxxxx

Module	IERABQ
Explanation	Informational. The address of the routine used to move records internally within the sort during Phase 3.

IER944I DCB Table Addr –xxxxxx

Module	IERAGK, IERAPF, IERAPK
Explanation	Informational. The address of the table containing the DCB addresses from the field PPISTDCB during Phase 3.

IER945I Input CCW Addr –xxxxxx

Module	IERAGD, IERAGE, IERAGM, IERAGP, IER9GC
Explanation	Informational. The address of the CCW string used for reading records during the final merge phase.

IER961I Sort Technique –BALN/OSCL/POLY/CRCX

Module	IERBGA, IERBGB, IERRCK
Explanation	Informational. One of the four possible different sequencing techniques will appear in the message identifying the sequence technique selected for this sorting operation.

IER962I Phase x Number of Buffers = xxx, Buffer Size =xxxxxx

Module	IERBGA, IERBGB, IERRCK
Explanation	Informational. The phase about to run, the number of I/O buffers allocated for the phase and the length of the I/O buffers are identified in this message. This message will be repeated for each of the three phases.

IER963I Storage = xxxxxx

Module	IERBGA, IERBGB, IERRCK
Explanation	Informational. The data in the message is the total amount of storage, in bytes, available to the Sort/Merge program for this sorting run.

IER964I Phase x Storage =xxxxxx

Module	IERBGA, IERBGB, IERRCK
Explanation	Informational. The data in the message is the amount of storage, in bytes, available to the identified phase for this sorting run This message will be repeated for each of the three phases.

IER965I Merge Order = xxxx

Module	IERBGA, IERBGB, IERRCK
Explanation	Informational. The data in the message identifies the degree of complexity for the merge phase.

IER980I Calling IERxxx {Return Code = xxxx}

Module	IERRCM, IERRCZ, IERRC9
Explanation	Informational. The Sort/Merge Program will call the identified module. The module called is typically a definition phase module or an assignment phase module to initialize a running module. If the return code is non zero then the additional Return Code part of the message is included in the message with the return code value.

IER981I ddddddddd,uuu,xxxx,vvvvvv ccccc hh – ccccc hh,Tracks = ttttt

Module	IERRC4
Explanation	Informational. Module IERRC4 gathers system information for the Sort/Merge Program's definition phase. This message provides information for each of the SORTWKdd data sets that will be used for this sorting operation.

ddddddddd	DD name
uuu	UCB address
xxxx	DASD unit type
vvvvvv	Volume serial number
cccc hh – cccc hh	Start address of DASD extent – End address of DASD extent If the dataset is allocated in extents then the cccc hh – cccc hh line will be repeated for each extent
ttttt	number of tracks in the extent

IER982I CPI post IERxxx Processing

Module	IERRCM
Explanation	Informational. When the SNAPCPI DEBUG parameter is active then after each definition phase module has been invoked and control has been returned to IERRCM the CPI control block is print dumped. The CPI control block is mapped by DSECT IERRC5 which is generated by the macro SMCPI. A listing of the CPI is generated by assembly of the module IERRC1. Figure 28 shows an example of a print dump of the CPI.

```
 SYS17072.T131008.RA000.T1.JOB08039 ------------------------------------------------------------------------------ Line 130 Col 2 133
 Command ===>                                                Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER982I CPI post IERRCY Processing
0A4DC0                                          00000000 000A4FA8 00000000 500A924C   *                  ~~~~~~~|y~~~~~&~k<*
0A4DE0   12C8C0FC 500A924C 00C68010 00000000    000A99EC 000A99EC 0000003B 000A7C58   *~H{~&~k<~F~~~~~~~~~r~~~~r~~~~~~~~@~*
0A4E00   009C4160 000A9168 00000046 00C68010    500A924C 00C68010 C3D7C9C1 40404040   *~~~~-~~j~~~~~~~F~~&~k<~F~~CPIA    *
0A4E20   C2F0F0F5 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *B005~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4E40   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4E60   00000000 00000000 00000000 00000000    00005000 00004982 27BC0000 00000000   *~~~~~~~~~~~~~~~~~~~~&~~~~b~~~~~~~~~*
0A4E80   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
      LINE 0A4EA0 SAME AS ABOVE
0A4EC0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 E2D6D9E3   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~SORT*
0A4EE0   F3F3F9F0 40404040 0007000A 00000032    00000000 00200000 00040000 00010000   *3390    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4F00   00018000 00080000 000A7A2C E2E8E2D6    E4E34040 C9C5D9D7 C1D9D440 006D0011   *~~~~~~~~~~~:~SYSOUT  IERPARM ~_~~*
0A4F20   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4F40   00000000 00000000 00000000 0000179C    000B2860 000B292E 000B2860 000B28A5   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4F60   000B28A6 000B28EB 00000000 00000000    00000000 00000000 000B28EC 000B292D   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4F80   00000000 000B3860 000B405F 00000000    02000020 E3F1C9E5 D7F24040 6BC9E5D7   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A4FA0   F2404040 40400000                                                            *2     ~~                          *
```

Figure 31 Example print dump of CPI

IER983I EXCP Issued by IERxxx+yyy

Module	IERDTE
Explanation	Informational. When the TRACE=EXCPREQ DEBUG parameter is active then the message IER983 and its associated data is generated for every EXCP I/O request made to the intermediate storage SORTWKdd data sets. The message IER983I identifies the Sort/Merge Program module and the offset location within the module that issued the EXCP request. For channel programs that write data, the first 256 bytes of the I/O area are provided in the trace together with a fully formatted IOB. The CCW chain used for the I/O operation is also formatted. No data is formatted for channel programs that read data. Note that for a sorting operation with a large number of records, resulting in many I/O operations to the intermediate storage SORTWKdd data sets, a considerable number of lines of diagnostic data will be generated. The data being read or written for the CRCX technique differs from the data being read or written for the BALN sequencing technique. The data block shown in Figure 29 is trace output from a sorting operation using the CRCX technique. The first eight bytes of the data block contain the address of the next block of data in the sequence set. The first byte is the offset into the DCB table for the data set with an offset of four being the offset for SORTWK01, eight for SORTWK02 and so on. The last three bytes are the relative TTR address of the next block of the sequence set which in this example is the next track in the data set. The next four bytes are used as a sequence indicator. The characters HHHH signify that this is the first block or a middle block in this particular sequence set. The last block in the sequence set has the character value of HGHH signaling the end of the sequence set. The data blocks shown in Figures 30 and 31 are trace output from a sorting operation using the BALN technique. The major differences between the BALN technique and the CRCX technique are firstly the way that the blocks in a particular sequence set are chained and secondly the management of free space in the intermediate work data sets. Figure 30 shows the last data block in a sequence set. The sequence indicator, in the first four bytes of the record, is set to indicate the end of sequence with the character string HGGH. Note that with the BALN technique there is no chaining in the data block in itself to the next data block. BALN sequence sets are contiguous until terminated with an end of sequence set indication in the last data block. Figure 29 shows a BALN sequence set directory block with the addresses of eight different sequence sets. The format of each eight byte address in a directory block is the same as the format used for the CRCX technique. The first byte is the offset into the DCB table for the data set with an offset of four being the offset for SORTWK01, eight for SORTWK02 and so on. The last three bytes are the relative TTR address of the location of the sequence set.

```
 SYS17072.T131008.RA000.T1.JOB08039 ------------------------------------------------------------------------------ Line 725 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER983I EXCP Issued by IER8PA+025E
IOB 0AD0F0
    -08 IERECB   00000000  -04 IERALTCW   000ACF90
    +00 IOBFLAG1 40 IOBCMDCH              +01 IOBFLAG2 00 +02 IOBSENS0 00 +03 IOBSENS1 00
    +04 IOBECBCC 00 +05 IOBECBPB 0AD0E8   +08 IOBFLAG3 00 +09 IOBCMDA  000000
    +12 IOBUSTAT 00                       +13 IOBCSTAT 00                   +14 RESIDUAL     0
    +16 IOBSIOCC 00 +17 IOBSTRTB 0ACF78   +20 IOBFLAG4 00 +21 IOBDCBPB 0AD178    DDNAME SORTWK01
    +24 IOBRESTR 00000000         +28
    +32 IOBSEEK  +36 IOBCC   001  +38 IOBHH 00     +40 IOBR  00
CCW 0ACF78  +00 31 SEARIDEQ       +01 DATA  0AD113 +04 FLAGS 60 CC+SLI         +06 COUNT     5
                      CC   001         HH 00            R  00
CCW 0ACF80  +00 08 TIC            +01 DATA  0ACF78 +04 FLAGS 60 CC+SLI         +06 COUNT     0
CCW 0ACF88  +00 1D WRITECKD       +01 DATA  0B63C8 +04 FLAGS 20 SLI            +06 COUNT  7232
                      CC   001         HH 00            R  01           KL   00  DL     7224
    0B63D0 04040000 00000101 C8C8C8C8 5C5C5C5C 5CF0F0F0 F0F6F45C 5C5C5C5C 5C5C5C5C *^^^^^^^^HHHH*****000064**********
    0B63F0 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C40 40C6F5F0 F0D2E5D8 D35C5C5C *********************  F500KVQL****
    0B6410 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C F0F0F0F1 F2F5F0F1 5C5C5C5C **********************00012501****
    0B6430 5CF0F0F0 F0F0F15C 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C **000001**************************
    0B6450 5C5C5C40 C1D8C1F0 F0F7C3C3 C45C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C **** AQA007CCD********************
    0B6470 5C5C5C5C F0F0F0F0 F4F2F0F1 5C5C5C5C 5CF0F0F0 F0F1F05C 5C5C5C5C 5C5C5C5C *****00004201*****000010**********
    0B6490 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5CC1 C5E8D4F0 F0E5D1F4 F85C5C5C **********************AEYM00VJ48****
    0B64B0 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C F0F0F0F0 F5F1F0F1 5C5C5C5C **********************00005101*****
```

TTR addr of next block

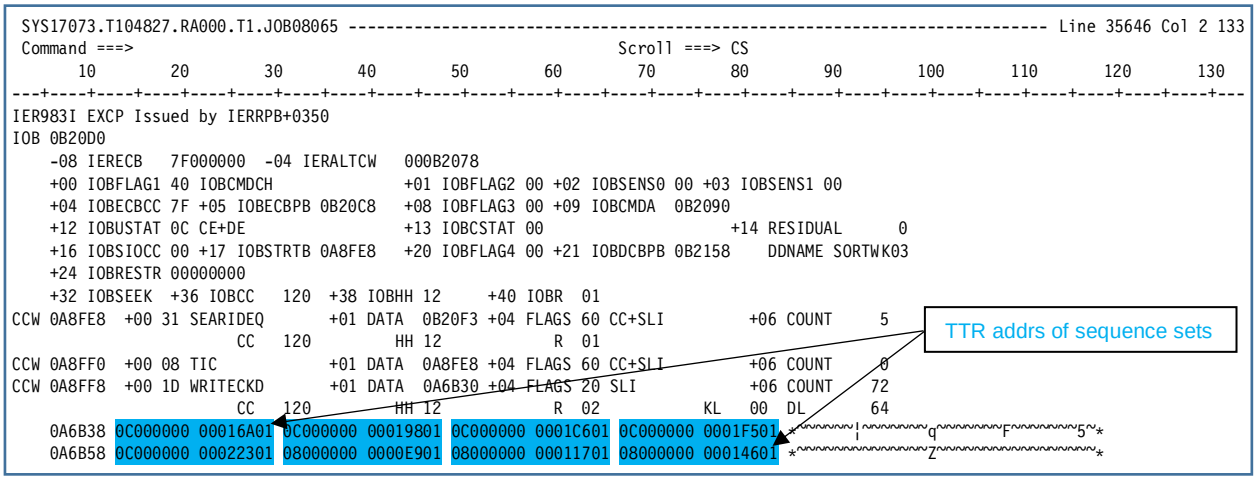
Sequence indicator

Figure 32 Example EXCPREQ data for CRCX sequencing technique

```
 SYS17073.T104827.RA000.T1.JOB08065 ------------------------------------------------------------------------------ Line 35610 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER983I EXCP Issued by IERRPB+0190
IOB 0B20D0
    -08 IERECB   7F000000  -04 IERALTCW   000B2078
    +00 IOBFLAG1 40 IOBCMDCH              +01 IOBFLAG2 00 +02 IOBSENS0 00 +03 IOBSENS1 00
    +04 IOBECBCC 7F +05 IOBECBPB 0B20C8   +08 IOBFLAG3 00 +09 IOBCMDA  0B20A8
    +12 IOBUSTAT 0C CE+DE                 +13 IOBCSTAT 00                   +14 RESIDUAL      0
    +16 IOBSIOCC 00 +17 IOBSTRTB 0B2078   +20 IOBFLAG4 00 +21 IOBDCBPB 0B2158    DDNAME SORTWK03
    +24 IOBRESTR 00000000
    +32 IOBSEEK  +36 IOBCC   120  +38 IOBHH 10     +40 IOBR  00
CCW 0B2078  +00 31 SEARIDEQ       +01 DATA  0B20F3 +04 FLAGS 60 CC+SLI         +06 COUNT     5
                      CC   120         HH 10            R  00
CCW 0B2080  +00 08 TIC            +01 DATA  0B2078 +04 FLAGS 60 CC+SLI         +06 COUNT     0
CCW 0B2088  +00 1D WRITECKD       +01 DATA  12B5E0 +04 FLAGS 20 SLI            +06 COUNT 18976
                      CC   120         HH 10            R  01           KL   00  DL    18968
    12B5E8 C8C7C7C8 E7E7E7E7 F0F0F0F4 F3F9F3F0 F4F4E7F0 F0F0F0F2 F2F6F3F5 F6E7E7E7 *HGGHXXXX0004393044X0000226356XXX*
    12B608 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 *XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
    12B628 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 F0F0F0F4 F3F9F2F6 *XXXXXXXXXXXXXXXXXXXXXXXX00043926*
    12B648 F8F1E7F0 F0F0F0F2 F2F1F6F8 F8E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 *81X0000221688XXXXXXXXXXXXXXXXXX*
    12B668 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 *XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
    12B688 E7E7E7E7 E7E7E7E7 F0F0F0F4 F3F9F2F2 F2F5E7F0 F0F0F0F2 F2F0F2F8 F4E7E7E7 *XXXXXXXX0004392225X0000220284XXX*
    12B6A8 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 *XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
    12B6C8 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 F0F0F0F4 F3F9F0F0 *XXXXXXXXXXXXXXXXXXXXXXXX00043900*
```

Sequence indicator

Figure 33 Example EXCPREQ data for BALN sequencing technique




```
 SYS17073.T104827.RA000.T1.JOB08065 ------------------------------------------------------------------------------ Line 35646 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER983I EXCP Issued by IERRPB+0350
IOB 0B20D0
    -08 IERECB   7F000000  -04 IERALTCW   000B2078
    +00 IOBFLAG1 40 IOBCMDCH              +01 IOBFLAG2 00 +02 IOBSENS0 00 +03 IOBSENS1 00
    +04 IOBECBCC 7F +05 IOBECBPB 0B20C8   +08 IOBFLAG3 00 +09 IOBCMDA  0B2090
    +12 IOBUSTAT 0C CE+DE                 +13 IOBCSTAT 00                   +14 RESIDUAL      0
    +16 IOBSIOCC 00 +17 IOBSTRTB 0A8FE8   +20 IOBFLAG4 00 +21 IOBDCBPB 0B2158    DDNAME SORTWK03
    +24 IOBRESTR 00000000
    +32 IOBSEEK  +36 IOBCC   120  +38 IOBHH 12     +40 IOBR  01
CCW 0A8FE8  +00 31 SEARIDEQ       +01 DATA  0B20F3 +04 FLAGS 60 CC+SLI         +06 COUNT     5
                      CC   120         HH 12            R  01
CCW 0A8FF0  +00 08 TIC            +01 DATA  0A8FE8 +04 FLAGS 60 CC+SLI         +06 COUNT     0
CCW 0A8FF8  +00 1D WRITECKD       +01 DATA  0A6B30 +04 FLAGS 20 SLI            +06 COUNT    72
                      CC   120         HH 12            R  02          KL   00  DL          64
    0A6B38 0C000000 00016A01 0C000000 00019801 0C000000 0001C601 0C000000 0001F501 *^^^^^^^^j^^^^^^^^q^^^^^^^^F^^^^^^^^5^*
    0A6B58 0C000000 00022301 08000000 0000E901 08000000 00011701 08000000 00014601 *^^^^^^^^^^^^^^^^Z^^^^^^^^^^^^^^^^^^^^*
```

Figure 34 Example EXCPREQ data directory block for BALN sequencing technique

IER984I WAIT on EXCP Issued by IERxxx+yyy

Module	IERDTE
Explanation	Informational. When the TRACE=EXCPCOMP DEBUG parameter is active then all WAIT requests for I/O completion to the intermediate storage SORTWKdd data sets generate the message IER984 and its associated data. The trace data is generated after the WAIT for the I/O event has been posted complete. The message IER984 identifies the Sort/Merge Program module and the offset location within the module that issued the WAIT for I/O request completion. For channel programs that read data, the first 256 bytes of the I/O area are provided in the trace together with a fully formatted IOB. The CCW chain used for the I/O operation is also formatted. No data is formatted for channel programs that write data. Note that for a sorting operation with a large number of records, resulting in many I/O operations to the intermediate storage SORTWKdd data sets, a considerable number of lines of diagnostic data will be generated. The data being read or written for the CRCX sequencing technique differs from the data being read or written for the BALN sequencing technique. The data block shown in Figure 32 is a trace record from a sorting operation using the BALN technique. Refer to the explanation provided for message IER983 for a description of the differences in the data format between the BALN sequencing technique and the CRCX sequencing technique.

```
 SYS17073.T104827.RA000.T1.JOB08065 ------------------------------------------------------------------------------ Line 39924 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER984I WAIT on EXCP Issued by IERRGC+03A6
IOB 0A4768
    -08 IERECB   7F000000  -04 IERALTCW   020A45B8
    +00 IOBFLAG1 40 IOBCMDCH              +01 IOBFLAG2 00 +02 IOBSENS0 00 +03 IOBSENS1 00
    +04 IOBECBCC 7F +05 IOBECBPB 0A4760   +08 IOBFLAG3 00 +09 IOBCMDA  0A45B8
    +12 IOBUSTAT 0C CE+DE                 +13 IOBCSTAT 00                   +14 RESIDUAL      0
    +16 IOBSIOCC 40 +17 IOBSTRTB 0A45A0   +20 IOBFLAG4 00 +21 IOBDCBPB 0A48F0    DDNAME SORTWK02
    +24 IOBRESTR 00000000
    +32 IOBSEEK  +36 IOBCC   036  +38 IOBHH 20     +40 IOBR  01
CCW 0A45A0  +00 31 SEARIDEQ       +01 DATA  0A478B +04 FLAGS 60 CC+SLI         +06 COUNT     5
                      CC   036         HH 20            R  01
CCW 0A45A8  +00 08 TIC            +01 DATA  0A45A0 +04 FLAGS 60 CC+SLI         +06 COUNT     0
CCW 0A45B0  +00 06 READDATA       +01 DATA  0EE5E8 +04 FLAGS 20 SLI            +06 COUNT 18968
    0EE5E8 C8C8C8C8 E7E7E7E7 F0F0F1F5 F9F9F9F6 F3F7E7F0 F0F0F0F0 F8F7F8F7 F0E7E7E7 *HHHHXXXX0015999637X0000087870XXX*
    0EE608 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 *XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
    0EE628 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 F0F0F1F5 F9F9F7F2 *XXXXXXXXXXXXXXXXXXXXXXX00159972*
    0EE648 F7F3E7F0 F0F0F0F0 F8F2F8F1 F1E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 *73X0000082811XXXXXXXXXXXXXXXXXX*
    0EE668 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 *XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
    0EE688 E7E7E7E7 E7E7E7E7 F0F0F1F5 F9F9F6F9 F5F7E7F0 F0F0F0F0 F8F9F8F5 F4E7E7E7 *XXXXXXXX0015996957X0000089854XXX*
    0EE6A8 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 *XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
    0EE6C8 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 E7E7E7E7 F0F0F1F5 F9F9F3F9 *XXXXXXXXXXXXXXXXXXXXXXX00159939*
```

Figure 35 Example EXCPCOMP data for BALN sequencing technique

IER985I SORTWORK Device uuuu, Cyls per Vol =cccccc,Trks per Cyl =ttttt,TrkIen =lllll

Module	IERRCI
Explanation	Informational. When module IERRCI has determined the DASD unit type that has been selected for intermediate working storage for a sorting operation then the physical attributes of the selected unit type are provided in this message. The data provided by this message is extracted from the operating system IECZDTAB Device Characteristics Table entry for the device type. The value provided for the track length is the total number of bytes that can be written to the track and does not reflect the length of the largest block that can be written to the device with standard DASD track formatting. The largest data block that can be written is always than the track length.

uuuu	DASD unit type
cccccc	Number of cylinders per volume
ttttt	Number of tracks per cylinder
lllll	Track length

IER986I Cntl Stmt area post IERxxx Processing

Module	IERRCM
Explanation	Informational. When the SNAPCPI DEBUG parameter is active then after each definition phase module has been invoked and control has been returned to IERRCM the CPI Control Statement Analysis Area is print dumped. This message and the data provided will appear immediately after message IER982. Figure 33 shows an example of a print dump of the CPI Control statement processing area

```
 SYS17073.T121949.RA000.T1.JOB08062 ------------------------------------------------------------------------------ Line 202 Col 2 133
 Command ===>                                                Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER986I Cntl Stmt area post IERRCE processing
0B2860   E2D6D9E3 40C6C9C5 D3C4E27E 4DF4F06B    F1F06BC3 C86BC15D 40404040 40404040   *SORT FIELDS=(40,10,CH,A)        *
0B2880   40404040 40404040 40404040 40404040    40404040 40404040 40404040 40404040   *                                *
0B28A0   40404040 4040D9C5 C3D6D9C4 40E3E8D7    C57EC66B D3C5D5C7 E3C87E4D F8F05D40   *      RECORD TYPE=F,LENGTH=(80) *
0B28C0   40404040 40404040 40404040 40404040    40404040 40404040 40404040 40404040   *                                *
0B28E0   40404040 40404040 40404040 C4C5C2E4    C740E3D9 C1C3C57E 4DC5E7C3 D7D9C5D8   *            DEBUG TRACE=(EXCPREQ*
0B2900   6BC5E7C3 D7C3D6D4 D76BD4D6 C4C6D3D6    E65D6BE2 D5C1D7C3 D7C94040 40404040   *,EXCPCOMP,MODFLOW),SNAPCPI      *
0B2920   40404040 40404040 40404040 40400000    00000000 00000000 00000000 00000000   *              ^^^^^^^^^^^^^^^^^^^^*
0B2940   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
      LINES 0B2960-0B3840 SAME AS ABOVE
0B3860   01C6C9C5 D3C4E2FF FF04F4F0 FFFFFFFF    FFFFF1F0 FFFFFFFF FFFFC3C8 FFFFFFFF   *^FIELDS^^^^40^^^^^^^10^^^^^^^CH^^^^^*
0B3880   FFFFC1FF FFFFFFFF FFFF0000 00000000    00000000 00000000 00000000 00000000   *^^^A^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
0B38A0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
      LINES 0B38C0-0B3FC0 SAME AS ABOVE
0B3FE0   00000000 00000000 00000000 00000000    00000000 00000000 00000000            *^^^^^^^^^^^^^^^^^^^^^^^^^^^^^    *
```

Figure 36 Example CPI Control Statement Analysis Area

IER987I PPI –INITIALIZATION/AFTER IERxxx Processing

Module IERRCZ

Explanation Informational. When the SNAPPPI DEBUG parameter is active then after the PPI has been initialized from data in the CPI and then again after each definition phase module has been invoked and control has been returned to IERRCZ the PPI control block is print dumped. The PPI control block is mapped by DSECT IERRCA which is generated by the macro SMPPI. A full listing of the PPI is generated by assembly of the module IERRC1. Figure 34 shows an example print dump of the PPI after it has been initialized.

```
 SYS17073.T123406.RA000.T1.JOB08066 ------------------------------------------------------------------------------- Line 152 Col 2 133
 Command ===>                                                Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER987I PPI - INITIALIZATION
0A5AE0                     00000000 000A4FA8    00000000 00000000 00000000 00000000   *        ~~~~~~|y~~~~~~~~~~~~~~~~*
0A5B00   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A5B20   00000000 00000000 00000000 00000000    00009710 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~p~~~~~~~~~~~~~~*
0A5B40   00000001 00000004 0000000C 0000000C    0000000C 00000000 00010027 00090000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A5B60   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
      LINES 0A5B80-0A5CE0 SAME AS ABOVE
0A5D00   00000000 00000000 00000000 00000000    00000000 00000000 A1128002 52120208   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A5D20   498227BA 00000000 C2F0F0F5 00000000    00000000 00000000 00000000 00000000   *~b~~~~~~B005~~~~~~~~~~~~~~~~~~~~*
0A5D40   00000000 00000000 00000000 00050D30    00000000 00000054 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A5D60   00001511 005A0001 0E0E0000 00000000    00000000 00000000 00000000 00500050   *~~~~~~!~~~~~~~~~~~~~~~~~~~~~~~~&~&*
0A5D80   00500000 00000006 00060000 00270009    00000000 00000000 00000000 00070000   *~&~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
0A5DA0   00000050 00001C38 02020050 00009710    00000610 00000000 00000000 0007BA30   *~~~~&~~~~~~~~&~~p~~~~~~~~~~~~~~~~*
0A5DC0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
      LINES 0A5DE0-0A5EE0 SAME AS ABOVE
0A5F00   00000000 00000000 00000000 00000000    E2D6D9E3 00000000 000000F0 F0F00000   *~~~~~~~~~~~~~~~~~~~~SORT~~~~~~~~000~~*
0A5F20   0000F3F3 F9F04040 40400007 00000032    000AE2E8 E2D6E4E3 4040C9C5 D9D7C1D9   *~~3390    ~~~~~~~~~SYSOUT  IERPAR*
0A5F40   D440006D 00110200 0020E3F1 C9E5D7F2    40406BC9 E5D7F240 40404040 00000000   *M ~_~~~~~~T1IVP2  ,IVP2     ~~~~*
0A5F60   07F10700 58F0D490 07FF07F1 07001311    58F0D490 07FF0000 00000000 00000000   *~1~~~0M~~~~1~~~~~0M~~~~~~~~~~~~~~~~*
0A5F80   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
      LINE 0A5FA0 SAME AS ABOVE
0A5FC0   00000000 00000000 1C680001 0DB00002    08E80003 06880004 05200005 04300006   *~~~~~~~~~~~~~~~~~~~~Y~~~h~~~~~~~~~~*
0A5FE0   03880007 03080008 02A00009 0000000A    002C0000 00017B66 00017AE0            *~h~~~~~~~~~~~~~~~~~~~~~~#~~~:\    *
```

Figure 37 Example print dump of PPI after Initialization

IER988I IERxxx Loaded at xxxxxx { - ffff }

Module IERRCV, IERRC6, IERRC7, IERRC8, IERRC9

Explanation Informational. Message IER988 has two formats. The first format identifies the normal loading of a module. The second format identifies the loading of a module that will form part of the group of modules that will implement a phase of the selected sort or merge sequencing technique. The function provided by the loaded module is listed after the address of the module. As specific modules are loaded depending on different configurations the Sort/Merge Program uses the name of the function in calling modules to locate the address of the module selected for a particular function. Figure 35 shows an example of both formats of the message. Figure 36 shows an example of the Sort/Merge Program source code where control is passed to various modules directly by locating their address by function name. The relevant function names have been highlighted in both Figures to show the relationship between the source code and the modules selected for this particular sorting configuration.

```
SYS17073.T124311.RA000.T1.JOB08066 ------------------------------------------------------------------------------ Line 345 Col 2 133
 Command ===>                                                  Scroll ===> CS
       10        20        30        40        50        60        70        80        90        100       110       120       130
---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+---
IER988I IERRC6 Loaded at 0A60C0
IER989I Deleting IERRC6
IER988I IER8ON Loaded at 0A7190 - ALG
IER988I IERRDP Loaded at 0A5028 - DEB
IER988I IERROB Loaded at 0A6240 - NET
IER988I IERRBY Loaded at 0A70A0 - BLK
IER988I IER8PA Loaded at 0A8CC0 - WRT
IER988I IER8GB Loaded at 0A8708 - RD
IER988I IERRDL Loaded at 0A7020 - DEB2
IER988I IERROV Loaded at 0A81A0 - NETM
IER988I IERRBT Loaded at 0A8080 - BLK2
IER988I IERROX Loaded at 0A6160 - INT
IER988I IERRGA Loaded at 0AB038 - EOF
IER988I IERRMA Loaded at 0A6040 - RMA
IER988I IERRMB Loaded at 0A9F28 - RMB
IER988I IERAMA Loaded at 0A9CC0 - AMA
IER988I IERAMB Loaded at 0A9B18 - AMB
IER988I IERAP1 Loaded at 0A8028 - OPEN
IER988I IERAPL Loaded at 0A9330
IER980I Calling IERAPL
```

Figure 38 Example of both formats of message IER988

```
REVEDIT  SYSD.SORTMVS.ASM(IER8PAI) - 1.01                COLUMNS 00001 00072
COMMAND ===>                                                 SCROLL ===> CS
000114          L     R11,PPIBLK+4        SORT BLOCK BASE
000115          B     8(,R11)
000116 *
000117 PA025    L     R11,PPIBLK2+4       MERGE BLOCK BASE
000118          B     8(,R11)
000119 *
000120 PARTALG  L     R6,KDCB             R6 -> DCB
000121          SR    R1,R1
000122          IC    R1,DCBOPTCD         R1 = M INCREMENT STORED IN DCBOPTCD
000123          CLI   KSAVE+15,X'04' [R4] SET COND CODE FOR ENTRY TO ALG
000124          L     R11,PPIALG+4        SET ALGORITHM'S BASE FOR
000125          BE    8(,R11)             EOS - WRITE AT EOS (PH1 OR PH2)
000126          BR    R11                 EOF - WRITE AT EOF (SIM OR REAL)
```

Figure 39 Example of locating the address of a module by function name

IER989I Deleting IERxxx

Module	IERRCV, IERRC9
Explanation	Informational. Message IER989I identifies the name of a module being deleted. Figure 35 shows an example of the message.

Appendix B. Syntax

This document uses two kinds of description for the syntax of control statements and parameters. The descriptions are:

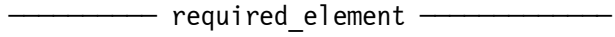
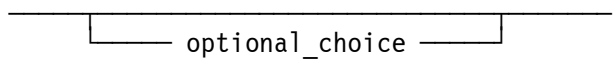
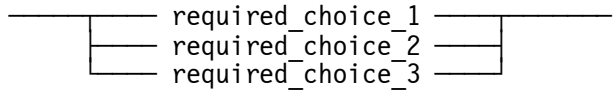
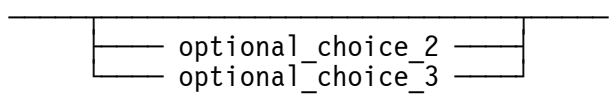
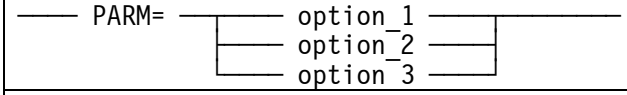
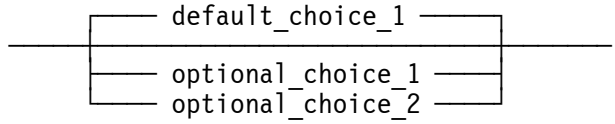
- Syntax descriptions
- Syntax diagrams

Reading Syntax Descriptions

Table 3 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.
	A required element (keyword or variable) appears on the main path of the horizontal line. This element must be specified
	An optional element (keyword or variable) appears below the main path of the horizontal line. This element may or may not be specified.
	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
	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.
	A keyword with options. Only one of the available options can be specified.
	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.

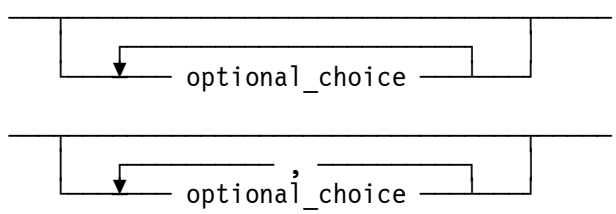
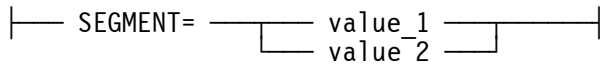
Symbol	Description
	An arrow returning to the left of an element below the main path of the horizontal line indicates an optional repeatable item. A character within the arrow path means that repeated items have to be separated by that character. If there is no character within the arrow path then the items are separated by a blank.
	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.
	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 40 Reading Syntax Diagrams

Sample Syntax Diagram

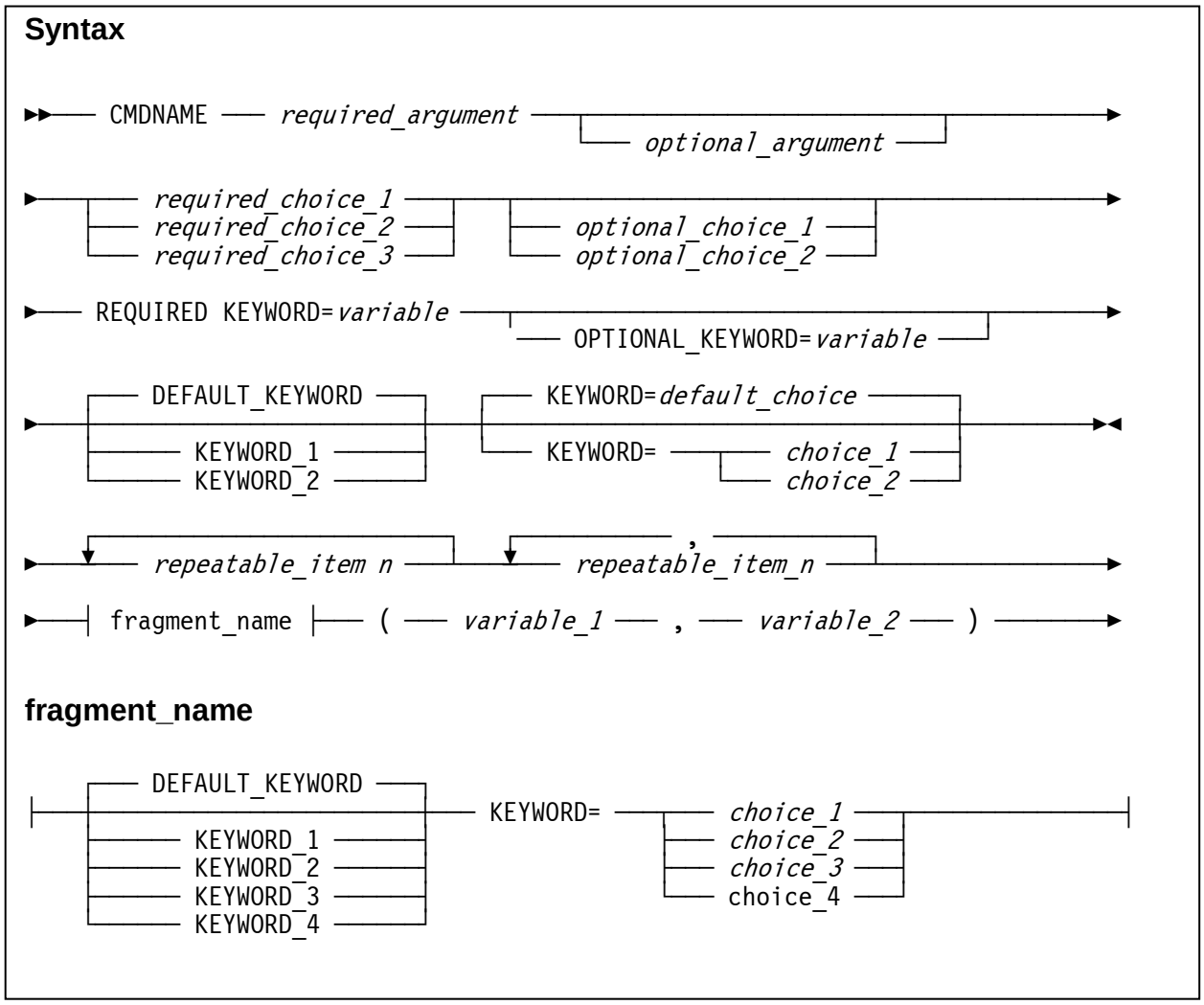


Figure 41 Sample Syntax Diagram

Index