SC33-4044-2

DOS/VS Sort/Merge Version 2 Programmer's Guide

Program Product

Program Number 5746-SM2

IBM

Third Edition (November 1979)

This is a major edition of, and obsoletes, SC33-4044-1. Changes and additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Release 3, Modification 0, of the DOS/VS Sort/Merge Version 2 Program Product (Program Number 5746-SM2) and to all subsequent modifications until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the IBM System/370 Bibliography, GC20-0001, for the editions that are applicable and current. It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for copies of IBM publications should be made to your IBM representative or to the branch office serving your locality.

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

Preface

Who This Manual is For

This manual is for programmers who wish to sort or merge records using the DOS/VS Sort/Merge Version 2 Program Product, 5746-SM2. To sort records is to arrange them in a given order. To merge records is to create one sorted sequence from two or more previously sorted sequences.

To use this manual, you should already have a basic understanding of DOS/VS and its job control language (JCL); in order to take advantage of all the options and facilities of the sort/merge program, you may need to refer to the manuals listed at the end of this preface.

Using this manual, you will be able to prepare all the input necessary to perform a sort or merge. You will also be able to link your own routines (written in assembler language) to the sort/merge program to perform such services as reading input from and writing output to non-supported file types, handling user labels, processing I/O errors, and providing VSAM passwords.

The functions of including and/or omitting certain records, specifying an alternative collating sequence, reformatting the records for output, and summarizing records, can be performed by means of user-specified control statements. It is not necessary to write a routine of your own to perform these functions.

References You May Need

DOS/VS Sort/Merge Version 2 Installation Reference Manual, Order No. SC33-4045

If You are Running under DOS/VS Release 33 or 34

DOS/VS Data Management Guide, Order No. GC33-5372
DOS/VS Supervisor and I/O Macros, Order No. GC33-5373
DOS/VS System Generation, Order No. GC33-5377
Introduction to DOS/VS, Order No. GC33-5370
DOS/VS System Management Guide, Order No. GC33-5371
DOS/VS System Control Statements, Order No. GC33-5376
DOS/VS System Utilities Reference, Order No. GC33-5381
DOS/VS Utilities: Access Method Services, Order No. GC33-5382

If You are Running under VSE/Advanced Functions

Using VSE/VSAM Commands and Macros, Order No. SC24-5144
Using the VSE/VSAM Space Management, Order No. SC24-5192
Introduction to DOS/VSE, GC33-6108
VSE/VSAM Messages and Codes, SC24-5146
VSE/VSAM Documentation Subset, SC24-5191

| VSE/VSAM General Information, GC24-5141
| VSE/Advanced Functicns Data Management Concepts, GC24-5209
| VSE/Advanced Functions Tape Labels, SC24-5212
| VSE/Advanced Functions DASD Labels, SC24-5213
| VSE/Advanced Functicns System Generation, SC24-6096
| VSE/Advanced Functions System Management Guide, SC33-6094
| VSE/Advanced Functions System Control Statements, SC33-6095
| VSE/Advanced Functions Macro Reference, SC24-5211
| VSE/Advanced Functions Macro User's Guide, SC24-5210

Contents

Figures

RELEASE 3 IMPROVEMENTS

- Support of VSAM managed SAM files for input, cutput, and work files.
- Support for simplified JCL for VSAM files.
- Simplified SYSNO handling for all disk files under VSE/Advanced Functions.

SUMMARY OF AMENDMENTS
FOR SC33-4044-1
DOS/VS SORT/MERGE 5746 (SM2)
RELEASE 2 MODIFICATION 0

RELEASE 2 IMPROVEMENTS

- Release 2 of SM2 can now run under DOS/VSE in either System/370 mode or ECPS:VSE mode.
- SM2 accepts input, work, output, or checkpoint files on devices operating in Fixed Block Mode (FBA).
- The program will, if sufficient storage is available, use command chaining when it reads SAM input or writes SAM output in sort applications. Command Chaining is cnly used for CKD and tape devices.
- OUTREC fields can be anywhere within the record (even beyond 4092).
- The program allows SAM input and output files to contain spanned reccrds.
- The standard default values for the program options can be changed to suit the installation when the program is installed.
- The control fields specified in the SORT or MERGE statement can overlap each other.
- The control statement images (parameter list) are optionally printed when the program is invoked.
- Messages may be routed to a device other than SYSLST, when the program is invoked.
- The ANALYZE function makes it possible to find out how SM2 will optimize, and what capacity a sort job will have, without actually sorting or merging.
- The information messages, as well as the diagnostics handling, have been improved.
- There is automatic sequence checking in sort applications.

Chapter 1. Introduction

The DOS/VS Sort/Merge Version 2 Program Product, 5746-SM2, is a generalized sort/merge program which executes as a processing program under DOS/VS Release 33 and 34, and DOS/VSE.

| The VSE/VSAM Release 2 program product is required for VSAM applications
| under DOS/VSE and AF/VSE Release 2. For VSE/VSAM managed SAM files
| support, the 'Space Management for SAM Feature' must also be installed.

This chapter describes SM2 briefly in terms of its functions; how it differs from the 5746-SM1 Program Product; its input and output requirements; the use of control fields, collating sequences and control statements; and its relationship to the operating system.

What the Program Can Do

The program's basic functions are:

1. To sort records from up to nine input files into a user-defined sequence onto an output file

2. To merge records from up to nine previously sorted files onto an output file

The input and output files can be VSAM or SAM. Output need not be the same as input--that is, you can have SAM input and VSAM output, or vice
| versa; but unmanaged input files must be either all SAM or all VSAM.

| VSAM managed SAM files in Control Interval format may be accessed as
| either SAM or VSAM. As input, however, if they are mixed with ordinary
| SAM files they must be accessed as SAM files, and if mixed with VSAM
| files they must be accessed as VSAM. Managed files not in Control
| Interval format cannot be accessed by the program directly but they may
| be read or written using user exits E15 and E35.

The files can reside on tape, Count-Key-Data (CKD), or Fixed Block Mode (FBA) disks as described below under 'Input/Output Devices'. Input files can be on any mixture of permitted devices.

SM2 handles all standard labels, as well as unlabeled tape files.

In addition, user-written routines can be linked to the sort/merge program at points called program exits. At these exits, the user-written routines may write or check nonstandard labels, open or close files, take checkpoints, insert, modify, or delete records, read the input file, write the output file, or process VSAM I/O errors.

Other functions such as sorting only a part of the input, modifying the standard EBCDIC collating sequence, reformatting the records for output, and summarizing records, can be invoked by means of program control statements.

| DIFFERENCES FROM RELEASE 2 OF 5746-SM2

| VSAM managed SAM files may be used as input, output and/or work files.

DIFFERENCES FROM RELEASE 1 OF 5746-SM2

| A major difference in Release 2 of SM2 is that it can now run under
| VSE/Advanced Functions. Under VSE/Advanced Functions it can run in either ECPS:VSE mode or System/370 mode. To use SM2 you do not need to know which of the systems is in use.

A second important difference is that SM2 will now accept input, work, output, or checkpoint files on FBA devices. Conventional CKD devices can be mixed with FBA devices for input and work.

DIFFERENCES FROM 5746-SM1

SM2 is functionally compatible with the DOS/VS Sort/Merge Program Product 5746-SM1, with the exceptions noted in Appendix C under 'Related Programs'. In addition it offers the following features:

- Reduced space requirements for work files (some sorts can be run with no work files at all).
- Code which is reenterable, and therefore eligible to reside in the Shared Virtual Area (SVA).
- Input to either a sort or a merge can be on any mixture of the device types accepted by the program.
- Subtasking is allowed, i.e., SM2 can be initiated by use of an ATTACH macro from within an assembler-language program.
- The user can specify that the input order of records with identical control fields should be preserved in the output file.
- The parameter list passed at exits E15, E32 and E35 has been extended and now includes information on record type and record length.

Input/Output Characteristics

INPUT/OUTPUT FILES

All files must be defined according to DOS/VS standards, as described in
| the DOS/VS System Management Guide, or the VSE/Advanced Functions System
| Management Guide. The characteristics of the input and output files that the sort/merge program can handle are given in Figure 1.

| Input files must be either all accessed as SAM, or all as VSAM. To
| access VSAM files, SM2 uses VSAM. To access unmanaged SAM files it uses
| EXCP commands.

VSAM MANAGED SAM FILES

These files always need the VSAM parameter on their DLBL card. If VSAM or ESDS is specified on the INPFIL or OUTFIL card as appropriate they are accessed as VSAM. Otherwise they are accessed as SAM using GET or PUT.

RECORDS AND DATA FORMAT

Records can be either fixed or variable-length. Variable-length spanned records are permitted; they exclude the use of the ADDROUT option.

The record data can be numeric or alphameric coded EBCDIC (or ASCII for tape input/output), or can have any of the other formats shown in Chapter 2 under 'SORT Control Statement'.

MINIMUM RECORD LENGTH

Minimum record lengths are shown in Figure 2.

	Sort input	Merge input	Output
Files Type of extent	Type 1 and 8	Type 1 and 8	Type 1 and 8
Organization[1]	SAM or VSAM (not mixed)	SAM or VSAM (not mixed)	SAM or VSAM (not mixed)
Number[2]	1 - 9 files	1 - 9 files	1 file (can be multi-volume)
Size	No restriction	No restriction	No restriction
Blocking	Blocked[3] or unblocked	Blocked[3] or unblocked	Blocked or unblocked
Blocksize	Can differ if biggest specified first	Can differ if biggest specified first	
Contents	Unsorted or sorted records (or can be empty)	Previously sorted records (or can be empty)	Sorted records (possibly modified or summarized), or addresses of records (with or without control fields)
Labels	Any or none (can be mixed)	Any or none (can be Mixed)	Any or none
Records Format	Fixed or variable (cannot be mixed)	Fixed or variable (cannot be mixed)	Fixed or variable
Code	EBCDIC or ASCII (may not be mixed)	EBCDIC or ASCII (may not be mixed)	Sam as input
Control fields Number	1 - 12	1 - 12	
Format (see Figure 12 for list of accepted formats)	Formats can be mixed	Format can be mixed	
Sequencing	Ascending and/or descending[4]	Ascending and/or descending[4]	Output to a VSAM KSDS file must be in primary key sequence
Combined lengths, max.	256 bytes	256 bytes	
Location	Must be all within first 4092 bytes of record	Must be all within first 4092 bytes of record	
Pooling[5]	Can be pooled with output	May not be pooled	Can be pooled with sort input

1 VSAM managed SAM files may be mixed with SAM or VSAM but not both at once.

2 If on tape, each unit may have as many alternates as permitted by DOS/VS.

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

4 When records with equal control fields are sorted or merged, their output order is unpredictable if EQUALS is not specified.

5 VSAM input and output can be the same data set only if the data set was defined with the REUSE attribute and the OUTFIL REUSE parameter is given to SORT/MERGE.

Figure 1. Input and Output Characteristics

	RECORD LENGTH:				BLOCK SIZE:
	Minimum		Maximum		
	TYPE=		TYPE=		
	F	V	F	V	Maximum
Tape input	12	12	32767	32767[1]	32767
Tape output	18	18	32767	32767[1]	32767
CKD input/output	1	5	Track capacity	32767[1]	Track capacity
FBA input/output	1	5	32761	32767[1]	32761
VSAM managed SAM	1	5	32761	32767[1]	32761
VSAM input/output	1	1[2]	32767[1]	32767[1]	N/A

[1]Implies spanned records
[2]You must add four bytes when specifying the length, because SM2 adds an RDW

Figure 2. Block Size and Record Length

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

MAXIMUM BLOCK SIZE AND RECORD LENGTH

Maximum block sizes and record lengths are shown in Figure 2.

The maximum block size for input and output files is 32,767 bytes for EBCDIC data and 9,999 bytes for ASCII data.

There are two restrictions on the use of large blocks:

- The larger the blocks, the more main storage your sort or merge will need. Main storage requirements in relation to block size are described in Appendix B.
- For CKD devices you cannot define an output block size larger than track capacity for the output device you have specified.

Input records can be as big as input block sizes, but here there are additional restrictions:

- If you are using any CKD work devices the internal record must not be longer than the track length of the work device. If work devices are mixed, the shortest track length is the limiting factor. Usually internal length is the same as input length, but the use of some control field formats, and the EQUALS parameter of the SORT statement, cause the record to be expanded internally.
- FBA work devices limit the internal record length to 32767 bytes.
- The record length must not exceed the maximum length specified by the user on the RECORD control statement.
- With variable-length records the four-byte record descriptor word is regarded as part of the record. Also, maximum record length is four bytes less than block size because of the block descriptor word.

Input/Output Devices

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

- IBM 2311, 2314, 2319, 3330 models 1, 2 or 11, 3333 models 1 or 11, 3340, 3344 or 3350 CKD direct-access storage devices
- IBM 2400 and 3400 series tape units or 8809 in start/stop mode
- IBM 3310 and 3370 FBA direct-access storage devices.

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

Input/Output Pooling

You can design a sort application where two of the I/O files share the same disk extents or tape units. This allows you to run your sort with fewer devices, and is known as I/O pooling. The rules for I/O pooling are:

- Sort input and output files can be pooled. This means that the output file will be written over the input file, so it is important to check that you have specified the files correctly.
- Any file name can be used.
- Files can be multi-volume and multi-extent.
- Merge-only files must not be pooled.
- Sort work files must not be pooled with any other sort file. If they are, an error message is produced and the program terminates.

Input Device Sharing

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

Work Storage Devices

If work storage (sometimes called intermediate storage) is needed for your sort application it must be on a disk device. Any of the devices listed in Figure 3 can be used as long as they are supported by your system.

Type:	Devices:
2314	2314, 2319
3330	3330 mods 1, 2, 11 3333 mods 1, 11
3340	3340, 3344
3350	3350
FBA	3310, 3370

Figure 3. Devices That Can Be Used For Work Files

SM2 allows the work files to be allocated on two different device types, where device type is considered to be the same if track capacity and the number of tracks per cylinder is the same.

Tape units cannot be used for work storage.

Label Processing

If your files have standard labels or are unlabeled, then SM2 will take care of all opening, closing, and label handling for you. If you have nonstandard labels (or extra headers or trailers in addition to standard labels) you must use a program exit to carry out label processing, as described in Chapter 5.

Standard Labels

Standard direct-access and tape labels are processed by the system's label processing facilities when the OPEN and CLOSE macro instructions are issued by SM2.

Unlabeled Tapes

Unlabeled input and output files are processed by the sort/merge program; no user programming is required. Unlabeled output files are normally preceded and followed by a tape mark, but the leading tape mark can be eliminated by specifying NOTPMK in the OUTFIL program control statement.

Nonstandard Direct-Access Labels

The sort/merge program will not attempt to use the first track of the first extent in any CKD input or output file specified by the user as having nonstandard labels. This track may be handled in any way the

user chooses by routines at exits E11, and E31. With an FBA file the amount of space left for label processing depends on the control interval size, as described in Chapter 5.

Nonstandard Tape Labels

If labels are not standard, or if the standard tape labels include additional header and trailer labels or user header and trailer labels, then the user must assume the responsibility for issuing the OPEN and CLOSE macro instructions and processing these extra labels.

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

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

DOS/VS Data Management Guide, GC33-5372
DOS/VS Tape Labels Reference, GC33-5374
DOS/VS DASD Labels Reference, GC33-5375
VSE/Advanced Functions Data Management Concepts, GC24-5209
VSE/Advanced Functions Tape Labels, SC24-5212
VSE/Advanced Functions DASD Labels, GC24-5209

Passwords for VSAM Files

SM2 allows the use of password-protected VSAM files for input and output. When a password is needed for a VSAM file, the appropriate user routine is entered at exit E18 for sort input, E38 for merge input, and E39 for sort/merge output. In this routine the user has the opportunity to supply the required password(s). If a password is not supplied at an exit, VSAM then asks the operator to supply the password. User exits E18, E38 and E39 are only available for VSAM managed SAM files if they are accessed as VSAM.

Control Fields and Collating Sequences

CONTROL FIELDS

The program determines the sequence of a file of data records by using a group of up to twelve control fields which must appear in the same relative position in each record. The sequence can be either ascending or descending.

The control fields are specified in the SORT or MERGE program control statement (see Chapter 2). The first control field specified is the major control field and will be compared first. The second and subsequent control fields (minor fields) will only be compared when the previous comparison has resulted in an equal condition.

The individual control fields may be contiguous or separated, or may overlap (see Figure 4). The control fields may be anywhere within the first 4092 bytes of a data record but their total lengths must not exceed 256 bytes.

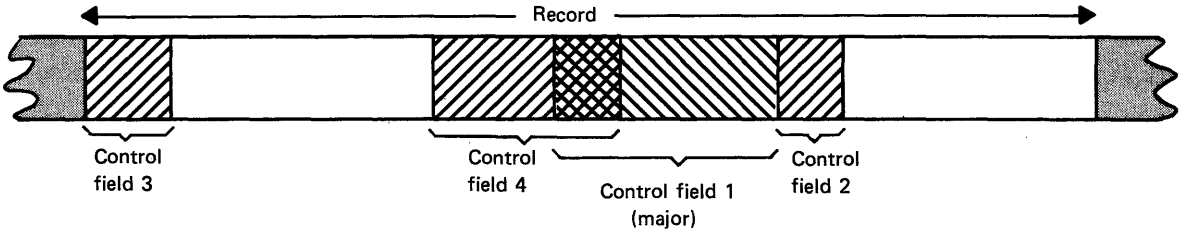


Figure 4. Control Fields

In Figure 4, fields 1 and 4 overlap, fields 1 and 2 are contiguous, and fields 3 and 4 are noncontiguous.

The types of data format that may be specified for control fields are described in Chapter 2, 'Program Control Statements', under the 'SORT Control Statement'. A detailed description of the formats is given in Appendix D. SM2 does not check that the control fields actually contain data of the specified format. If they do not the output is unlikely to be correctly sorted; under some circumstances SM2 might abend.

COLLATING SEQUENCES

When the contents of the control fields are compared, the resulting sequence is determined by either the standard IBM collating sequence (EBCDIC), or the ASCII collating sequence. The collating sequence for character data and binary data is absolute; that is, character and binary fields are not interpreted as having signs. For packed decimal, zoned decimal, fixed-point, normalized floating-point, and the signed numeric data formats, collating is algebraic; that is, each quantity is interpreted as having an algebraic sign.

Either an ascending or a descending sequence can be specified for each control field.

If your input and output are in EBCDIC format, you can choose between the EBCDIC and ASCII collating sequences. If, however, your input format is specified as ASCII (AC, ASL, or AST), then only the appropriate ASCII collating sequence is allowed.

Alternative Collating Sequence

The EBCDIC collating sequence can be modified by means of the ALTSEQ program control statement. This allows the positions of one or more characters in the sequence to be changed--for example, to allow correct collation of special national characters.

PROGRAM CONTROL STATEMENTS

Before SM2 can operate on the input data, you must supply the program with control statements that describe the actions and operations you want it to perform. The control statements you provide describe:

- The type of operation to be performed
- Control fields
- Modifications to be made by user-written routines
- The functions to be invoked (for example OMIT)
- The input and output data sets
- The options selected for each application

A full description of all the program control statements is given in Chapter 2.

Each control statement is checked for validity by SM2. If the program finds an error, it issues an error message. Descriptions of these messages can be found in Appendix E.

Control statement formats for all IBM DOS and IBM DOS/VS sort/merge programs are similar, even though operating environments and data descriptions are different. Compatibility of control statements among these programs is discussed in Appendix C.

JOB CONTROL STATEMENTS (JCL)

Standard job control statements are required to define a sort or merge application to the Job Control program. A discussion covering the relevant statements can be found in Chapter 3. For a full description of job control statements and their formats you should refer to DOS/VS System Control Statements or VSE/Advanced Functions System Control Statements.

INITIATING THE PROGRAM

As described in Chapter 4, 'Executing the Program', SM2 can be initiated in two ways:

- As an independent program, using an EXEC SORT statement in the input job stream.
- Invoked from another program written in one of the following languages:
  - Assembler, by use of system macros, as described in Chapter 4.
  - COBOL
  - PL/I
  - RPG II with the Auto-Report Feature

  How to do so from the high-level languages is described in the Programmer's Guide for the relevant language.

In either case the initiation must follow DOS/VS conventions.

PROGRAM MODIFICATION

SM2 allows you to incorporate your own routines into the main flow of a sort or merge operation. These routines can, for example, be used to process file labels; read input files; modify, insert, or delete records; or write output files.

The routines receive control at predesignated points in the sort/merge program called program exits. Chapter 5, 'Modifying the Program', gives a full description of the exits and how they can be used.

Relationship to DOS/VS and VSE Advanced Functions

SM2 is designed to run under DOS/VS Release 33 and 34, DOS/VSE with VSE/Advanced Functions and all subsequent releases. Figure 5 shows which supervisor/generation macros can affect SM2.

The VSE/VSAM Release 2 program product is required for VSAM applications. For VSE/VSAM managed SAM files support, the 'Space Management for SAM Feature' must also be installed.

For further details see DOS/VS System Generation, GC33-5377 or VSE/Advanced Functions System Generation, GC33-6096 and DOS/VS Sort/Merge Version 2 Installation Reference Manual.

SM2 can reside in either the system or any private core image library, and most modules can be loaded into the SVA. It operates as a processing program supervised by DOS/VS.

SM2 may be executed in any partition. A single copy in the core image library will serve all partitions.

Any files used by SM2 must be defined according to DOS/VS standards. The checkpoint and label-checking facilities of DOS/VS are used during a

System/370 mode		
	FOPT RELLDR=YES	ALWAYS REQUIRED (standard in DOS/VSE)
	FOPT AB=YES	to enable the DUMP option to be fully exploited
	FOPT ECPREAL=YES FOPT PFIX=YES	to enable page fixing to be done by sort/merge
	FOPT RPS=YES	aids efficiency if 3330, 3340 or 3350 devices are used
	FOPT VSAM=YES	required if VSAM files are to be used for input or output
	FOPT SYSFIL=YES	required if the distributed file is deblocked to disk when SM2 is to be installed
	SUPVR AP=YES	to enable subtasking to be used
	SUPVR ASCII=YES	required if ASCII data is to be used
ECPS:VSE mode		
	FOPT RPS=YES	aids efficiency if 3330, 3340 or 3350 devices are used
	FOPT VSAM=YES	required if VSAM files are to be used for input or output
	FOPT SYSFIL=YES	required if the distributed file is deblocked to disk when SM2 is to be installed
	SUPVR AP=YES	to enable subtasking to be used
	SUPVR ASCII=YES	required if ASCII data is to be used

Figure 5. Supervisor Generation Macros Relevant to SM2

sort/merge program execution at the user's option. For a general discussion of DOS/VS, refer to the publication Introduction to DOS/VS or Introduction to DOS/VSE as appropriate. For a discussion of record formats and data file organization, refer to the publication DOS/VS Data Management Guide or VSE/Advanced Functions Data Management Concepts.

Chapter 2. Program Control Statements

Before SM2 can operate on the input data it must receive program control statements. Some control statements are always required whereas others are optional and are only required for specific actions. The control statements describe:

- The type of operation to be performed
- Control field parameters
- Modifications to be made by user-written routines
- The functions to be invoked (for example OMIT)
- The input and output files
- The options selected for particular applications

Each control statement is checked for validity by SM2. If the program finds an error in a statement, it issues a diagnostic message and will usually skip the rest of the statement (including any continuation cards) and continue checking the next statement. If an error has been found, SM2 usually terminates when it has finished checking all the statements.

Control statements are read from SYSIPT. Any SYSIPT records can be read.

The Statements

The control statements that SM2 acts on are listed below and a summary of the statements with their operands is given in Figure 6.

{SORT / MERGE} One of these statements is always required. It is used to describe the control fields and the number of input files (and work files for SORT).

RECORD This statement is always required. It specifies record length and format information to the program.

INPFIL This statement describes the input files and specifies the procedures that will be followed when an input file is opened and closed. It is required if the default values are not applicable (for example, if SAM input is blocked).

OUTFIL This statement describes the output file and specifies the procedures that will be followed when the output file is opened and closed. It is required when the default values are not applicable (for example, if SAM output is blocked).

MODS This statement is required when user-written routines are to be executed. It associates the user routines with particular program exits.

[INCLUDE / OMIT] These statements are optional. One (but not both) can be used to specify that the output file should contain only certain of the input records.

ALTSEQ This statement is optional. It is used to specify changes to the standard EBCDIC collating sequence.

OUTREC This statement is optional. It can be used to specify that each input record should be reformatted before being written to the output file.

SUM This statement is optional. It can be used to designate numeric fields in the input records as summary fields, and specifies that whenever two records with equal control fields are found, the contents of the summary fields are to be added and placed in one of the records, and the other record deleted.

ANALYZE This statement is optional. It instructs SM2 to analyze the control statements, make its optimization calculations, issue appropriate messages, and then cancel without actually sorting or merging.

OPTION This statement specifies the options to be selected for a particular sort/merge operation. It is required if the default values are not applicable.

The control statements may appear in any order, but you are recommended to put the OPTION statement, if used, before all the others.

The 'default values' referred to above are those valid at your own installation. Some defaults are fixed; others can be changed after the program is installed, so the standard defaults supplied with the program, and described in this chapter, may not be those actually in effect.

Statement	Parameters
SORT	FIELDS=(p_1,m_1[,f_1],s_1[...,p_n,m_n[,f_n],s_n]) [,FORMAT=f]
	FILES=n
	EQUALS|NOEQUALS
	WORK=±n|DA}
	CKPT
MERGE	FIELDS=(p_1,m_1[,f_1],s_1[...,p_n,m_n[,f_n],s_n]) [,FORMAT=f]
	FILES=n
RECORD	TYPE=±F|V|D}
	LENGTH=(l_1,l_2,l_3,l_4,l_5)
MODS	PHn=(name,loading information,$exit_1$[...,$exit_n$])...
INPFIL	BLKSIZE=n
	EXIT
	BYPASS
	VSAM
	TOL
	SPAN
	VOLUME=(n[...,n])
	OPEN=±RWD|NORWD}
	CLOSE=±RWD|UNLD|NORWD}
	DATA=E|A
	BUFOFF=n
	NOCHAIN
OUTFIL	BLKSIZE=n
	EXIT
	KSDS|ESDS|RRDS
	TOL
	REUSE
	SPAN

Figure 6 (1 of 2). Program Control Statement Summary

Statement	Parameters
(OUTFIL)	OPEN={RWD|NORWD} CLOSE={RWD|UNLD|NORWD} NOTPMK BUFOFF=n
[INCLUDE OMIT]	COND=(logical expression) [,FORMAT=f]
ALTSEQ	CODE=(fftt[,ffft...,ffft])
OUTREC	FIELDS=(p_1,m_1[,a_1]...[,p_n,m_n[,a_n]])
SUM	FIELDS=(p_1,m_1[,f_1]...[p_n,m_n[f_n]]) [,FORMAT=f]
ANALYZE CALC	
OPTION	PRINT={ALL|NONE|CRITICAL} ROUTE={LST|LOG|xxx} STORAGE={n[,VIRT|NOVIRT] nK[,VIRT|NOVIRT]} LABEL=(output,input$_1$...,input$_n$) WORKNM=work FILNM=(output,input$_1$...,input$_n$) SORTOUT=output SORTIN=(input$_1$...,input$_n$) SORTWK=(work$_1$...,work$_n$) VERIFY|NOVERIFY ERASE|NOERASE DIAG|NODIAG DUMP|NODUMP ADDROUT

Figure 6 (2 of 2). Program Control Statement Summary

Control Statement Format

Control statement formats for all IBM DOS and IBM DOS/VS sort/merge programs are similar, even though operating environments are different. Compatibility of control statements among these programs is discussed in Appendix C.

The general format is shown in Figure 7.

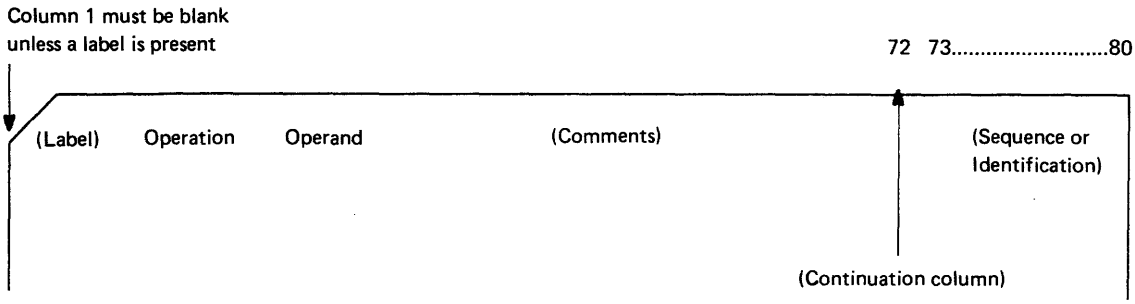
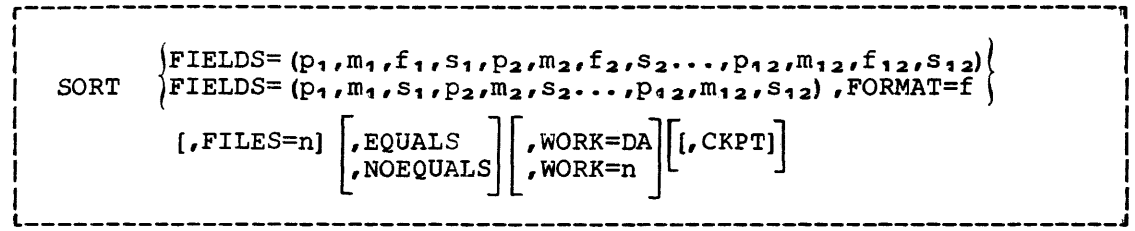


Figure 7. Example of a Control Statement

The control statements are free-form--that is, the operation definer, operand(s), and comments may appear anywhere in a statement, as long as they appear in the proper order, and are separated by one or more blank characters. Column 1 of each control statement must be blank, unless the first field is a label, in which case it must begin in column 1.

Examples are shown in the text which follows, and full details of all rules are given in Appendix A, 'Sample Job Streams, With Statement Format Rules'.

SORT Control Statement



The SORT control statement is required when a sort is to be performed. It supplies specifications for control fields, input files, and work files.

Operand	Description
FIELDS=	Keyword for control field parameters, always required. Field parameters must be described in descending order of priority. You can specify up to twelve control fields.
p	First byte of control field relative to beginning of logical record. The first data byte of a fixed-length record has a relative position 1. The first data byte of all variable-length records has a relative position 5 as the first four bytes are occupied by the RDW. All fields must start on a byte boundary and no field may extend beyond byte 4092.
m	Length of control field in bytes, which must include the sign for signed data. Acceptable lengths for different formats are given below. The sum of the control field lengths must not exceed 256 bytes.
f	Format of data in the control field, which may be any of the following codes:

Code	Length	Description
CH	1-256	EBCDIC character, unsigned
ZD	1-256	Zoned decimal, signed
PD	1-32	Packed decimal, signed
FI	1-256	Fixed point binary, signed
BI	1-256	Binary, unsigned
FL	1-256	Normalized floating point, signed
AC	1-256	ASCII character, unsigned
CSL	2-256	EBCDIC numeric, leading separate sign
CST	2-256	EBCDIC numeric, trailing separate sign
CLO	1-256	EBCDIC numeric, leading overpunch sign
CTO	1-256	EBCDIC numeric, trailing overpunch sign
ASL	2-256	ASCII numeric, leading separate sign
AST	2-256	ASCII numeric, trailing separate sign
AQ	1-256	EBCDIC character, alternative collating sequence

s — Order in which control field is to be sorted. Must be either:
A - ascending sequence or
D - descending sequence.

FORMAT=f — Optional. Can be used when all control field data formats are the same. The f_1-f_n can be omitted from the FIELDS operand and be replaced by this operand. f can be any of the formats specified in the FIELDS keyword.

FILES=n — n is the number of input files to be sorted and can be any number from 1 through 9. The default value is 1.

EQUALS — Specifies that the order of records whose control fields are identical is to be preserved from input to output. Use of this option degrades SM2's performance.

NOEQUALS — Specifies that the input order of equal records need not be preserved. This is the standard default, but can have been changed at your installation.

WORK= — Describes the work files.

DA — The default; means that the work file DLBL statement defines the file as multiextent (DA). (Not allowed for FBA workfiles).

n — Must be supplied for a sort if the default is not applicable. Gives the number of SORTWK DLBL statements supplied; can be 0-9. If 0 is specified the sort will use no work files and will attempt to complete the sort in main storage. Furthermore, if 0 is specified command chaining will not be attempted.

Note: Under DOS/VSE WORK=DA means the same as WORK=1. Sort will determine if the files are DA, SD, or VSAM managed SAM.

CKPT — Tells the sort/merge program to activate the checkpoint facilities of the operating system.

The checkpoint file must:

1. Be assigned to SYS000 (a tape or a direct-access device, CKD or FBA)
2. Have standard labels
3. Have the file name SORTCKP

Only one checkpoint is taken by the program, at the beginning of the last phase.

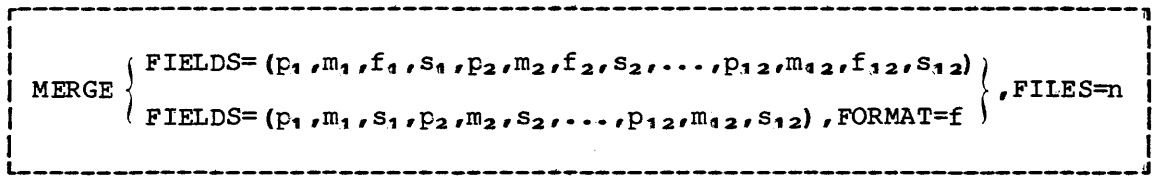
If SM2 was invoked from a user program, or if user-written routines are in use, the entire virtual partition is checkpointed. Otherwise, only the area being used by sort/merge is checkpointed. Checkpointing a large partition will adversely affect sort elapsed time.

You cannot specify the restart option via a sort/merge control statement; you must use the DOS/VS restart facilities to continue an interrupted job. The general procedure is (1) to submit a RSTRT job control statement to request the system restart facilities, and (2) ASSGN statements to reestablish the logical device assignments that were in effect at the time the checkpoint was taken. The DOS/VS checkpoint/restart facilities are described in

the publication: DOS/VS System Management Guide and VSE/Advanced Functions Macro User's Guide.

Checkpoint is ignored if a) no work files are used, or b) SM2 is subtasked, or c) exit E31 is specified in the MODS statement. In case (c) SM2 will pass a device list to your routine at E31 so that you can take a checkpoint there.

MERGE Control Statement



The MERGE control statement is required when a merge of presorted files is to be performed.

Operand	Description
FIELDS=	Keyword for control field parameters, always required.
p m f s	Control field definition parameters. Same notation as for SORT statement.
FORMAT=f	Same use as for SORT statement.
FILES=n	Must be specified. It defines the number of input files to be merged, where n is any number from 1 through 9.

SORT/MERGE STATEMENT PROGRAMMING NOTES

1. All control fields must be located within the first 4092 bytes of a logical record.
2. All floating-point data must be normalized before SM2 can collate it properly. You may provide a routine at exit E15 (sort) or E32 (merge) to normalize floating-point data at execution time.
3. In control fields, +0, 0, and -0 are treated as the same number and compare equal.
4. The maximum length of packed decimal fields is 32 bytes.
5. The total number of bytes occupied by all control fields must not exceed 256.
6. If all control field data formats 'f' are the same, they can be omitted and the FORMAT keyword used instead.
7. Input files must be all SAM or all VSAM.
8. If any of the input files are multivolume and unlabeled, you must specify the number of volumes using the VOLUME operand of the INPFIL statement.
9. If WORK=DA is specified or defaulted, the first extent must contain at least two tracks.
10. EQUALS is ignored when SUM is specified.

SORT/MERGE STATEMENT EXAMPLES

Example 1

```
SORT    FIELDS=(2,5,CH,A,12,10,BI,D),WORK=1
```

Instructs the program to sort one input file using two control fields. One sequential disk extent is available for work storage.

The first control field contains unsigned EBCDIC character data starting on the second byte and is 5 bytes long. It is to be sorted into ascending order.

The second control field contains unsigned binary data starting on the twelfth byte and is 10 bytes long. It is to be sorted into descending order.

Example 2

```
SORT    FIELDS=(25,4,A,48,8,A),FORMAT=ZD,WORK=DA
```

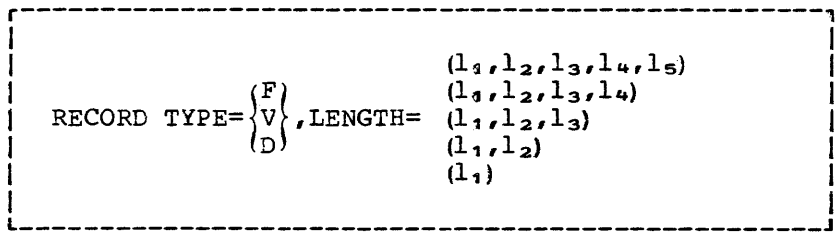
Since both control fields contain zoned decimal data, the FORMAT operand can be used. The input file is to be sorted into ascending sequence based on the contents of two control fields. The first control field is 4 bytes long starting on byte 25 and the second field is 8 bytes long starting on byte 48. Work storage is on a multiextent direct-access disk file. Under DOS/VSE WORK=DA could refer either to a DA file or a VSAM managed file.

Example 3

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

SM2 is instructed to merge the input records from three files into ascending order based on a single, character control field. The field begins in the second byte of each record and is five bytes long.

RECORD Control Statement



A RECORD control statement must always be provided. It defines the logical records to be sorted or merged.

Both the TYPE and the LENGTH operands must be included in the statement.

Operand	Description
TYPE={F, V, D}	Record type: F - Fixed-length records V - Variable-length EBCDIC records D - Variable-length ASCII records
	For Fixed-Length Records
LENGTH=	Keyword for record lengths, which must be expressed in bytes.
l_1	Length of logical record in the input file. It is always required.
l_2 (note 2)	Required when the length is changed at exit E15 (sort). The value is the length after E15. Default is l_1.
l_3 (note 2)	Required when the length is changed at exit E35. The value is the length after E35. Default is l_2 for a sort or l_1 for merge.
	For Variable-Length Records
LENGTH=	All lengths must be expressed in bytes and include the four byte record descriptor word (RDW).
l_1	Maximum record length in the input file. Always required.
l_2 (note 2)	Required when the maximum record length is changed at exit E15 (scrt). The value is the maximum record length after the exit routine. Default is l_1.
l_3 (note 2)	Required when maximum output record length is changed at exit E35. The value is the maximum record length after the exit routine. Default is l_2 for a sort or l_1 for merge.
l_4	Specifies the minimum record length after exit E15. Default is the length from the beginning of the record to the end of the last (rightmost) field referred to in any SORT, MERGE, INCLUDE, OMIT, SUM, or OUTREC statement - unless that length is less than 14 bytes, in which case 14 bytes is assumed.

l_5 Specifies the modal (most frequent) record length and should be as accurate as possible to aid performance. Default is the average of l_2 and l_4:

$$l_5 = \frac{l_2 + l_4}{2}$$

RECORD STATEMENT PROGRAMMING NOTES

1. A LENGTH parameter must be supplied, with the l_4 value. Other values can be omitted as indicated above.

2. Do not specify l_2 or l_3 with ADDROUT or OUTREC:

   a. If ADDROUT is specified in the OPTION statement it is unnecessary to specify l_2 or l_3. If, however, you do specify l_2 and l_3 then the values must be as shown below or the program will terminate.

      - l_2 = 10 + CF for SAM or 5 + CF for VSAM files

      - l_3 = 10 for SAM or 5 for VSAM files

      where CF is the sum of the control field lengths in bytes.

      l_4 and l_5 will be calculated by SM2.

   b. If OUTREC is specified, you need specify only l_4: SM2 will calculate and insert the correct values for l_1 and l_3 for you.

3. The lengths specified for variable-length records (V or D type) must include the four-byte Record Descriptor Word (RDW) that DOS/VS standards require. This applies even for VSAM variable-length records, which normally have no RDW, since SM2 has to build an RDW for these records when reading them in.

4. The minimum logical record length is given in Chapter 1, 'Introduction'. Tape input with records less than 12 bytes is accepted but may cause problems if the last block is not full and error recovery is attempted.

5. Record size must not exceed the track size of any CKD direct-access device used for work files.

RECORD STATEMENT EXAMPLES

Example 1

```
RECORD TYPE=F,LENGTH=80
```

The statement defines the input records as fixed-length, 80 bytes long. This format of the RECORD statement is sufficient for all fixed-length record statements provided user routines are not used to modify the record lengths.

Example 2

```
RECORD TYPE=F,LENGTH=(80,,50)
```

The statement defines fixed-length records 80 bytes long which are changed by a user routine at exit E35. The omission of the l_2 value is indicated by the two commas.

Example 3

```
RECORD  TYPE=D,LENGTH=(104,,,44,84)
```

The statement defines variable-length ASCII input records which are not modified by user routines. The maximum record length is 104 bytes and the minimum length 44 bytes. The modal length (most frequent record length) in the input file is 84 bytes.

Example 4

```
RECORD  TYPE=F,LENGTH=(215,22,5)
```

Assuming that this statement is used in conjunction with the ADDROUT parameter in the OPTION statement and the records are on a VSAM file, then the statement tells the program:

- The input record length l_1 is 215 bytes.
- The l_2 value shown (22) represents five bytes for disk address plus a 17-byte control word for each record.
- Each output record contains a five-byte address rather than data. The l_3 value (if given) must be 5 for a VSAM file with the ADDROUT option specified (see programming note 2 above). However you do not need to give this value, nor that for l_2, as they can be supplied by default: the statement RECORD TYPE=F,LENGTH=215 is also correct for the above example.

MODS Control Statement

```
MODS  PHn=(name,loading information,exit1,exit2...,exitn),...
```

The MODS control statement specifies how SM2 is to be modified by user-written routines. It identifies the program exits which will be active in each phase, and associates your routines with these exits. Chapter 5, 'Modifying the Program' describes the phases, and the uses of program exits.

Operand	Description
PHn	n specifies the program phase which is to be modified: • PH1 specifies the internal sort phase (phase 1) • PH3 specifies the final merge phase (phase 3) Both may be specified in one MODS statement.
name	Specifies the cataloged name of routines to be executed at the exits specified. The 'name' must be omitted if the routines are already in main storage.
loading information	Describes either of two ways the user routines may be loaded into main storage by SM2. The value can be the absolute loading address (expressed in decimal), or the length of the user routines to be executed (expressed decimally in bytes and prefixed with L). Loading information must be omitted if the routines are already in main storage.
exit	Specifies the program exits that are to be used in the specified program phase. The values are expressed as E31, E38, and so on. They can be in any order within the phase.

MODS STATEMENT PROGRAMMING NOTES

1. The absolute loading address must be a virtual address if SM2 is to run in virtual mode, and a real address if run in real mode.
2. If the absolute address is used then the length of the module containing your routines is unknown to SM2. This means that all main storage beyond the address given is unavailable to SM2 for other use.
3. If the length of the user routines is given, the routines must be:
   - Self-relocating (unmodified address constants cannot be used)
   - Or, eligible for relocation by the system loader program

   This enables SM2 to load your routines in the most suitable position to leave maximum storage available for other uses.

MODS STATEMENT EXAMPLES

Example 1

```
MODS     PH1=(USERB,L500,E15),PH3=(USERC,L800,E31,E38,E39)
```

User routines are specified for the first and third phases of the sort/merge program. USERB contains a routine 500 bytes long, which is executed at exit E15. USERC is 800 bytes long and contains the routines executed at exits E31, E38, and E39.

Example 2

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

Two program exits are specified for the third phase. The user routine code is already in main storage, so its name and loading specifications are not included. The absence of these two values is indicated by commas. Preloaded user routines are used only for sorts or merges that are initiated by another program.

INPFIL Control Statement

```
INPFIL  [BLKSIZE=n] [,EXIT] [,BYPASS] [,VSAM] [,TOL] [,SPAN]
        [,VOLUME=n      ] [,OPEN=RWD   ]
        [,VOLUMN=(n,...)] [,OPEN=NORWD ]
        [,CLOSE=RWD   ] [,DATA=E                ] [,NOCHAIN]
        [,CLOSE=UNLD  ] [,DATA=A [,BUFOFF=n]    ]
        [,CLOSE=NORWD ]
```

The INPFIL control statement defines the input files to SM2 and specifies the procedure to be followed when tape files are opened or closed.

The statement is only required if the default values are not applicable.

Operand	Description
BLKSIZE=n	n specifies the maximum input block size in bytes. It is not needed if: - Input is VSAM, accessed as VSAM, or - All input is on FBA devices, or - All input files are unblocked Otherwise it must be supplied, because SM2 does not support the DLBL BLKSIZE parameter. When specified for variable-length records BLKSIZE must include the block descriptor word (4 bytes). For ASCII data it must include the BUFOFF value. Defaults are: - Record type F: l_1 value from RECORD statement - Record type V: l_1 value plus four bytes - Record type D: l_4 value plus BUFOFF value If input files have different block sizes, n must be equal to the largest block size. The maximum permissible block size for an input file is shown in Figure 2 in Chapter 1.
EXIT	Specifies that you wish to read all input data yourself and pass each record to the program. You must: • Activate program exit E15 (sort) or E32 (merge) by specifying the exit on the MODS statement. • Provide a routine at exit E15 or E32 which will open the file(s), read them and pass the records, one at a time, to SM2. When EXIT is specified all other INPFIL parameters except DATA are ignored. See INPFIL Programming Notes 1-3 for further details.
BYPASS	Specifies that the program is to skip incorrectly read CKD input data blocks and wrong-length physical records. The program will continue but prints an information message. If the operand is not specified the program will print a message and terminate on encountering the above conditions.

If BYPASS is specified for VSAM managed SAM files the rest of the control interval in error is bypassed.

If BYPASS is specified for FBA input it is ignored.

If BYPASS is specified in conjunction with the SPAN parameter, all records partly or wholly in a block with an error will be skipped.

BYPASS prevents command chaining of the input CCWs.

VSAM — Indicates to the program that the input files are VSAM. If VSAM is not specified it is assumed that the input files are SAM files. The VSAM parameter may also be used to access VSAM managed SAM files with VSAM.

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

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

TOL — Indicates that SM2 will tolerate a warning code from VSAM when opening a VSAM input data set. It is only valid for VSAM input. See Note 6 below.

SPAN — Indicates that input consists of variable-length EBCDIC records in non-VSAM files, and that the records may be spanned. The RECORD statement must show TYPE=V. This parameter is not required if input consists of spanned VSAM records.

VOLUME=(n_1,n_2..,n_9) — This operand should only be used for unlabeled files; if specified for a labeled file, it is ignored. n is the number of volumes in each input file and can have the value 1 through 255. The values must be specified in the order SORTIN1, SORTIN2...,SORTIN9. Default value is 1.

OPEN= — This parameter only applies to tape input files.

- RWD — First volume of each input file is to be rewound before being read (default).
- NORWD — First volume of each input file is not to be rewound before being read.

CLOSE= — This parameter only applies to tape input files. At end-of-file the tape input volumes are:

- RWD — To be rewound (default)
- UNLD — To be rewound and unloaded.
- NORWD — Not to be rewound.

DATA=E — Specifies that the data input is EBCDIC (default).

DATA=A — Specifies that the data input is ASCII. It can only be specified for nine-track tape.

BUFOFF=n — Only used with ASCII data. It specifies the block prefix size at the front of each physical record on the input file. n can be any value from 0 through 99. Default is 0.

NOCHAIN Indicates that SM2 should not use command chaining when reading input. Should only be used (for performance reasons) when input consists of blocked, fixed-length records on tape files, and you know that a large number of blocks are shorter than the specified block size, for example:
- when input files have different block sizes, or
- one or more files contain a large number of short blocks

See Chapter 6 for a discussion of performance.

INPFIL STATEMENT PROGRAMMING NOTES

1. Any parameters which precede EXIT in the INPFIL statement are checked for validity and flagged. Parameters which follow EXIT are also checked and flagged for syntax errors but all values are ignored.

2. The presence or absence of the EXIT parameter affects the default symbolic unit names for the other files used by the program. See Figure 11 in Chapter 3 for details.

3. If the EXIT parameter and the ADDROUT option are specified the program will terminate and issue the message 7D09I ADDROUT OPTION INVALID.

4. SM2 adds a record descriptor word (RDW) of four bytes to every variable-length record received from a VSAM file. The user-written routines which handle variable-length records received from SM2 need not, therefore, be designed separately for handling VSAM and non-VSAM files.

5. If TOL is not specified for a VSAM input data set, a warning return code from VSAM will normally be recognized as an error by SM2; an error message will be printed and the program will terminate.

   By specifying TOL you can sort or merge from the data set without repairing it, and no error message will be printed. Conditions producing warning messages are, for example: 'NOT PROPERLY CLOSED' or 'SYSTEM TIME STAMPS OF DATA AND INDEX DO NOT MATCH'.

   A critical return code from VSAM will always cause SM2 to terminate, regardless of whether TOL has been specified.

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

7. For VSAM managed SAM files the BLKSIZE parameter refers to the logical block size. That is, it should match the RECORDSIZE (maximum) parameter in the DEFINE statement for the file when it was created with VSAM Access Method Services. Note also that too long fixed length input blocks are truncated by data management and cannot be checked by the program.

INPFIL STATEMENT EXAMPLES

Example 1

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

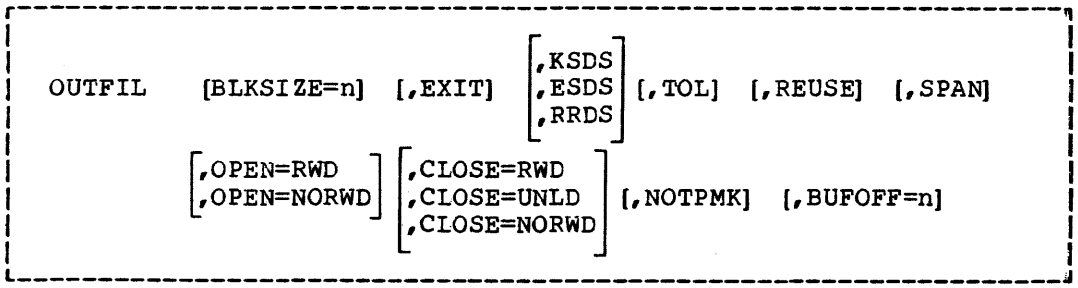
The four input files consist of three, five, two, and one unlabeled tape volumes respectively; block size is 400 bytes; the first volume of each input file is not to be rewound before being read; incorrectly read input data blocks are to be ignored; input data is ASCII; and each block has a buffer offset of 99 bytes.

Example 2

```
INPFIL  EXIT
```

All input is received by SM2 from a user-written routine at either E15 (for a sort) or E32 (for a merge).

OUTFIL Control Statement



The OUTFIL control statement defines the output file to SM2, and specifies the procedure to be followed when tape files are opened or closed.

Operand	Description
BLKSIZE=n	n specifies the maximum block size, in bytes, of the output records. It is not needed if output is VSAM or accessed as VSAM, nor if output is to be unblocked. Otherwise it is needed, because SM2 does not support the DLBL BLKSIZE parameter. When specified for variable-length records it must include the block descriptor word (4 bytes). For ASCII data it must include the BUFOFF value. Defaults are: Fixed-length records: l_3 value from RECORD statement Variable-length records: l_3 value plus four bytes ASCII records: l_3 value plus BUFOFF value If OUTREC is specified SM2 will calculate the output block size according to the OUTREC statement.
EXIT	Specifies that your own routine will take responsibility for the output file instead of SM2. You must: • Activate program exit E35 by specifying it on the MODS statement. • Provide a routine at E35 which will receive each output record from SM2. This routine must take complete responsibility for output. When EXIT is specified, all other OUTFIL parameters are ignored.
{KSDS / ESDS / RRDS}	Any one of these parameters indicates that the output file is to be stored in a VSAM data set. May also be used to write a VSAM managed SAM file with VSAM access. KSDS: Specifies that the VSAM data set is to be key-sequenced. The records must have been sorted into ascending key sequence on the primary VSAM key. ESDS: Specifies that the VSAM data set is to be entry-sequenced. RRDS: Specifies that the VSAM data set is to be a relative record data set.

If any one of these parameters is specified, all other non-VSAM parameters on the OUTFIL statement are ignored, except for EXIT which overrides KSDS/ESDS/RRDS.

TOL — Specifies that SM2 is to tolerate a warning code from VSAM when opening a VSAM output data set. It is only valid for VSAM output. See Note 5 below.

REUSE — Specifies that you want to write over an existing non-empty VSAM data set defined with the REUSE attribute. For non-VSAM data sets this operand is ignored.

Note: The DISP parameter on the output DLBL card will override REUSE if it conflicts.

SPAN — Indicates that output is to consist of spanned variable-length EBCDIC records in a non-VSAM file. The RECORD statement must specify TYPE=V.
If a BLKSIZE parameter is also provided, the records will be spanned blocked. Otherwise they will be spanned unblocked, with a default block size of the maximum allowed for the output device.

OPEN= — This parameter only applies to tape output files.

RWD — Rewind tape before writing output records (default).

NORWD — Do not rewind tape before writing output records.

CLOSE= — This parameter only applies to tape output files. It indicates how the tape is to be treated at end of file.

RWD — Rewind tape (default).

NORWD — Do not rewind tape

UNLD — Rewind and unload tape.

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

BUFOFF=n — Only used when variable-length ASCII data has been specified in the RECORD statement. It specifies the block prefix at the front of each output block. The value of n may only be 0 or 4. Default is BUFOFF=0

OUTFIL STATEMENT PROGRAMMING NOTES

1. Parameters which precede EXIT in the OUTFIL statement are checked for validity and flagged. Parameters which follow EXIT are also checked and flagged for syntax errors but all values are ignored.

2. The presence or absence of the EXIT parameter affects the default symbolic unit names for the other files used by the program. See Figure 11 in Chapter 3 for details.

3. VSAM output data sets must be previously created with the VSAM access method services utility program. See the DOS/VS Data Management Guide or VSE/VSAM General Information for a discussion of VSAM organization, and the DOS/VS Utilities: Access Methods Services or Using VSE/VSAM Commands and Macros or the VSE/VSAM Documentation Subset manuals for how to define a VSAM data set.

4. Before writing variable-length records to a VSAM output file, SM2 removes the four-byte record descriptor word (RDW). Thus, those user-written routines at exit E35 which pass variable-length records back to SM2, do not need to be designed separately for handling VSAM and non-VSAM files.

5. If TOL is not specified for a VSAM output data set, a warning return code from VSAM will normally be recognized as an error by SM2; an error message will be printed and the program will terminate.

   By specifying TOL you can write output from SM2 to the data set without repairing it, and no error message will be printed. Conditions producing warning messages are, for example: 'NOT PROPERLY CLOSED' or 'SYSTEM TIME STAMPS OF DATA AND INDEX DO NOT MATCH'.

   A critical return code from VSAM will always cause SM2 to terminate, regardless of whether TOL has been specified.

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

OUTFIL STATEMENT EXAMPLES

Example 1

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

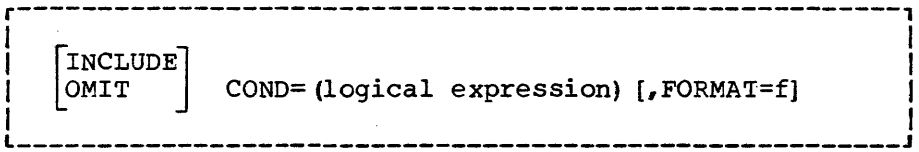
The block size for the output records is 300 bytes. The output tape will be rewound (by default) before writing begins and it will be rewound and unloaded at end of file. Data records are ASCII, as shown by the BUFOFF parameters, and each output block will have a block prefix of length 4.

Example 2

```
OUTFIL   EXIT
```

SM2 provides the output records, one at a time, to be handled by the user's routine at exit E35.

INCLUDE/OMIT Control Statement



An INCLUDE or OMIT statement is used if you do not want all of the input records to appear in the output file. You can achieve this in two ways: by defining the records which qualify for inclusion, by use of an INCLUDE statement; or by defining those which do not qualify, by use of an OMIT statement.

You make the definition by specifying a field in each record which is to be compared with another control value; the result of the comparison determines whether or not the record is included or omitted. The control value can be specified either as another field in the same record, or in the form of a constant. For example, you could compare the first six bytes of each record with its last six bytes, and omit from output all records where those fields are identical. Or you could compare a field with a specified date, and include in output only those records with a more recent date.

You must not supply both an INCLUDE and an OMIT statement to the same sort/merge run. If neither is supplied, all input records are included in the output file.

COND PARAMETER

The 'logical expression' of the COND parameter can be expanded into the following format:

$$\text{COND=(relational condition}_1\left[\left\{\begin{matrix}\text{,AND}\\\text{,OR}\end{matrix}\right\}\text{,relational condition}_2\ldots\right])$$

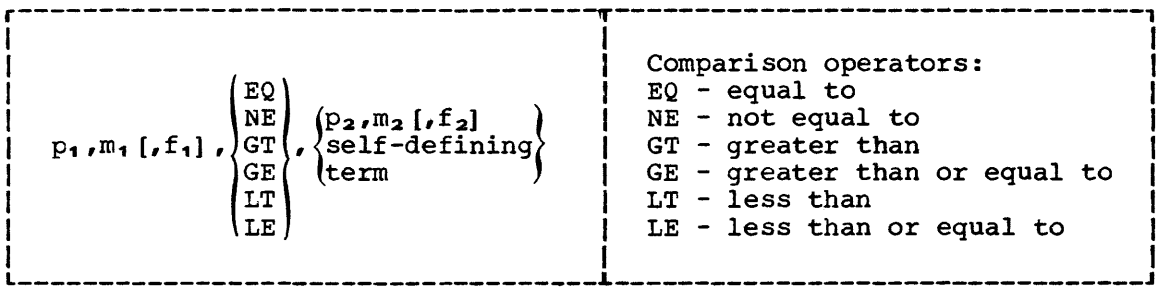
RELATIONAL CONDITION

The relational condition specifies a comparison to be performed. Its format is described below. Relational conditions can be logically combined, with AND or OR, to form a logical expression. If they are combined, the following rules apply:

1. Only one extra level of parentheses is allowed inside the syntax parentheses.
2. Expressions inside extra parentheses are always evaluated first.
3. 'AND' statements are evaluated before 'OR' statements.
4. The signs & (AND) and | (OR) may be used instead of the words.

RELATIONAL CONDITION FORMAT

The format of the relational condition is:



p_1, m_1, f_1

These parameters define the first data field.

- The variables p, m, and f have the same meaning as those described for the SORT and MERGE statements.
- Permissible field formats (f) are the same as for SORT and MERGE statements except that you may not specify FL (floating point); ZD fields can only be up to 18 bytes long, and PD fields can only be up to 16 bytes.
- If DATA=A is specified on the INPFIL statement you may only use formats AC, AST, and ASL.
- If all the data fields contain the same type of data, this value may be omitted, and you may use the optional FORMAT=f operand with the same abbreviations.

p_2, m_2, f_2

These parameters specify another field in the logical record, with which the first field (specified by p_1, m_1, f_1) is to be compared. Permissible comparisons between fields with different formats are shown in Figure 8.

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

Figure 8. Permissible Field-to-Field Comparisons for INCLUDE/OMIT

Self-Defining Term

A self-defining term is a constant with which a field is to be compared. It can be decimal, character, or hexadecimal. The different formats are shown in detail below. Permissible comparisons between types of field and types of constant are shown in Figure 9.

Decimal Number Format

```
[ ± ]  [nn...n]  [.nn...n]
                      ↑
                 decimal point
```

where n represents any decimal digit.

- Any number of digits can be specified except when the constant is to be compared with a field of FI format, when the constant may not be larger than 2,147,483,647 nor smaller than -2,147,483,648.
- The decimal point is not allowed in comparisons with fields of format FI, ZD, or PD.

Field format	Self-defining term		
	Decimal number	Character string	Hexadecimal string
BI		x	x
CH		x	x
ZD	x		
PD	x		
FI	x		
AC		x	x
ASL	x		
AST	x		
CST	x		
CSL	x		
CLO	x		
CTO	x		
AQ		x	x

Figure 9. Permissible Field-to-Constant Comparisons for INCLUDE/OMIT

- The decimal point, if specified, is counted in the length of the constant, and included in it as a character.

Examples of valid and invalid decimal self-defining terms are shown below.

Valid	Invalid	
15	++15	too many sign characters
+15	15+	sign in wrong place
-15	15.	invalid decimal point
131.588	1.5.0	too many decimal points
18000000	1,500	contains invalid character

Character String Format

```
C'xx...x'
```

where x represents any EBCDIC character.

If you wish to include a single apostrophe in the character string, you must specify a double apostrophe in your self-defining term. Thus:

Required: O'NEILL Specify: C'O''NEILL'

Examples of valid and invalid character string self-defining constants are shown below.

Valid	Invalid	
C'JOHN DOE INC'	C'''''	apostrophes not paired
C' $#'	'ABCDEF'	C identifier missing
C'+0.193'	'ABCDEF'C	C identifier in wrong place

Hexadecimal String Format

```
X'yy...yy'
```

where yy represents any pair of hexadecimal digits.

Examples of valid and invalid hexadecimal self-defining terms are shown below.

Valid	Invalid	
X'FF'	X'ABGD'	invalid hexadecimal digit
X'BF3C'	X'F1F'	incomplete pair of digits
X'AF050505'	'BF3C'	missing X identifier
	'BF3C'X	X identifier in wrong place

Padding and Truncation

In a field-to-field compare, the shorter field is padded. In a field-to-constant compare, the constant is padded or truncated to the length of the field.

Strings are truncated and padded on the right. The padding characters are:

X'00' for hexadecimal strings and BI fields
X'20' for ASCII character strings
X'40' for EBCDIC character strings

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

FORMAT OPERAND

FORMAT=f can only be used when all the fields in the whole COND expression have the same format. The permissible field formats are the same as for the SORT and MERGE statements except that you may not specify FL (floating point). If you have specified DATA=A on the INPFIL statement, you may only use formats AC, AST, and ASL.

INCLUDE/OMIT STATEMENT PROGRAMMING NOTES

1. The size of the routine generated by SM2 to handle the INCLUDE/OMIT function is dependent on how many fields are referenced, and what lengths and formats they have. The size of the routine must not exceed 4096 bytes.

2. Floating point fields (format FL) may not be referenced in INCLUDE or OMIT statements.

3. Any selection can be performed with either an INCLUDE or an OMIT statement.

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

5. Remember that if several compare statements are joined with a combination of AND and OR logical operators, the AND statement is evaluated first. The order of evaluation may be changed by adding one extra level of parentheses inside the COND expression.

   Figure 10 shows how SM2 will react to the result of a relational condition compare, depending on whether the statement is INCLUDE or OMIT, and whether the relational condition is followed by an 'AND' or 'OR' logical operator.

   When writing complex statements be sure the result will be what you want. The table in Figure 10 should help you.

Statement	Relational Condition	Program Action if Next Logical Operator is:	
	Compare	'AND'	'OR'
OMIT	True	Check next compare, or if last compare OMIT record	OMIT record
OMIT	False	INCLUDE record	Check next compare, or if last compare INCLUDE record
INCLUDE	True	Check next compare, or if last compare INCLUDE record	INCLUDE record
INCLUDE	False	OMIT record	Check next compare, or if last compare OMIT record

Figure 10. Logic Table for INCLUDE/OMIT Statement

INCLUDE/OMIT STATEMENT EXAMPLES

Example 1

```
OMIT COND=(1,10,CH,EQ,C'STOCKHOLM',&,(21,8,ZD,GT,+50000,|,
          31,4,CH,NE,C'HERR'))
```

This statement instructs SM2 to omit records in which:

- The first ten bytes contain 'STOCKHOLM' (the constant was padded with a blank).

AND

- The zoned-decimal number in bytes 21 to 28 is greater than 50,000 OR bytes 31 to 34 do not contain 'HERR'.

Note that the AND and OR operators can be written with the AND and OR signs; that parentheses are used to change the order of evaluation of the AND and the OR; and that only one extra level of parentheses is used inside the COND parameter.

Example 2

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

This statement instructs sort/merge to only include records in which:

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

OR

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

Note that all four fields have the same format.

ALTSEQ Control Statement

```
ALTSEQ  CODE=(fftt[,fftt...])
```

This statement specifies an alternative collating sequence to be used by SM2 when comparing control fields, including those specified in an INCLUDE or OMIT statement.

The statement is only valid for EBCDIC data. If the statement is omitted the standard collating sequences (EBCDIC or ASCII) will be used. If it is supplied when input and output are in ASCII form, it is ignored.

Operand	Description
CODE=	Parameter keyword
ff	Two hexadecimal digits specifying the character whose position is to be changed in the collating sequence.
tt	Two hexadecimal digits specifying the new position in the collating sequence the character is to occupy.

ALTSEQ STATEMENT PROGRAMMING NOTES

1. The moved character (ff) is considered equal to any character already occupying the position (tt).
2. A character can only be moved once.
3. The control field to which the alternative sequence is to apply must be described as format AQ in the SORT, MERGE, INCLUDE or OMIT statements that reference it.
4. Each group of hexadecimal digits must contain exactly four digits.

ALTSEQ STATEMENT EXAMPLES

Languages other than English often use characters not represented in English (with accents, for instance). The first two examples below show how such characters can be made to collate into their correct positions in the alphabet, by means of the ALTSEQ statement.

Example 1

The Swedish alphabet contains three letters not represented in the English alphabet. In data processing applications, the 'national' EBCDIC characters ($,#,@) are used to represent them. However, $, #, @ do not collate correctly for this purpose; the ALTSEQ statement below is necessary to specify that the characters appear after Z, in their correct order.

```
ALTSEQ  CODE=(5BEA,7BEB,7CEC)
```

Example 2

In the German alphabet the character Ä is collated with A, Ö is collated with O, and Ü is collated with U. The example statement shown below specifies the collating sequence.

```
ALTSEQ CODE=(49C1,B3D6,CBE4)
```

Example 3

The following example specifies that uppercase A is to collate before lowercase a, B before b and so on through to Zz.

```
ALTSEQ  CODE=(C180,C282,8283,C384,8385,C486,8487,C588,8589,
        C68A,868B,C78C,878D,C88E,888F,C990,8991,D192,9193,
        D294,9295,D396,9397,D498,9499,D59A,959B,D69C,969D,
        D79E,979F,D8A0,98A1,D9A2,99A3,E2A4,A2A5,E3A6,A3A7,
        E4A8,A4A9,E5AA,A5AB,E6AC,A6AD,E7AE,A7AF,E8B0,A8B1,
        E9B2,A9B3)
```

OUTREC Control Statement

```
OUTREC FIELDS=(p1,m1[,a1]...[,pn,mn [,an]])
```

The OUTREC control statement requests reformatting of the input records; that is, defines which parts of the input record are to be included in the output record, in what order they are to appear, and how they are to be aligned.

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

If the statement is not used, the output record is identical to the input record.

Operand	Description
FIELDS=	Parameter keyword.
p	First byte of a field in the input record which is to become part of the output record. The field can start anywhere within the record. Otherwise the rules for defining p are the same as for the SORT and MERGE statements. See the Programming Notes below for special rules for variable-length records.
m	Length of the field to be included in the output. It must include the sign if the data is signed, and must be a whole number of bytes.
a	Specifies the alignment (displacement) of the data in the output record, relative to the start of the output record. The permissible values are: H Halfword aligned. This means that the displacement of the field from the beginning of the record, in bytes, is a multiple of two. F Fullword aligned. The displacement is a multiple of four. D Doubleword aligned. The displacement is a multiple of eight. Alignment can be necessary if, for example, the data is to be used in a COBOL application program where COMPUTATIONAL items are aligned through the SYNCHRONIZED clause. If the parameter is omitted, no alignment is performed. Unused space preceding aligned fields will always be padded with binary zeros.

OUTREC STATEMENT PROGRAMMING NOTES

1. For variable-length records the first entry in the FIELDS parameter must specify or include the four byte RDW.

   If the first field in the data portion of the record is to appear in the output, the entry in the FIELDS parameter can specify both RDW and data field in one. Otherwise, the RDW must be specifically included in the output record.

2. SM2 sets the correct length value in the RDW even if you change the record length with your OUTREC statement.

3. The variable part of the input record (that part beyond the minimum record length) may be included in the output record as the last part. In this case, a value should be specified for p_n that is less than or equal to the minimum record length (l_4) plus 1 byte, and m_n and a_n should be omitted.

   Note that the output record must receive at least one byte of the fixed portion of the input record as well as the RDW, otherwise 'null' records containing only an RDW could appear in your output. SM2 checks your OUTREC statement for this possibility.

4. You need not specify l_2 and l_3 in the LENGTH parameter of the RECORD statement when using OUTREC, as SM2 fills in the correct values by default.

5. You must consider the effective record length (including padding, if any) of the reformatted record when specifying the BLKSIZE parameter on the OUTFIL statement.

6. Fields referenced in OUTREC statements may overlap, and may be control fields.

7. The ADDROUT option on the OPTION statement is ignored when an OUTREC statement has been specified.

8. If input is variable records the output will also be variable. This means that each record will be given an RDW by SM2, and applies even if the records are all the same length.

OUTREC STATEMENT EXAMPLES

Example 1

```
OUTREC FIELDS=(11,32)
```

This statement specifies that the output record should contain only bytes 11 to 42 of the input record. This statement can only be used with fixed-length input records because it does not include the first four bytes.

Example 2

```
OUTREC FIELDS=(1,4,11,32,D,101)
```

This statement is for variable-length records of minimum length 100 bytes, and specifies that the output record should contain an RDW plus bytes 11 to 42 of the input record (aligned on a doubleword boundary, relative to the start of the record) plus the entire variable portion of the input record.

Note that no extra comma is coded to indicate the omission of the first alignment parameter. If you do include an extra comma you will get a syntax error message, and the program will terminate.

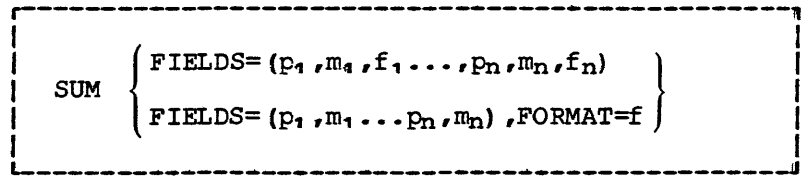
Example 3

```
OUTREC FIELDS=(1,42,D,101)
```

This statement is for variable-length records of minimum length 100 bytes, and specifies that the output record should contain an RDW plus the first 38 data bytes of the input record plus the entire variable portion of the input record.

The 'D' parameter has no effect, since the first field is always placed at the beginning of the output record.

SUM Control Statement



The SUM control statement designates numeric fields in the input record as summary fields. It specifies that whenever two records are found with equal control fields, the contents of their summary fields are to be added, the sum is to be placed in one of the records, and the other record is to be deleted.

Operand	Description
FIELDS	Parameter keyword.
p	First byte of a summary field (field to be added) relative to the beginning of the logical record. The general rules for defining p are the same as for the SORT and MERGE statements.
m	Length of the summary fields to be added. The value must include the sign, if signed data. See below for permissible length values.
f	Format of the data in the summary field, which can only be of the following types:

Code	Length
BI	2, 4, or 8 bytes
FI	2, 4, or 8 bytes
PD	1-16 bytes
ZD	1-18 bytes

FORMAT=f	Optional. Can be used when all the summary fields contain the same type of data. The values for f are listed above.

SUM STATEMENT PROGRAMMING NOTES

1. The size of the routine generated by SM2 to handle the SUM function is dependent on how many fields are referenced, and what lengths and formats they have. The size of the routine must not exceed 4096 bytes.
2. Summary fields must not be control fields, must not overlap control fields, and must not overlap each other.
3. Floating-point fields must not be summarized.
4. When records are summarized, the choice of which record is to receive the sum (and be retained), and which record is to be deleted, is unpredictable.

5. Fields other than summary fields remain unchanged, and are taken from the record which receives the sum.

6. If overflow occurs during summation, the records are left unsummarized (that is, the contents of the records are left undisturbed, and no record is deleted).

7. If both the SUM statement and the EQUALS parameter of the SORT statement are specified, the EQUALS parameter will be ignored.

8. If both the SUM statement and the ADDROUT option are specified, the ADDROUT option will be ignored.

SUM STATEMENT EXAMPLES

Example 1

```
SUM FIELDS=(41,8,ZD,49,4,FI)
```

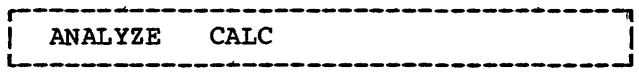
This statement designates an eight-byte zoned decimal field at byte 41, and a four-byte fixed integer field at byte 49, as summary fields.

Example 2

```
SUM FIELDS=(41,8,49,4),FORMAT=FI
```

This statement illustrates the use of the FORMAT operand. The statement designates two fixed integer fields, one 8 bytes long starting at byte 41, and the other 4 bytes long starting at byte 49.

ANALYZE Control Statement



This statement enables you to test your input job stream before running the sort or merge. It causes SM2 to terminate, without actually sorting or merging, after analyzing the input control stream and making its optimization calculations based on the information in the control statements.

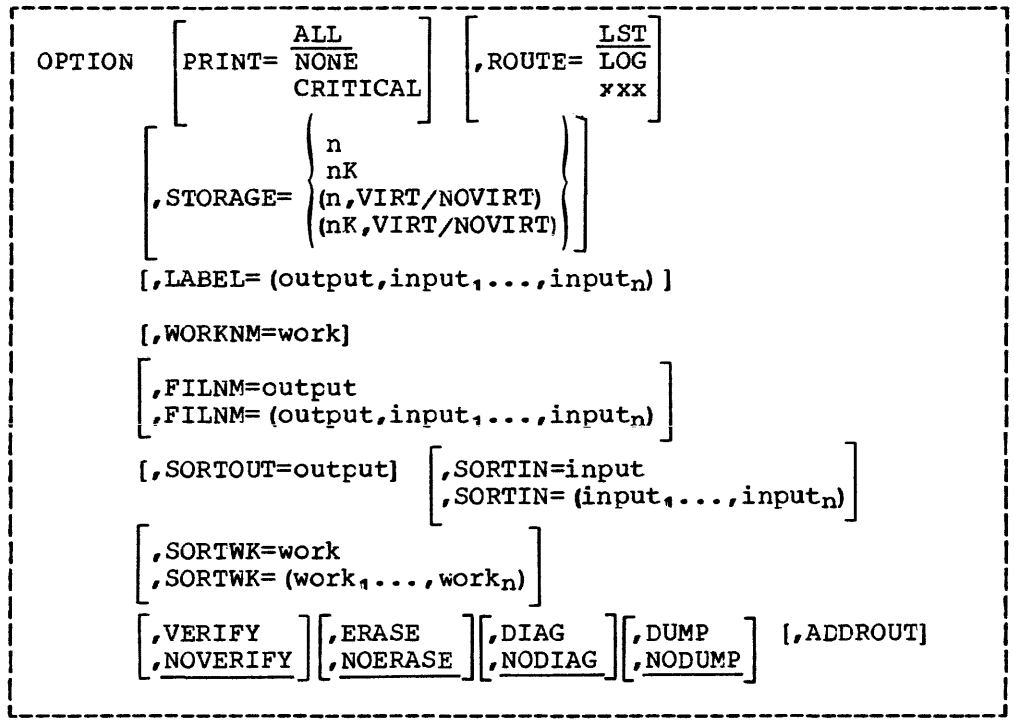
It causes the following options to be forced, regardless of what has been specified in the OPTION statement:

```
DIAG,NODUMP,ROUTE=LST|xxx,PRINT=ALL
```

It will thus produce all diagnostic messages, for example relating to the program's storage requirements. It then issues message 7C19I ANALYZE END, and terminates with a return code of 16.

If SM2 is invoked, and a value for ROUTE=xxx has been supplied (explicitly or by default), that value will be used instead of ROUTE=LST.

OPTION Control Statement



This statement specifies the options for the associated sort/merge program application. The values underlined are the standard defaults supplied with the program, but the defaults might have been changed for your installation.

Operand	Description
PRINT=	Option keyword. Its parameters specify which messages are to be printed by SM2. Default: PRINT=ALL, but can be changed after SM2 is installed.
ALL	All standard SM2 messages are to be printed, including error and end of job messages, various size calculations, and other informative messages.
NONE	No SM2 messages are to be printed. If this is specified, the DUMP option is automatically set to NODUMP.
CRITICAL	Only critical error messages are to be printed. That is error messages signaling conditions that can cause program termination.
ROUTE=	Option keyword. Its parameters specify where the program messages are to be routed. Default: ROUTE=LST, but can be changed after SM2 is installed.
LST	All SM2 messages are to be routed to the SYSLST file, and critical messages to the system console.
LOG	All SM2 messages are to be routed to the system console.

xxx — Only valid when SM2 is invoked from another program. Program messages are to be routed to SYSxxx, where xxx can be any valid SYS number between 000 and 221. Critical messages will also be sent to SYSLOG, but the system dump (if any) will go to SYSLST. If specified for an independent sort or merge this option will be ignored and the default (LST or LOG) taken instead.

STORAGE= — Option keyword.
Default: STORAGE=(n,NOVIRT), where 'n' is described in Chapter 6 (but can have been changed after SM2 was installed).

n
nK — The value n specifies the decimal number of bytes (or K bytes, where K=1024) of main storage to be available to SM2 and any associated user routines loaded by SM2.

A useful value for n can usually be found by subtracting the real storage requirements of the supervisor from the CPU main storage, and dividing the result by the number of partitions.

Minimum: the value specified for n must not be less than 32K. If it is, the action taken is as follows:
- if SM2 has been invoked from another program, it terminates immediately
- if SM2 is being run as an independent program, the value specified is ignored and 32K used instead.

Maximum: the value specified for n must not be greater than the smallest of the following:

1. The default storage value described in Chapter 6.
2. The difference between the SM2 load point and any calling program return address (in register 14) above the SM2 load point.
3. The difference between the SM2 load point and any preloaded user exit address which is above the SM2 load point.

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

If the final value for storage available to SM2 is less than 32K SM2 will terminate. If it is greater than 64K see programming note 2 in the section 'Option Statement Programming Notes'. For a given application, the minimum can be more than 32K. See the notes on storage in Chapter 6, 'Factors of Importance for Performance'.

VIRT — If VIRT is specified SM2 will not attempt to fix pages when running in virtual mode. VIRT is ignored if SM2 is run in real mode. See programming note 3 following this table.

NOVIRT — Instructs SM2 to fix pages when running in virtual mode.

LABEL={N / S / U}

Specifies the type of label associated with the output and input files, and must be in the order (output, input$_1$...,input$_n$).

The three label types are:

N - nonstandard labels (including user standard labels)
S - standard labels
U - unlabeled

Default: S (standard label). The positional subparameters may be replaced by a comma if the default value S is applicable.

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

If you specify N, for nonstandard label files (including standard labels with additional user headers or trailers), you must provide routines at the label checking exits to open and close the files, and process the labels, as described in Chapter 5, 'Modifying the Program'.

WORKNM=

Option keyword. The parameter specifies the first four letters of the name in the DLBL job control statement for the work file(s). The letters replace 'SORT' in the names SORTWK1, SORTWK2, and so on.
Default: WORKNM=SORT, but can be changed after SM2 is installed.

FILNM=

Option keyword. The parameter specifies the file name or names that are used in the TLBL and DLBL job control statements for the output and input files. The file names must be in the order (output,input$_1$...,input*).

Default: FILNM=(SORTOUT,SORTIN1-SORTIN9)

If the FILNM option is omitted, the default file names are assumed for all the files. See Figure 11 in Chapter 3.

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

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

SORTOUT=

Specifies the logical unit number of the output file. The value can be a maximum of three digits in the range 1-221.

Default: SORTOUT=001; this can be changed after SM2 is installed. Only needed for tape files under DOS/VSE.

SORTIN= Specifies the logical unit numbers of the input files. The values can be a maximum of three digits in the range 1-221, or a comma (for the default number). Only needed for tape files under DOS/VSE.

Default: SORTIN=(002...(,n+1)), where n is the number of input files specified. The default can be changed after SM2 is installed. (See Figure 11 in Chapter 3.)

SORTWK= Specifies the logical unit numbers of the work files. The values can be a maximum of three digits in the range 1-221, or a comma (for the default number). Not needed under DOS/VSE.

Default: SORTWK=((n+2)...,(n+m+1)), where n is the number of input files specified, and m the number of work files; this can be changed after SM2 is installed. (See Figure 11 in Chapter 3.)

VERIFY Specifies that when a direct access device is being used to store the output file, each block will be checked to ensure that it was written correctly. VERIFY is ignored for VSAM output files. Its use degrades SM2 performance. It prevents command chaining of the output CCWs.

NOVERIFY Output blocks will not be verified. This is the default, which can however be changed after SM2 is installed.

ERASE Specifies that the sort work files will be erased if they have been used. For CKD devices this is done by means of an End of File record written on every used track of the SORTWK area. Used parts of FBA areas are filled with zeros. Use of ERASE degrades SM2 performance.

NOERASE Work files will not be erased. This is the default, which can however be changed after SM2 is installed.

DIAG Specifies that special diagnostic messages are to be produced. This option is useful when tuning the performance of sort applications. (For further details see 'Using the DIAG Option' in Chapter 6.)

NODIAG Diagnostic messages will not be produced. This is the default, which can however be changed after SM2 is installed.

DUMP Specifies that a dump of main storage is to be made on SYSLST whenever SM2 terminates abnormally. If the STXIT function is available in the supervisor, SM2 will receive control on all types of abnormal condition. A dump of SM2 main storage will be produced, and a formatted dump of the communication area and trace table information will be provided. If STXIT is not available only errors detected by SM2 will result in program dumps.

Note: When running under VSE/Advanced Functions with EXEC REAL you must specify a SIZE parameter leaving sufficient real storage for system GETVIS functions.

NODUMP No dump is to be made if SM2 terminates abnormally. This is the default, which can however be changed after SM2 is installed.

ADDROUT Specifies that the final sort output should be only the direct-access addresses of the input records. These addresses can be used to retrieve the input records in sequence. The application must be a sort, not a merge.

Input can be either SAM files or VSAM files. SAM files must be on CKD direct-access devices; they must not be on more than one volume, and they must not be VSAM managed SAM files. However, ADDROUT can be used with VSAM managed SAM files if they are defined to sort/merge as VSAM files, that is, if VSAM is specified on the INPFIL card.

Input records must not be spanned. That is, the SPAN parameter must not be specified; nor must input consist of a VSAM KSDS defined with the SPANNED attribute. ADDROUT addresses produced in these cases would be meaningless.

If either an OUTREC or a SUM statement is provided, ADDROUT is ignored.

If INPFIL EXIT is specified with the ADDROUT option, SM2 will terminate.

If the output file is on tape the output block size must be 20 bytes or more. SM2 ensures that the last block contains at least 18 bytes by padding with blanks as necessary.

Output records are fixed-length, regardless of the type of input records.

The l_2 and l_3 values in the RECORD statement must, if specified, be as shown in the RECORD statement programming note 2. If they are not specified, the correct values are assumed by default.

SAM Files: For these files the addresses are ten-byte binary numbers in the form:

```
mbbcchhrdd
where
 m  = input file number (0-8)
      m=0 for records that have come from SORTIN1,
      m=1 from SORTIN2, and so on.
 bb = bin number (always 00)
 cc = cylinder number
 hh = head number
 r  = number of the record (block) on the
      input file track
 dd = displacement within block: 00 for unblocked
      fixed-length records, or the displacement, in bytes
      (relative to zero), of the record within the block.
      04 for unblocked variable-length records, or the
      displacement within the block.
```

For a SAM file the output block size must be a multiple of ten.

VSAM Files: For these files, the addresses are five-byte binary numbers in the form:

```
myyyy
where
   m  = SORTIN file number (1-9)
        m=1 for records that have come from SORTIN1,
        m=2 from SORTIN2, and so on.
 yyyy = relative byte address (RBA) for key sequenced
        (KSDS) or entry sequenced (ESDS) data sets, record
        number for relative record data sets (RRDS).
```

For a VSAM file the output block size must be a multiple of five.

OPTION STATEMENT PROGRAMMING NOTES

1. If you want to use the PRINT and/or ROUTE parameters you are advised to put the OPTION statement before all other SM2 control statements; and to put those parameters on the first card or line, not a continuation. This is because the defaults are assumed for those parameters until contrary information is read in.

2. When running in virtual mode, if neither the SIZE parameter or command nor the STORAGE option is specified the whole partition is reserved for SM2 by default. Since the page activity may be high in such an environment the storage used by SM2 will be kept to the largest of 64K bytes virtual storage or real size + 12K. However, if sort/merge cannot execute in that partition size it will be increased so that sort/merge is able to continue. If WORK=0 is specified in the SORT control statement all the available storage is used.

3. It may be necessary to specify VIRT to prevent interference with other jobs running simultaneously, or to allow a user-written routine to fix pages. Specifying VIRT may have an unfavourable effect on SM2 performance.

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

   If the checkpoint (CKPT) parameter has been specified in the SORT control statement the ERASE option will be ignored if sort terminates abnormally.

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

OPTION STATEMENT EXAMPLES

Example 1

```
OPTION  PRINT=CRITICAL,ROUTE=LOG,STORAGE=48K,LABEL=(,N,N)
```

This statement specifies that:

- Only critical SM2 messages will be produced and routed to the system control console.
- The virtual storage available to the program is 49,152 bytes.
- The output file is to have a standard label (by default) and the two input files have nonstandard labels.

Example 2

```
OPTION  STORAGE=64K,VERIFY,ERASE,SORTOUT=002,SORTIN=(003,004),
          SORTWK=(005,006)
```

This statement specifies that:

- All SM2 messages will be routed to the SYSLST file and be printed (by default), unless this default has been changed since SM2 was installed. Critical messages will also go to SYSLOG.
- The virtual storage available to the program is 64K bytes.
- When producing the output file, each block will be checked to ensure that it has been written correctly.
- The work files used by sort are to be erased on completion of sort.
- SYS002 is the logical unit number of the output file. SYS003 and SYS004 the logical unit numbers of the input files, and SYS005 and SYS006 the logical unit numbers of the work files.

Example 3

```
OPTION  FILNM=(SORT2,IN,,IN3),WORKNM=JOB2,SORTWK=10
```

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

Furthermore, if we assume that a multiextent direct access work file has been specified in the DLBL statement (code DA), only the logical unit number of the first extent is necessary (SYS010). The other logical unit numbers will be picked up by SM2 internally. In this case the WORK operand of the SORT statement must be specified or defaulted as WORK=DA.

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

```
OPTION  FILNM=(SORT2,IN,,IN3,,,,,,,JOB2),SORTWK=10
```

Example 4

```
OPTION  SORTOUT=5,SORTIN=(,,3),SORTWK=(010,11,12,,14,15)
```

Assume FILES=2 and WORK=3 (specified on SORT statement); then, using N and M from Figure 11 in Chapter 3 (N=2 and M=3), SM2 will allocate as follows:

SYS005 is the logical unit number of the output file.

SYS002 and SYS004 are the logical unit numbers of the input file. Two default specifications are made and the SYS numbers chosen by default would be SYS002 and SYS003 (SYS(N+1)=SYS(2+1)); but the latter is already defined and the next consecutive number is chosen - SYS004. The third parameter in the SORTIN operand is not used, as FILES=2.

SYS010, SYS011 and SYS012 are the logical unit numbers of the work files.

SYS006: default setting (would be the fourth workfile if it were used).
Note: SYS005 is already in use.

SYS006, SYS014 and SYS015 are not used in this application, since WORK=3.

This example shows how the values interact. The example may be understood as showing a sort which was set up to run with three input files and six work files, but which for this particular run has only two input files and three work files (note the assumption that FILES=2 and that WORK=3).

Chapter 3. Job Control Statements and Commands

Job control language (JCL) statements and commands are required to define a sort or merge job application and the input, output, and work files needed for that job. JCL is necessary for both independent and invoked sort/merge jobs.

The JCL that may be required for a sort or merge job is described briefly below. For a complete discussion of job control statements and commands and their format refer to DOS/VS System Control Statements or VSE/Advanced Functions System Control Statements.

JOB — Job name, etc.

ASSGN — ASSGN statements are required only if the devices to be used in an application have not previously been assigned to the appropriate symbolic names (SYS numbers) used in the SM2 application. Not required for VSAM or VSAM managed SAM files under DOS/VSE Release 2 with VSAM/VSE Release 2.

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

DLBL — A DLBL statement is required for every direct-access file.
- The BLKSIZE parameter must not be specified.
- If any file is on the same disk pack as another file used by the program the two files must have different file IDs.
- For VSAM, the file ID must specify the cluster name.

EXTENT — One EXTENT statement is required for each direct-access area to define the limits which will be used by the program. Extents defined may be Type 1 or Type 8 for input/output files and must be Type 1 for work files. The defined extents must include the SYS number of the device containing the extent. Not required for VSAM or VSAM managed SAM files under DOS/VSE Release 2 with VSAM/VSE Release 2, unless they are to be implicitly defined (managed files only).

EXEC — SORT is the required operand entry. The EXEC statement is followed by the SM2 program control statements. It should contain a SIZE parameter if storage size is not specified elsewhere (SIZE command, STORAGE option) or defaulted, and if a GETVIS area is required, that is:

- if SM2 is to use any VSAM files, and/or
- if EXEC REAL is specified, and SM2 will run under VSE/Advanced Functions.

LBLTYP — A LBLTYP statement is required if SM2 is invoked from another program and uses 'DA' work files or any standard labeled tape input/output files. It is not required for independent sort/merge jobs.

ALLOCR — Defines real storage for the partition when executing in real mode, thus defining the amount of storage that can be fixed.

ALLOC — Defines virtual partition size.

SIZE — Defines the GETVIS area of the partition when running under DOS/VSE.

Defining Files

All files that are to be used in a sort or merge application must be defined according to DOS/VS standards.

- The file SYS number must be assigned to a device address (ASSGN JCL statement) except with VSAM or VSAM managed SAM files under DOS/VSE Release 2.
- If the SYS number default values, as shown in Figure 11 (or changed for installation) are not to be used, the values must be specified in the appropriate SORTIN, SORTOUT, or SORTWK parameter of the OPTION statement (except under DOS/VSE for disk files).
- The file name must be included in the DLBL or TLBL JCL statement.
- If the default file names are not to be used the file names must be specified in the FILNM parameter of the OPTION statement.
- At least one EXTENT JCL statement is required for each DLBL statement, except with VSAM or previously defined VSAM managed SAM files under DOS/VSE Release 2.
- If output is to an FBA device, the DLBL statement should include the CISIZE parameter. If it does not, SM2 will use the minimum valid CI size that will hold the specified or defaulted OUTFIL block size.
- If CI size is specified for a work file it is ignored.
- DISP=(NEW,DELETE) should be specified on the DLBL card for VSAM managed SAM work files.
- VSAM files and VSAM managed SAM files should be previously defined using the VSAM Access Method Services program. Under DOS/VSE VSAM with the 'Space Management for SAM Feature' VSAM managed SAM output and work files only may be defined implicitly with the RECORDS and RECSIZE parameters on the DLBL card. This will of course add to the overall time taken by sort/merge. See Using VSE/VSAM Commands and Macros or VSE/VSAM Documentation Subset and Using the VSE/VSAM Space Management for SAM Feature, for more information. Implicitly defined managed output files will be defined with record format undefined (RECFM=UNDEF). These may be read using SAM as either fixed or variable as appropriate, but if they are read using VSAM the access method will present a whole block to the reading program which must then do its own deblocking. If this is not desired these files must be defined explicitly with RECFM=FB or VB as required.

File default names are shown in Figure 11. If a default SYS number is occupied by another file (as specified in the OPTION statement), SM2 will use the next free number.

INPUT FILE STATEMENTS

When the file name is of the form SORTINn, n can be any value from 1 to 9 for a sort or merge, depending on the number of input files. The file ID of the input file to be read must be included on each TLBL or DLBL statement. Where the input file is a direct-access multiextent file,

only the first EXTENT statement need contain the specified or defaulted SYS number for the input file. Other EXTENT statements may specify any valid SYS number.

OUTPUT FILE STATEMENTS

Multivolume and/or multiextent output on disk is accomplished by use of DOS/VS standards: one DLBL card is supplied for the entire file followed by one EXTENT card for each separate extent that the file occupies on the disk pack or packs. Where the output file is a direct-access multiextent file, only the first extent statement need contain the specified or defaulted SYS number. Other EXTENT statements may specify any valid SYS number.

WORK FILE STATEMENTS

Under DOS/VS Release 33 and 34

There are two methods of setting up multiple disk work files: they can be specified as multiextent, or multifile. Examples of the two methods as applied to the same sort are given in Appendix A. No appreciable performance advantage or disadvantage results from either method.

Notes

1. With FBA work files at least 64 blocks must be allocated per extent. If CISIZE is specified for the work extents, it is ignored.

2. If the WORKNM (or FILNM) parameter of the OPTION statement has been used to specify the first four characters for the work file name, this name must be used instead of the default name SORTWK1 in the DLBL statement.

3. Specifying a retention period of, say, one day prevents DOS/VS Job Control from regarding the work files as expired files, while at the same time requesting no longer retention than absolutely necessary. This is shown in the DLBL statements in the example below.

Multifile

In this method, the SYS numbers for work files are decided either by specifying them on the OPTION statement or by accepting the defaults shown in Figure 11 or those specified in the sort/merge default macro at installation time.

Supply one DLBL statement and one EXTENT statement for each work file your sort is to use. The DLBL filename entries must be either the default file names SORTWK1 through SORTWKn or the explicit work file names xxxxWK1 through xxxxWKn as specified in the WORKNM (or FILNM) parameter of the OPTION statement. n can be any number from 1 through 9. The 'code' parameter of the DLBL statements must be SD, explicitly or by default. In addition you must specify WORK=n (number of work files) on the SORT control statement. If you do not specify WORK=n, the default value DA will cause an I/O error on OPEN.

The filename numbers must start at 1 and be consecutive. For example:

```
// ASSGN SYS003,X'191'
// DLBL SORTWK1,,1
// EXTENT SYS003, ...
// ASSGN SYS004,X'192'
// DLBL SORTWK2,,1
// EXTENT SYS004, ...
// ASSGN SYS005,X'191'
// DLBL SORTWK3,,1
// EXTENT SYS005, ...
```

Multiextent

If you use this method of specifying your disk work file, you must specify (explicitly or by default) WORK=DA on the SORT statement.

You supply one DLBL statement, with default file name SORTWK1 or specified work file name, and 'code' DA. Follow this statement with from 1 to 9 EXTENT statements, which together specify the requisite amount of space.

Only the first EXTENT statement need contain the specified or defaulted SYS number for the work file.

The remaining EXTENT statements must be in consecutive ascending order of SYS number, but you may specify more than one extent on the same symbolic unit. Remember that for multivolume DAM files each different symbolic unit must be assigned to a separate physical device. Also that for multiextent DAM files all extents on one physical unit must have the same SYS number. For example:

```
// ASSGN SYS003,X'191'
// ASSGN SYS004,X'193'
// ASSGN SYS005,X'194'
// DLBL SORTWK1,,1,DA
// EXTENT SYS003, ...
// EXTENT SYS003, ...
// EXTENT SYS004, ...
// EXTENT SYS005, ...
```

Under DOS/VSE Advanced Functions

Sort work files are defined by SORTWKn DLBL cards and EXTENT cards. The files may be defined as SD (with one extent only), DA (with up to nine extents each) or VSAM managed SAM files if supported by the system (only the first extent is used). There may be up to nine of these files. The number is to be specified in the WORK parameter on the SORT statement.

Note: DA may not be used for files on FBA devices.

Users of VSAM managed SAM files are recommended to use the special file-ID prefix to cause a single extent to be allocated for the primary allocation, and to specify DISP=(,DELETE). See Using the VSE/VSAM Space Management for SAM Feature.

Any valid SYSNO which correctly defines the device used may be used for sort work files under DOS/VSE.

For example (WORK=3 on SORT card).

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

Note: The VSAM managed SAM file is defined implicitly. No EXTENT card is needed if the default model for the volume is used.

Use of Device	Filename	Symbolic Unit Names When:			
		Sort/Merge Reads Input and Writes Output	User Routine at E15 Reads Input	User Routines at E35 Writes Output	User Routines Read Input and Write Output
Output	SORTOUT	SYS001	SYS001		
Input	SORTIN1 . . . SORTIN9	SYS002 . . . SYS(n+1)		SYS001 SYS(n)	
Work	SORTWK1 . . . SORTWK9	SYS(n+2) . . . SYS(n+m+1)	SYS002 . . . SYS(m+1)	SYS(n+1) . . . SYS(n+m)	SYS001 . . . SYS(m)
CHECKPOINT	SORTCKP	SYS000	SYS000	SYS000	SYS000

n = the number of input files, as specified in the FILES parameter of the SORT or MERGE statement. Maximum value is 9.
m = the number of work files, as specified in the WORK parameter of the SORT statement. Maximum value is 9.

Figure 11. File Names and SYS Numbers Allocated by Default

Chapter 4. Executing the Program

This chapter describes how you can execute SM2 as an independent program and how you can invoke SM2 from within your own assembler language program. The SM2 program can also be invoked from programs written in COBOL, PL/I, and RPG II with the Auto-Report Feature. How to do this is described not in this manual but in the respective program user manuals or guides that are valid for your compiler. The JCL statements required to execute SM2 program are, however, the same regardless of how SM2 is initiated.

Independent Program

Figure 12 shows the job stream for an independent sort program whose input, output, and work files are all on disks. The input is a VSAM file containing fixed-length 80 byte records and the output is to a VSAM ESDS file.

Lines 1-11, 17 and 18 are DOS/VS JCL statements, and lines 12-16 are program control statements. If the same sort program were to be invoked from another program the JCL statements (2-10) would be needed unchanged, and the program control statements (12-16) would be needed in a similar form in the invoking program.

Detailed explanations of job streams are given in Appendix A.

```
 1   // JOB EXAMPLE STAND ALONE
 2   // ASSGN SYS001,X'160'                SORT OUTPUT
 3   // ASSGN SYS003,X'163'                SORT WORK
 4   // ASSGN SYS006,X'164'                SORT INPUT
 5   // DLBL INPUT,'ACCOUNTS',,VSAM
 6   // EXTENT SYS006,DISK01
 7   // DLBL SORTWK1,,0
 8   // EXTENT SYS003,,,,150,6
 9   // DLBL SORTOUT,'OUTPUT',0,VSAM
10   // EXTENT SYS001,DISK02
11   // EXEC SORT,SIZE=32K
12      OPTION ROUTE=LST,SORTIN=6,FILNM=(,INPUT)
13      SORT FIELDS=(1,30,CH,A),WORK=1
14      RECORD TYPE=F,LENGTH=80
15      INPFIL VSAM
16      OUTFIL ESDS
17   /*
18   /&
```

Figure 12. Job Stream for an Independent Sort Program

Initiating from an Assembler Program

The SM2 program can be initiated from an assembler program by issuing a LOAD followed by a CALL or ATTACH system macro instruction. The ATTACH macro should only be used if you are working in a multiprogramming environment and intend to subtask SM2.

In order to initiate execution of the program with a system macro instruction, you must:

- Write the required DOS/VS job control language statements.
- Write the sort/merge program control statements as operands of assembler DC instructions.
- Write a parameter list containing the addresses of the program control statement images and other information to be passed to SM2.

When SM2 is loaded by another program it will use all space from the load point to the partition's upper limit (or to the return address of the caller), unless its storage space is limited by the STORAGE option or the // EXEC SIZE= parameter. If SM2 is subtasked STORAGE must be specified.

Note: The sort/merge program is not reusable, which means that it must be loaded each time it is wanted.

INTERFACE REQUIREMENTS

The linkage conventions are standard. This means that when SM2 receives control, it expects general registers 13, 14, 15, and 1 to contain the following information:

Register 13: This must contain the address of a nine-doubleword area in which SM2 can save the contents of the user's registers. The user's registers will be restored when SM2 completes its processing; they will not be restored, however, when SM2 branches to user-written routines at an exit.

Register 14: This register must contain the address in the user's coding to which SM2 will return control upon completion.

Register 15: This register must contain the address at which SORT has its entry.

Register 1: This register must contain the address of a parameter list, the format and contents of which are described below (unless SM2 is subtasked, see 'Register 2' below).

Register 2: If SM2 is subtasked, as described below, then the address of the parameter list must be in this register instead of Register 1.

SUBTASKING

SM2 can be subtasked by using the assembler macro ATTACH. For details see DOS/VS Supervisor and I/O Macros, GC33-5373 or VSE/Advanced Functions Macro Reference and VSE/Advanced Functions Macro User's Guide.

There are two things that must be done if subtasking is to be used:

1. The input, output, and work files must be allocated unique file names for each task that will or can be run concurrently. This is to prevent the different tasks from trying to use the same input, output, and work files. The WORKNM and FILNM parameters of the OPTION statement are used to allocate the file names.

2. The STORAGE parameter of the OPTION statement must be specified so that the program knows how much storage it may use. If it is not specified, sort will try to use the whole partition.

If the program is subtasked checkpoints cannot be taken by SM2.

When SM2 is subtasked overprinting of messages routed to SYSLST may occur if the main task or other subtasks are using the same printer file. SM2 provides a number of options that can be used to control this:

- Specify ROUTE=xxx in the OPTION statement image, thus routing SM2 messages to the device of your choice (SYSxxx). A system dump, if any, will still appear on SYSLST, and critical messages will also appear on SYSLOG. This parameter can also be set as a default after installation of SM2.

- Route SM2 messages to the console by specifying ROUTE=LOG

- Write only critical messages from SM2 by specifying PRINT=CRITICAL

- Write no SM2 message at all by specifying PRINT=NONE

- Suppress the special formatted dump of SM2 areas by requesting NODUMP.

If the parameter list pointer is in register 2, SM2 will issue DETACH on completion. If, however, SM2 is called from a subtask, register one (1) must point to the parameter list, as described above. The calling program must then issue DETACH for its own subtask.

PASSING PARAMETERS

The parameter list consists of a series of address constants pointing to control statement images and other parameters. Figure 13 shows how the list can be coded. Figure 14 gives a complete coding example.

No compatibility problems will result from applications using the fixed-length parameter list previously containing only ten address constants.

The Address List

The first ten addresses (from SORT or MERGE to the return code halfword for phase 3 routines) must always be supplied. The first two must contain valid addresses.

The remaining five addresses (from ALTSEQ to ANALYZE) are optional. If supplied they can be in any order.

```
PLIST   DC A(SORT or MERGE statement)                                  01
        DC A(RECORD statement)                                         02
        DC A(INPFIL statement)                                         03
        DC A(OUTFIL statement)                                         04
        DC A(OPTION statement)                                         05
        DC A(MODS statement)                                           06
        DC A(Branch table for phase 1 preloaded user routines)         07
        DC A(0) 4-byte address (not used by sort)                      08
        DC A(Branch table for phase 3 preloaded user routines)         09
        DC A(Return code halfword)                                     10
        DC A(ALTSEQ statement or AQTT constant)                        11
        DC A(OUTREC statement)                                         12
        DC A(SUM statement)                                            13
        DC A(INCLUDE or OMIT statement)                                14
        DC A(ANALYZE statement)                                        15
           .
           .
SORT    DC C'SORT FIELDS=(1,24,CH,A,46,4,CH,D),FILES=2,'               16
        DC C'WORK=4 '                                                  17
RCD     DC C'RECORD TYPE=F,LENGTH=80 '                                 18
           .
           .
```

Figure 13. How to Code Parameters and Control Statement Images

Any addresses that are not needed can be filled with zeros, as shown in line 8 of the coding in Figure 13.

Lines 1-6 and 11-15 in Figure 13 are the address constants of SM2's program control statement images.

Control Statement Images

Lines 16-18 show examples of control statement images. The images must be coded in the form shown. No extra blanks, continuation characters, card sequence numbers, or comments are allowed in these images.

Lines 16 and 17 show how a continuation line is coded. Note that there is no space between the comma and the apostrophe at the end of line 16.

User Routines at Program Exits

Lines 7 and 9 may contain the address constants of branch tables for the preloaded user routines of each SM2 phase. These routines may be part of the program that is initiating SM2. For more information on preloaded user routines see Chapter 5, 'Modifying the Program'.

Line 8 is not used by this program, but must be included for compatibility reasons.

Return Codes: Successful and Unsuccessful Termination

Line 10 is the address constant of a halfword that can be used by SM2 to return a code to the initiating program. If SM2 completes successfully it returns a code of 0.

If it is unsuccessful it returns a code of 16, and the job is then canceled, unless the supervisor has been generated with the AB=YES option, in which case any STXIT routine will get control. If more than one STXIT routine is present, the last encountered will be used; earlier ones will be canceled.

SM2 has an STXIT routine of its own, for use if the program was initiated with the DUMP option specified. You can also supply STXIT routines in the calling program and/or at one or more program exits. The STXIT routines are generally encountered in the following order: first, the one in your calling program; then SM2's; last, those in routines at program exits.

SM2's STXIT routine is used to produce a dump, with a formatted listing of the program's communications area and trace table. If you want this information you should take care not to include an STXIT routine at a program exit, which could cause SM2's STXIT routine to be canceled. Conversely, if you supply an STXIT routine in your calling program you should run SM2 with the NODUMP option, thus ensuring that SM2's STXIT will not get priority over yours.

Alternative Sequence

Line 11 may be used in one of two ways. It can contain either the address constant of the ALTSEQ control statement image or supply a pointer to your own alternative sequence translate table.

If the ALTSEQ control statement image is specified, SM2 will use this information to build a 256 byte translate table known as the AQ-table. The AQ-table is then used as the second operand in a TR (translate) instruction that is applied to those control fields with AQ format.

You also have the option of building your own AQ-table and passing it directly to SM2. To do this, you place in the ALTSEQ entry of the parameter list a pointer to a fullword containing the four characters AQTT. In the fullword following these four characters you place the address of your AQ-table (256 byte translate table). An example of the coding involved follows.

```
PLIST    DC A(SORT or MERGE statement)
         DC A (RECORD statement)
         .
         .
         DC A(Return code halfword)
         DC A(AQTT)                      ALTSEQ entry
         .
         .
AQTT     DC C'AQTT'                      AQTT constant
         DC A(AQTABLE)
         .
         .
AQTABLE  DC X'0001020304050607'          256-byte
         DC X'08090A0B0C0D0E0F'          user-built
                                         AQ-table
         .
         .
         DC X'F8F9FAFBFCFDFEFF'
```

SAMPLE CODING

Figure 14 is an example of code that could be used to initiate execution of SM2. It is a sorting example with two preloaded user routines active.

```
            LOAD     SORT,LOADLOC                          1
            LR       15,1                                  2
            LA       1,PARAM                               3
            LA       13,SAVAREA                            4
            BALR     14,15                                 5
* FOLLOWING STATEMENTS ARE EXECUTED UPON COMPLETION OF SORT
            CLC      RETURN(2),=H'0'                       6
            BNE      SORTERR                               7
            --
SORTERR     --                                             8
*
PARAM       DC       A(SORT)                               9
            DC       A(RCD)                                10
            DC       A(INPFL)                              11
            DC       A(OUTFL)                              12
            DC       A(0)                                  13
            DC       A(MOD)                                14
            DC       A(E11)                                15
            DC       A(0)                                  16
            DC       A(E31)                                17
            DC       A(RETURN)                             18
            DC       A(ALTSEQ)                             19
            DC       A(SUM)                                20
            DC       A(OMIT)                               21
            DC       A(0)                                  22
SORT        DC       C'SORT FIELDS=(10,5,CH,A),WORK=5 '    23
RCD         DC       C'RECORD TYPE=F,LENGTH=80 '           24
INPFL       DC       C'INPFIL EXIT '                       25
OUTFL       DC       C'OUTFIL BLKSIZE=320,OPEN=NORWD '     26
MOD         DC       C'MODS PH1=(,,E15),PH3=(,,E35) '      27
SUM         DC       C'SUM FIELDS=(20,4,ZD) '              28
ALTSEQ      DC       C'ALTSEQ CODE=(5BEA,7BEB,7CEC) '      29
OMIT        DC       C'OMIT COND=(1,1,CH,EQ,C'*') '        30
SAVAREA     DS       9D                                    31
RETURN      DC       H'0'                                  32
            LTORG                                          33
*
* PHASE 1 BRANCH TABLE
            USING    E11,15                                34
E11         DC       A(0)                                  35
E15         B        INPUT                                 36
E17         DC       A(0)                                  37
E18         DC       A(0)                                  38
* PROGRAMMER'S PHASE 1 PROCESSING ROUTINES FOLLOW
INPUT       SAVE     (14,12)                               39
            --
            RETURN   (14,12)                               40
*
* PHASE 3 BRANCH TABLE
            USING    E31,15                                41
E31         DC       A(0)                                  42
E32         DC       A(0)                                  43
E35         B        OUTPUT                                44
E37         DC       A(0)                                  45
E38         DC       A(0)                                  46
E39         DC       A(0)                                  47
* PROGRAMMER'S PHASE 3 PROCESSING ROUTINES FOLLOW
OUTPUT      SAVE     (14,12)                               48
            --
            RETURN   (14,12)                               49
            LTORG                                          50
LOADLOC     DC       D'0'                                  51
            END
```

Figure 14. Sample Coding to Initiate the Program

1. This instruction loads the first phase of SM2. It is loaded on the doubleword boundary at LOADLOC (LOADLOC is defined by instruction no. 51).

2. The address of SM2's entry point is placed in register 15.

3. Register 1 is loaded with the address of the parameter list. This list is defined below by instructions 9-22.

4. This instruction loads register 13 with the user's save area address.

5. This instruction loads register 14 with the user's return address, and gives control to SM2. After execution of the sort, control returns to the next instruction.

6. Before returning control to the user, SM2 places a code in the halfword named RETURN. This instruction tests the code for successful sort completion.

7. If the return code is not zero, control goes to the user's error routine.

8. Instructions begin here to process a nonzero return code upon completion of the sort.

9-14. The parameter list begins at instruction no. 9. The first six address constants point to sort/merge control statement images that are defined below. Note that the constant at instruction no. 13, which usually contains the address of the OPTION statement image, contains zeros. In this situation SM2 uses default values for the OPTION statement parameters.

15-17 These constants point to the required branch tables. No. 16 must always contain zeros, since SM2 has no Phase 2 exits; space is provided for a Phase 2 branch table for reasons of compatibility.

18. This is the address of a halfword in which SM2 can place a return code. The halfword is defined by instruction no. 32.

19-22. These four address constants point to sort/merge control statement images. One (at instruction 22) contains zero, and will be ignored; it could have been omitted. The SUM, ALTSEQ, and OMIT functions are defined in instructions 28-30.

23-30. These instructions define the control statement images. Note that on the MODS statement no entries are required for the phase name and the address/length parameters, since the user routines are preloaded.

31. This instruction defines an area in which SM2 can save the contents of the user's registers.

32. This halfword is set aside for the return code from SM2.

33. All literals generated by previous coding are to be collected here.

34. This instruction establishes addressability for the Phase 1 branch table and the processing routines that follow the table.

35. This is the first instruction in the branch tables for Phase 1, the internal sort phase. Since no user routine is provided for Exit E11, SM2 will never give control to this instruction. Each unused branch table entry must be replaced by a four-byte displacement.

36. This instruction is a branch to the user routine at exit E15. INPUT is the label of the entry point for this routine.

37-38. Exits E17 and E18 are not used.

39. The user routine at exit E15 follows the Phase 1 branch table, and the first instruction will save registers that are used in the routine.

40. This instruction will restore registers to their status upon entry into the user routine and will return control to SM2.

41. This instruction establishes addressability for the Phase 3 branch table and the processing routines that follow the table.

42-43. These are the first instructions of the Phase 3 branch table; no user routines are provided for exits E31 and E32.

44. This instruction is a branch to the user routine at exit E35. OUTPUT is the label of the entry point for this routine.

45-47. Exits E37, E38 and E39 are not used.

48. The user routine for output follows the Phase 3 branch table, and the first instruction should save registers that are used in the routine.

49. This instruction will restore registers to their status upon entry into the user routine and will return control to SM2.

50. All literals generated by previous coding are to be collected here.

51. SM2 will be loaded here on a double-word boundary.

Note: If you use logical IOCS in your program, and you assemble the logic modules with your program, SM2 can be loaded over these modules. You must therefore use the linkage editor to ensure that the logic module CSECT is loaded before your program instead of after it. For example:

```
PHASE    USERPROG,S
INCLUDE  LOGICMOD,(logmcd CSECT name)
INCLUDE  USERPROG,(user CSECT)
```

For more information see under 'Linkage Editor' in DOS/VS System Control Statements or VSE/Advanced Functions System Control Statements.

Chapter 5. Modifying the Program

SM2 allows you to incorporate routines you have written into the main flow of a sort or merge job. There are several fixed points in the code, called program exits, at which control can be handed to your routines.

Figure 15 shows where the exit points are located, and Figure 16 summarizes what can be done at each exit point.

How the Program is Organized

As shown in Figure 15, the program is in four phases. All phases are usually executed for a sort, but only the first and last for a merge.

The exit names are in the form Exy, where x is the number of the phase, and y is the number of the exit within that phase.

PHASE 0: INITIALIZATION

There are no exits in Phase 0, which essentially collects, checks, and stores the information supplied by control statements.

PHASE 1: SORT

This phase, which is not used for a merge, reads the input files and sorts them into sequences, or strings. If the whole of the input can be contained in main storage there will be only one string, and the records will all be in the correct order. The sort is therefore complete, and all that remains to be done is to write the output file.

In most cases, however, there will be too many input records to fit into main storage, and work files have to be used. Record strings are built up in main storage, and written out to the disk areas specified as work storage.

There are four program exits in Phase 1: E11, E15, E17, and E18.

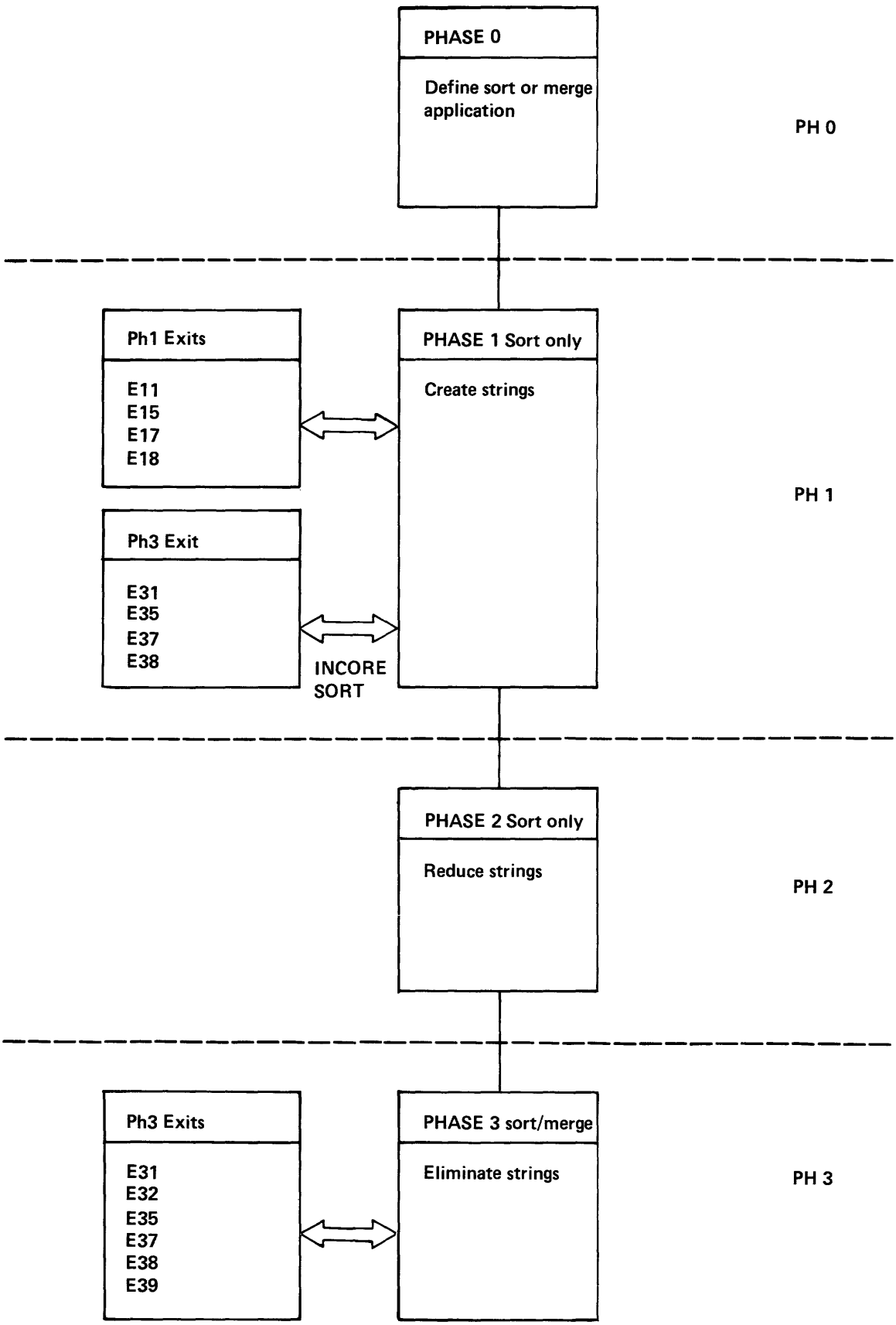


Figure 15. Overview of Program Flow and Exits

USE FOR EXITS	PHASE 1				PHASE 3					
	E11	E15	E17	E18	E31	E32	E35	E37	E38	E39
Take checkpoints					x[1]					
Process labels	x		x		x			x		
Open files	x				x					
Close files	x		x		x			x		
Supply password list for VSAM files				x					x	x
Supply VSAM exit list				x					x	x
Read input to a sort		x								
Count input records		x								
Insert/delete records		x					x			
Lengthen/shorten records		x[3]					x			
Modify record data		x[3]				x[3]	x			
Read input to a merge						x[2]				
Summarize record							x			
Substitute records in a merge						x				
Write output							x			

[1]Only activated if CKPT is specified on SORT statement
[2]Only when INPFIL EXIT is specified
[3]If control field lengths are changed, they must match those given in the SORT or MERGE statement

Figure 16. Uses for Program Exits

PHASE 2: MERGE STRINGS

If strings have been written to work storage in Phase 1, then Phase 2 is used to merge strings together until their number is such that they can all be merged in one pass by Phase 3, the final merge.

This phase has no program exits.

PHASE 3: FINAL MERGE

This is the phase which, after one merge pass, produces the output file. For a merge, the program reads the input files and merges them; and for a sort, it reads the work files and merges them.

There are six exits in Phase 3: E31, E32, E35, E37, E38, and E39.

Uses of Program Exits

The uses of the exits are summarized in Figure 16, and described in more detail in the sections which follow. There are four main ways in which they can be used: to handle nonstandard labels; to checkpoint; to manipulate input or output records; and to modify VSAM processing.

Your routines for use at the exits must be coded in assembler language. Some examples are given at the end of this chapter.

You must supply a MODS control statement to the program if you want your exit routine to be invoked. Chapter 2 describes how to code the MODS statement.

COMPARISON WITH OTHER SORT/MERGE PROGRAMS

If you have already written routines for use with other IBM DOS or DOS/VS sort/merge programs, you may be able to use the same routines with SM2. Details are given in Appendix C.

Handling Input and Output File Labels

If your tapes or disks have standard labels, or if you use unlabeled tapes, you need do nothing about either label processing or opening and closing files. The work will be done by SM2, using the standard facilities of the operating system.

The program cannot, however, handle nonstandard labels, or 'user-standard' labels (standard labels with an extra header or trailer). You must process these labels, and open and close the files, at the appropriate program exits. Figure 17 summarizes what needs doing, and at which exits it should be done.

You should be familiar with the label processing procedures described in the DOS/VS Data Management Guide, and in the publication DOS/VS Tape Label Reference or DOS/VS DASD Label Reference or VSE/Advanced Functions Data Management Concepts and VSE/Advanced Functions Tape Labels or VSE/Advanced Functions DASD Labels as appropriate.

Figure 17 is equally applicable to tape and disk files, with the exception of remarks on the subject of trailer labels: only tape files can have trailer labels.

VSAM managed SAM files must have standard labels.

	Input (sort)	Input (merge)	Output
Open file, process header labels	E11	E31	E31
Process all trailers except the last	E11	E31	E31
Process last trailer, close file	E17	E37	E37

Figure 17. Which Exits to Use for File Label Handling

INPFIL OR OUTFIL EXIT SPECIFIED

The parameter EXIT on the INPFIL control statement is a promise to SM2 that you will take care of all input to the sort or merge - not just the individual records, but the files as well. In the same way, OUTFIL EXIT means you will take complete charge of output.

With INPFIL EXIT specified you must therefore use E15 (for sort) or E32 (for merge) to read the input file, and pass the records one at a time to SM2. For any application with OUTFIL EXIT you must use E35 to take output records one at a time from SM2 and write them to file. You can if you wish open and close the files at the same exits. However your routines will be simple to code, and more generalized, if you use the label handling exits instead, as shown in Figure 17.

Individual input records can be fed to SM2 at E15 for a sort or E32 for a merge regardless of whether you have specified EXIT on the INPFIL control statement.

Checkpointing

Only one checkpoint can be taken during a sort operation and none during a merge. If you want the sort to take a checkpoint you must specify the CKPT parameter in the SORT control statement. This will cause SM2 to take a standard checkpoint of main storage and work files at the beginning of phase 3.

If the SM2 program is subtasked the CKPT parameter will be ignored.

You can handle checkpointing yourself by specifying the CKPT parameter in the SORT statement and by supplying a checkpoint routine at exit E31. SM2 will set up checkpoint parameter list but take no checkpoints itself. You must take full responsibility for the checkpoints in your own routine.

SM2 will not take a checkpoint if exit E31 is specified for any reason.

Modifying, Deleting, and Inserting Records

There are three exits at which you can manipulate individual records: E15, for records to be sorted; E32, for records to be merged; and E35, for records to be written to the output file.

AT SORT INPUT (E15)

Routines at exit E15 receive control before records are processed by Phase 1. If you have specified the EXIT parameter in the INPFIL control statement then routines at E15 must take complete responsibility for all the sort input. See the heading 'INPFIL or OUTFIL Exit Specified' above. The INCLUDE/OMIT, SUM, and OUTREC function are performed after E15.

Modifying a Record: You can alter the contents of any field in a record and you can change the length of a logical record by adding or deleting fields after the last control field, as long as you comply with the record description supplied in the SORT/MERGE and RECORD statements.

If you do change the contents or alter the length of any record field you must ensure that the control fields in the changed record still match these control fields specified on the SORT statement.

If the length of a fixed-length record is changed, the modified length must be specified by the l_2 value in the RECORD control statement.

If the length of a variable-length record is changed such that either the maximum or the minimum record length is altered then the respective l_2 or l_3 value should be specified in the RECORD control statement.

Checking Record Length: A routine can be set up to check the maximum and minimum lengths of variable-length records at this exit. The program checks the record lengths at a later stage and an incorrect record length may cause the program to terminate.

Deleting Records: Any record that you do not want in the output file can be deleted. Doing this at input rather than output (E35) saves program time. However the INCLUDE/OMIT function may be used for this purpose, instead of a user routine.

Inserting Records: Records can be inserted at any time with a routine at the E15 exit.

AT MERGE INPUT (E32)

Exit E32 functions in two different ways, depending on whether or not you specify INPFIL EXIT.

When INPFIL EXIT is Not Specified: A routine at E32 can only modify the contents of a record, including control fields, but it must not be used to alter the length of a record.

Input records cannot be inserted or deleted, but may be substituted that is, the record the merge passes at E32 can be replaced by one of your records.

When INPFIL EXIT is Specified: Your routine has complete responsibility for reading records into the merge, i.e., for all operations on the input files: defining them, opening them, and processing their labels.

The contents of the input records can be modified and the record lengths can be altered. Records may also be deleted or inserted.

When the records are ready for merging, your routine passes them, one at a time, to merge; merge performs the necessary compare operations, writes a record on the output file (or passes the record to your routine at exit E35), and then returns to your routine at E32 to obtain the next record to be merged.

AT OUTPUT (E35)

Modifying a Record: You can add, modify or delete fields anywhere in the record.

Checking Record Length: A routine can be set up to check, for example, the maximum and minimum lengths of variable-length records.

Deleting Records: Any unwanted record can be deleted.

Inserting Records: Records can be inserted at will, but you have responsibility for inserting them in the correct sequence.

Processing VSAM Files

There are three exits which can be used in conjunction with VSAM files to supply passwords, or an exit list.

The exits are E18 for sort input, E38 for merge input, and E39 for output files.

PASSWORDS

Your password routine at E18 is entered only once, and must therefore supply all necessary passwords for sort input files. The same applies to E38, where you must supply all merge input passwords.

EXIT LISTS

You must construct your exit lists using the VSAM EXLST macro, and observe all conventions governing VSAM exit lists. Details are given in the DOS/VS Supervisor and I/O Macros manual.

VSAM exits are not entered for null files.

Relocatable Routines are Best

Main storage that follows your routines may not be available for use by SM2. For this reason, self-relocating and loader-relocatable routines are recommended. These subjects are discussed in the DOS/VS System Management Guide.

When you use relocatable routines, remember to specify their length in the MODS statement, as described in Chapter 2. SM2 can then use main storage efficiently (usually by placing your routines at the highest addresses in the partition).

Loading and Linking to User Routines

LOADING YOUR ROUTINES

SM2 will load your routines for you.

All routines for each phase must be treated as an entity and cataloged under a unique name in the core image library. This is the name you specify on the MODS statement.

When you invoke SM2 from another program by use of the LOAD macro, you have the option of loading your own routines. If you do so, you must inform SM2 of their location in the parameter list as well as supplying a MODS statement. The parameter list is explained in detail in Chapter 4.

PASSING CONTROL

Since all of your routines for each phase are loaded in a single module, SM2 cannot know the entry point of each routine. You have to provide this information in a branch table at the beginning of the module.

The format of the branch tables is shown in Figure 18. If there is any exit which is not used, it must still have an entry in the table, as in the example in Figure 18 of a branch table for Phase 1.

```
PH1   B    E11          *BRANCH TO ROUTINE CALLED E11
      DC   A(0)         *E15 IS NOT USED
      B    E17          *BRANCH TO ROUTINE CALLED E17
      B    E18          *BRANCH TO ROUTINE CALLED E18
```

```
Phase 1 Branch Table:

USING ENTRY1,15
ENTRY1   B    E11
         B    E15
         B    E17
         B    E18
* Programmer's Phase 1 Processing
* Routines Follow
```

```
Phase 3 Branch Table:

USING ENTRY3,15
ENTRY3   B    E31
         B    E32
         B    E35
         B    E37
         B    E38
         B    E39
* Programmer's Phase 3 Processing
* Routines Follow
```

Figure 18. Branch Tables for Program Exits

USE OF REGISTERS TO PASS INFORMATION

SM2 uses registers 1, 13, 14, and 15 in the standard way to pass information to your routines.

Register 13 contains the address of a save area nine doublewords long, in which you should save the contents of the registers; you must restore the registers before returning control to SM2.

Register 14 contains the return address. Your routine returns control by branching to this address.

Register 15 contains the address of your branch table. You can use this as a base register at the start of your program.

Register 1 contains a pointer to a list of addresses (parameter list), each pointing to an item of information. The contents of the parameter lists are different for each program exit, and are described below in the coding instructions given for each exit. When your routine needs to pass a return code back to SM2, the same parameter list conventions must be used. The valid return codes for each exit are also described in the coding instructions.

If you pass an invalid return code, message '7M09 RETURN CODE ERROR, Exx' will be issued and the program will terminate.

Figure 19 shows the general method used by SM2 to pass parameters at the different exits.

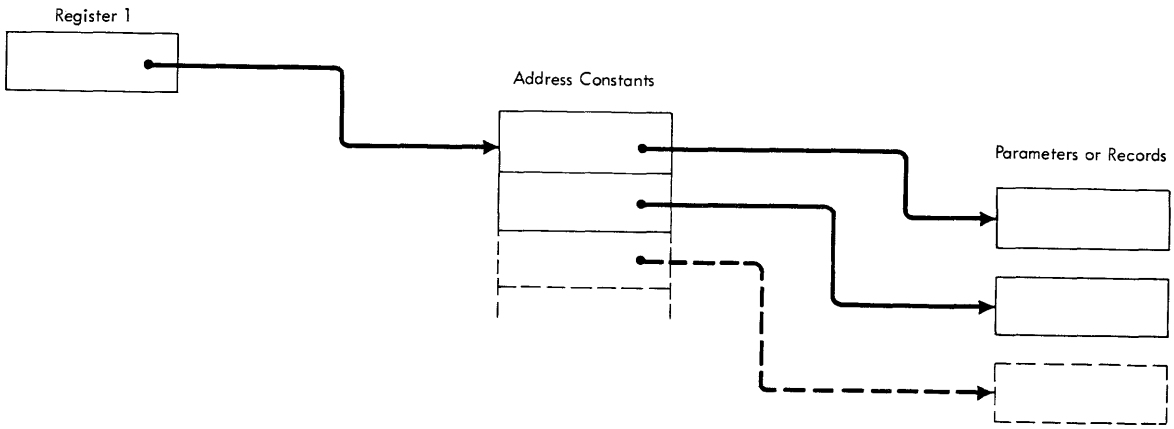


Figure 19. General Method for Passing Parameters

E11 Coding Instructions

The following notes apply tc both disk and tape files, with the exception of references to trailer labels. Trailer labels are only relevant to tape files, and all references to them should be ignored when disk files are involved.

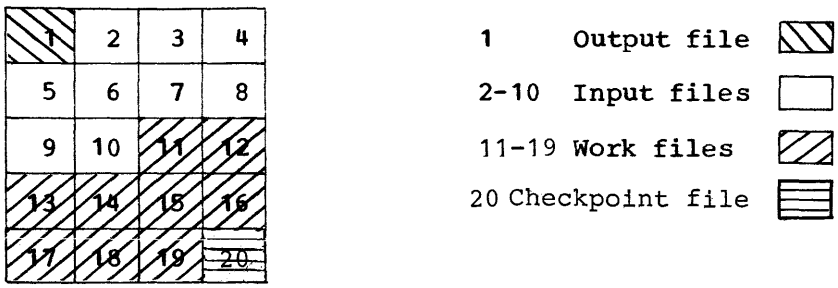
Parameter List

1.	Reserved	
2.	Reserved	
3.	Addr of previous volume unit	Logical unit no. of volume just processed (2 bytes) *
4.	Addr of next volume unit	Logical unit no. of next volume to process (2 bytes) *
5.	Addr of block count	Block count for trailer device
6.	Addr of SYS number table	SYS number table (see below)

* Only the SYS number is given, not the full CCB format.

SYS number Table

The SYS number table contains 20 one-byte entries, one for each device.



Each one-byte entry contains the symbolic unit number in binary used by the sort for the file in question. If there are fewer than the maximum number of files, unnecessary bytes contain zeros.

On First Entry

When your E11 routine first receives control, parameters 3 and 4 contain zeros. You must open the first volume of input and process its header label. You will find its symbolic unit number, in binary, in the second byte of the SYS number table pointed to by parameter 6. Parameter 5 is not used.

On Subsequent Entries

On subsequent entries if there is an address in parameter 3, process the trailer label for the volume on the unit pointed to, using the block count indicated in parameter 5. If the contents of parameters 3 and 4 are different (and parameter 3 is not 0), this is the last volume of the file, so you must also close the file. Note that you will not get control to close the last input file - this must be done at exit E17.

If there is an address in parameter 4, open the volume on the unit pointed to, and process its header labels.

EXAMPLES OF LABEL PROCESSING

The examples illustrated in Figure 20 show the label processing required of a user routine at exit E11 for various input configurations. The figure also shows the contents of parameters each time exit E11 receives control, and indicates the processing carried out at E17. Any references to trailer labels only apply to tape files and should be ignored when using disk files.

Example 1. One Multivolume Input File

This example illustrates the nonstandard or user standard label processing required for a single input file that is contained on three volumes:

1. Exit E11 is first activated to open the file and process the header label of the first volume. All label processing parameters contain zeros; only parameter 6 is of interest.

2. Exit E11 is given control the second time to process the trailer label of the first volume (for tape volumes only) and the header label of the second volume (for both tape and direct access volumes).

3. The third and last time exit E11 receives control, the user routine must again process a trailer label (tape only) and a header label (tape and direct access).

4. A routine at exit E17 must process the final tape trailer label and close the file.

Example 2. Multifile, Multivolume Input

In this example, the input consists of two files. The first is assigned to logical unit SYS002, and it extends over two volumes. The second file is on SYS003, and also extends over two volumes.

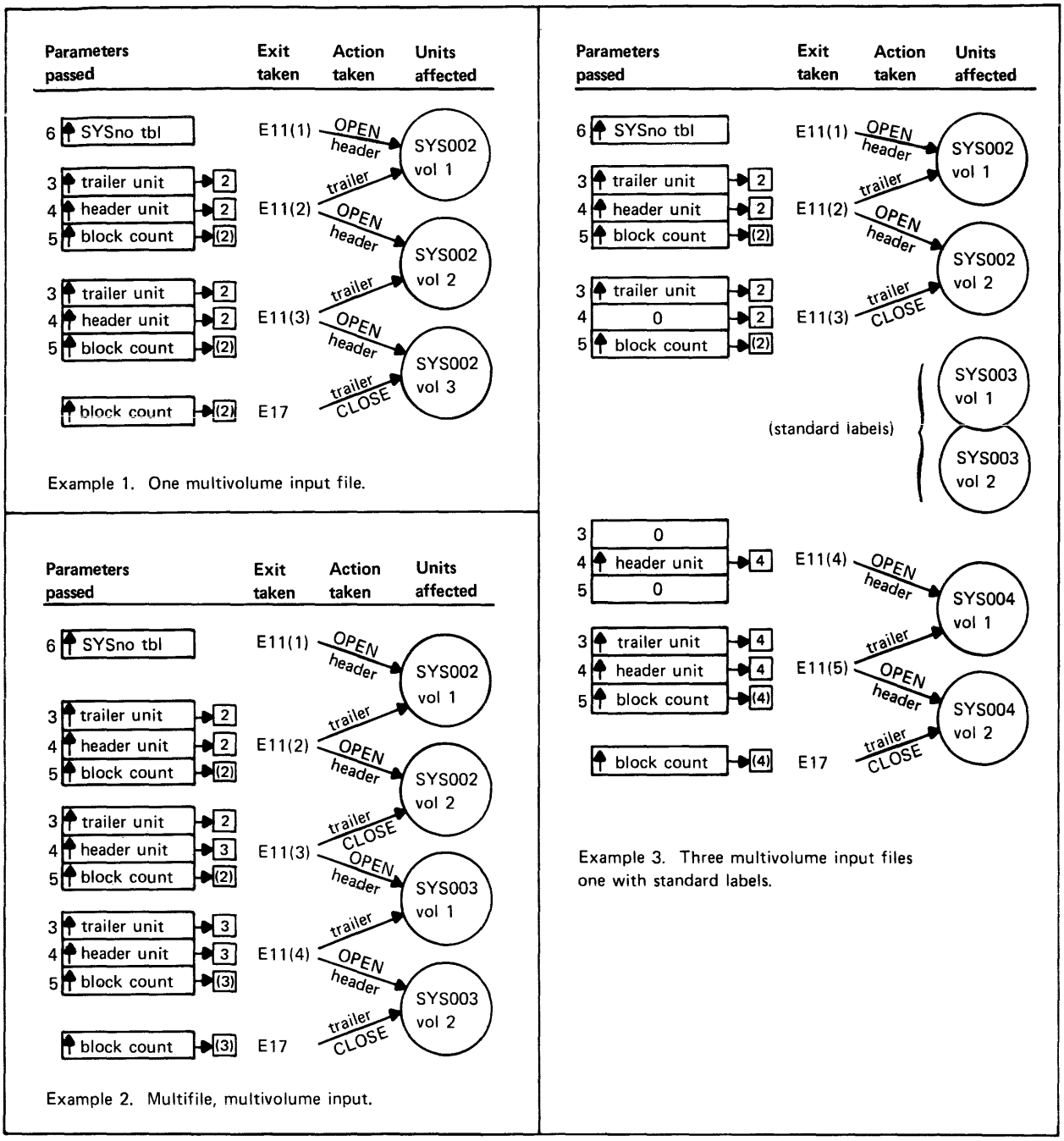
Exit E11 receives control three times to process the labels of the first file. On the third occasion it also opens the second file and processes its first header labels.

The fourth time it handles the second file's labels. A routine at exit E17 processes the final trailer label (tape only) and closes the second file.

Example 3. Three Multivolume Input Files, One With Standard Labels

In this example, the first and third files have nonstandard or user standard labels, while the second file has standard labels (the second file could also be an unlabeled tape file).

The processing required at E11 is exactly the same as described in Example 2, since the system handles standard labels completely.



Example 1. One multivolume input file.

Example 2. Multifile, multivolume input.

Example 3. Three multivolume input files one with standard labels.

Figure 20. Label Processing at E11 and E17

E15 Coding Instructions

Parameter List

1.	Addr of record (or zeros)[1]	Next record to process
2.	Addr of action word	Action word (1 word)
3.	Addr of record length vector	l_1 l_2 l_3 l_4 l_5 (5 words)
4.	Addr of record type	Record type switch (1 byte)

[1]Zeros when INPFIL EXIT specified, and at end of file.

Parameter 1 points to the record to be processed by your routine.

Parameter 2: The action word or return code tells SM2 what action to take when contrcl is returned to it. You must insert the appropriate code in the rightmost byte of the action word. Valid codes are:

```
X'00'  Process next record normally*
X'04'  Delete next record*
X'08'  Do not return to this exit
X'0C'  Insert a record
X'10'  Terminate sort
```

*Invalid when EXIT is specified on the INPFIL statement.

Parameter 3 gives the address of a 20-byte area containing the record's length parameters l_1 through l_5.

Parameter 4 gives the address of a byte containing the record type code. Bit 0 is on and bit 1 is off for fixed-length records and bit 1 is on and bit 0 is off for variable length records. Other bits may be used by sort for flags.

Procedure

Normally the sort will read the first input record and then pass control to you, with the record's address in parameter 1.

1. If the record is acceptable you return a code of X'00'.
2. If you want to delete it, you return a code of X'04'.
3. If you want to pass a record to sort which you have read in from a different file ('insert'), you put your record's address in parameter 1 and return a code of X'0C'. Next time, parameter 1 will have been restored to its former value.
4. If you want to change the record you must move it to a work area, change it, put its new address in parameter 1, and return a code of X'00'.

This process is repeated for every input record until you return a code of X'08' or X'10'.

Parameters 3 and 4 make it possible for you to write a generalized E15 routine, by giving you all the information you need about record type and length.

If you have specified INPFIL EXIT only step (3) is applicable: parameter 1 will always contain zeros, which you must replace with the address of the record you have read in. Note that return codes X'00' and X'04' are invalid with INPFIL EXIT specified.

Two examples of coding for routines at exit E15 are shown below.

Coding at E15: Example 1

This shows a routine that handles card input to sort.

```
EXIT15   CSECT
         PRINT NOGEN
         USING *,15
START    DC    A(0)              EXIT 11 UNUSED
         B     E15               ENTRY FOR EXIT 15
         DC    A(0)              EXIT E17 NOT USED
         DC    A(0)              EXIT E18 NOT USED
E15      SAVE  (14,12)           SAVE REGISTERS
         DROP  15
         USING START,11          SET UP NEW BASE REGISTER
         LR    11,15             REG 15 POINTS TO 1ST BYTE OF RTN
         LA    4,0(,1)           ADDR OF RECORD ADDR PARAMETER
         L     5,4(,1)           ADDR OF ACTION WORD
         LA    6,CARDIN          GET ADDRESS OF DTF
         CLI   FIRSTSW,X'00'     TEST IF FIRST TIME
         BNE   GET               NO, BYPASS OPEN
         OPENR (6)               OPEN THE FILE
         MVI   FIRSTSW,X'FF'     SET FIRST TIME SWITCH
GET      GET   (6)               READ A RECORD
         LA    7,RCDIN           GET ADDR OF RECORD JUST READ
         ST    7,0(,4)           STORE IN PARAMETER LIST
         LA    7,12              CODE FOR ACTION WORD = INSERT RCD
RETURN   ST    7,0(,5)           STORE IT IN PARAMETER LIST
         RETURN (14,12)          BACK TO SORT
EOF      LA    7,8               SET ACTION CODE TO 8 = END OF FILE
         CLOSER (6)              CLOSE THE FILE
         B     RETURN            BACK TO SORT
*
RCDIN    DC    CL80' '           INPUT AREA
FIRSTSW  DC    X'00'             FIRST TIME SWITCH
*
CARDIN   DTFCD DEVADDR=SYSIPT,IOAREA1=RCDIN,EOFADDR=EOF
IJCFZIZ  CDMOD
         END
```

Coding at E15: Example 2

This shows a routine which compares two bytes of each input record with a constant. If the compare is not high, the record is printed and deleted from the input.

```
EXIT15  CSECT
        PRINT NOGEN
        USING *,15              REG15 POINTS TO START OF EXIT
START   DC    A(0)              E11 NOT USED
        B     E15               BRANCH TO CODE FOR E15
        DC    A(0)              E17 NOT USED
        DC    A(0)              E18 NOT USED
*
        DROP  15                REG15 TO BE USED BY PUT RTN
        USING START,11          REG11 AS BASE REGISTER
E15     SAVE  (14,12)           SAVE REGISTERS
        LR    11,15             SET UP BASE REGISTER
        LA    4,PRINTER         GET ADDR OF PRINTER DTF
        LM    2,3,0(1)          LOAD ADDR OF RECORD & ACTION WORD
        LTR   2,2               TEST IF END OF FILE
        BZ    EOF               YES, BRANCH
*
        CLC   10(2,2),FIRST     CHECK FIRST CONTROL FIELD
        BH    NOACT             HIGH: SET RETURN CODE
*
OPEN    CLI   SW1,X'FF'         TEST IF FIRST TIME
        BE    PUT               IF NOT, BRANCH
        OPENR (4)               OPEN PRINTER FILE
        CNTRL (4),SK,1          SKIP TO CHANNEL 1
        MVI   SW1,X'FF'         BYPASS OPEN & CTL NEXT TIME
*
PUT     MVC   OUTAREA+1(120),0(2) MOVE RECORD TO OUTAREA, AND
        PUT   (4)                    PRINT IT
        LM    7,8,LINECNT       GET CURRENT AND MAX LINECOUNT
        LA    7,1(,7)           INCREMENT CURRENT LINE NUMBER
        ST    7,LINECNT         SAVE IT
        CR    7,8               TEST IF END OF PAGE
        BH    SKIP1             YES, GO SKIP TO CHANNEL 1
DELETE  LA    4,4               LOAD ACTION CODE 4 (DELETE)
        B     RETURN            RETURN TO SORT
*
NOACT   SR    4,4               LOAD ACTION CODE 0 (NO ACTION)
        B     RETURN            GO BACK TO SORT
*
SKIP1   CNTRL (4),SK,1          SKIP TO CHANNEL 1
        XC    LINECNT,LINECNT   CLEAR CURRENT LINE NUMBER
        B     DELETE            GO BACK
*
EOF     CLOSER (4)              CLOSE PRINTER FILE
        LA    4,8               LOAD ACTION CODE 8 (NO RETURN)
RETURN  ST    4,0(3)            STORE ACTION WORD
        RETURN (14,12)
*
LINECT  DC    F'0',F'50'        CURRENT AND MAX LINECOUNT
OUTAREA DC    CL121' '          OUTPUT AREA
FIRST   DC    C'22'             CONSTANT TO COMP. WITH CTL FLD
PRINTER DTFPR DEVADDR=SYSLST,IOAREA1=OUTAREA,CONTROL=YES
LJDFCZZZ PRMOD CONTROL=YES
SW1     DC     X'00'
        END
```

E17 Coding Instructions

Parameter List

1. Addr of block count — Block count for last volume of last input file

Procedure

Your routine at E17 receives control only once. It must then process the trailer label of the last volume of input, using the block count passed as a parameter, and close the last input file.

E18 Coding Instructions

Parameter List

1. Type indicator
2. Zeros
3. Addr of action word — Action word (1 word)

You must put a return code in the rightmost byte of the action word. Valid codes are:

X'00'	No reply
X'04'	Reply provided
X'08'	Do not return

Procedure

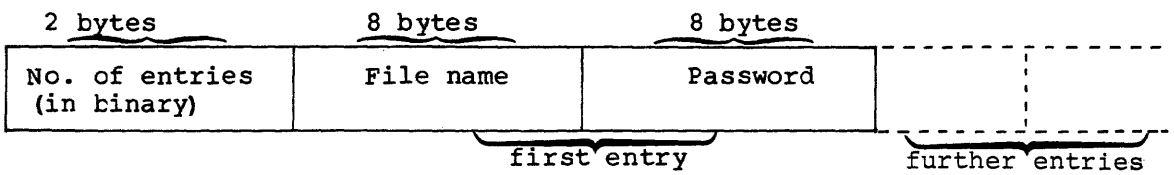
Your E18 routine is entered twice. The first time, parameter 1 contains C'PWD' (plus a padding byte). This means that passwords are requested. The second time it contains C'EXL', which is a request for an exit list.

If you have no reply, return with a code of X'00' in the action word.

Otherwise put the address of the password list or exit list (whichever has been requested) in parameter 2, and return with a code of X'04'.

Password List

The list must begin with a two-byte entry count, and continue with a sixteen-byte entry for each password-protected sort input file:



Exit List

The exit list must be built according to the rules laid down in DOS/VS Supervisor and I/O Macros or Using VSE/VSAM Commands and Macros (or VSE/VSAM Documentation Subset). All routines pointed to by the list must use standard VSAM linkage to return to VSAM. They must not use Register 13, which is in use by VSAM; instead they must provide their own save area.

For end-of-file processing do NOT use the EODAD exit, as the sort will not then be able to detect end-of-file. Instead supply a LERAD exit and test the FDBK code for X'04', which indicates EOF. Do not change the code, as SM2 also uses it.

E31 Coding Instructions

Parameter list

1. Reserved	
2. Addr of device list	Device List
3. Addr of previous volume unit	Logical unit no. of volume just processed (2 bytes) (CCB format)
4. Addr of next volume unit	Logical unit no. of next volume to process (2 bytes) (CCB format)
5. Addr of block count	Block count for trailer device (1 word)
6. Addr of SYSnumber table	SYSnumber table

Procedure

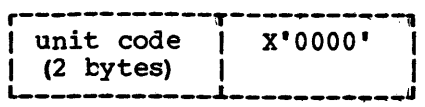
Your E31 routine will receive control at the beginning of phase 3 for both file handling and checkpointing. If only checkpointing is requested then the E31 exit is only entered once. If nonstandard label files are specified in the LABEL option then the E31 exit will be entered each time a nonstandard label input or output volume is required.

As with E11, you use parameters 3 and 4 to determine what action is needed. Either they will both contain all zeros, in which case this is the first entry; or they will both contain addresses. The addresses will be the same because there are no new files to be opened, only new volumes.

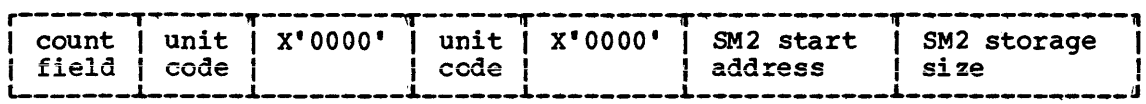
On First Entry

On first entry, parameters 3 and 4 both contain zero.

If you want to take a checkpoint, parameter 2 points to a checkpoint device list. The device list begins with a two-byte count field which contains the number of work file extents that the checkpoint device list will contain. The count field is followed by one four-byte entry for each sort work extent. The four-byte entries have the form:



Two four-byte fields at the end of the device list give the sort start address and the sort storage size. The checkpoint device list has the form:



For more details of checkpointing device lists see DOS/VS Supervisor and I/O Macros or the VSE/Advanced Functions Macro User's Guide.

If you are handling files, open all files with nonstandard labels (input and/or output), and process their header labels. You will find their SYSnumbers in the SYSnumber table pointed to by parameter 6: the first byte gives the number (in binary) of the output file, and the following nine bytes give the numbers of the input files. If you have fewer than nine input files the superfluous bytes contain zeros. See E11 coding instructions for more details of the SYSnumber table.

On Subsequent Entries

If parameter 3 contains an address, process the trailer of the volume pointed to, using the block count indicated in parameter 5. You will not be given control to close the files; this must be done at E37.

If parameter 4 contains an address, open the volume pointed to, and process its header.

The table below shows the actions that must be taken by the routines at exits E31 and E37 when the input to a merge consists of tape files:

SYS002, one volume, nonstandard labels
SYS003, standard labels
SYS004, two volumes, nonstandard labels

Output consists of one disk file:

SYS001, three volumes, nonstandard labels (user supplied labels).

Routine given control, parameters passed	SYS001			002	004	
	1	2	3	1	1	2
E31 (1st time) 3=0, 4=0	OPEN header			OPEN header	OPEN header	
E31 (2nd time) 3 ->001 4 ->001		OPEN header				
E31 (3rd time) 3 ->004 4 ->004					trailer	OPEN header
E31 (4th time) 3 ->001 4 ->001			OPEN header			
E37			CLOSE	trailer CLOSE		trailer CLOSE

Figure 21. Using E31 and E37 with a Merge

Example of Coding at E31 and E37

The following example shows a routine which opens the output file, writes the file label, and finally closes the file.

```
LABEXITS CSECT
         PRINT NOGEN
         USING *,15                REG 15 POINTS TO START OF EXIT
START    B     E31                 ENTRY POINT FOR EXIT E31
         DC    A(0)                EXIT E32 NOT USED
         DC    A(0)                EXIT E35 NOT USED
         B     E37                 ENTRY POINT FOR EXIT E37
         DC    A(0)                EXIT E38 NOT USED
         DC    A(0)                EXIT E39 NOT USED
*
         DROP  15
         USING START,11            USE REG 11 AS BASE REGISTER
*
***START OF ROUTINE FOR EXIT E31***
*
E31      SAVE  (14,12)             SAVE REGISTERS
         LR    11,15               SET UP BASE REGISTER
         LA    6,SORTOUT           GET ADDRESS OF DTF
         OPENR (6)                 OPEN OUTPUT FILE, WRITE LABELS
         RETURN (14,12)            RESTORE REGISTERS, BACK TO SORT
*
***START OF ROUTINE FOR EXIT E37***
*
E37      SAVE   (14,12)            SAVE REGISTERS
         LR     11,15              SET UP BASE REGISTER
         MVI    E37SW,X'FF'        SET E37 SWITCH FOR LABEL RTNE
         LA     6,SORTOUT          GET ADDRESS OF DTF
         CLOSER (6)                CLOSE THE FILE
         RETURN (14,12)
*
LAB      LA     0,UHL1             GET ADDRESS OF FIRST USER HEADER LAB
         CLI    FIRSTSW,X'FF'      TEST IF FIRST TIME
         BE     LAB1               NO, BRANCH
         MVI    FIRSTSW,X'FF'      SET FIRST TIME SWITCH
         LBRET  2                  LBRET 2 = MORE LABELS TO PROCESS
*
LAB1     CLI    E37SW,X'FF'        TEXT IF E37
         BE     LAB2               YES, TIME FOR TRAILER LABEL
         LA     0,UHL2             ADDRESS OF 2ND USER HEADER LABEL
         LBRET  1                  LBRET 1 = LAST LABEL PROCESSED
*
LAB2     LA     0,UTL1             LOAD ADDRESS OF TRAILER LABEL
         LBRET  1                  WRITE TRAILER LABEL - (LAST LABEL)
UHL1     DC     CL80'UHL1EXAMPLE OF A USER HEADER LABEL'
UHL2     DC     CL80'UHL2ANOTHER EXAMPLE OF A USER HEADER LABEL'
UTL1     DC     CL80'UTL1THIS IS AN EXAMPLE OF A USER TRAILER LABEL'
*
E37SW    DC     X'00'              USED IN LABEL RTNE TO IDENTIFY E37
FIRSTSW  DC     X'00'              1ST OR SEQUENCE ENTRY
*
SORTOUT  DTFPH  TYPEFLE=OUTPUT;LABADDR=LAB,DEVADDR=SYS001
*
         END
```

E32 Coding Instructions

Parameter List

1. Addr of next record (zeros when INPFIL EXIT specified)	Next record to process
2. Addr of input file number	File no. in hex code (1 word)
3. Addr of action word	Action word
4. Addr of record length vector	l_1 l_2 l_3 l_4 l_5 (5 words)
5. Addr of record type	Record type switch (1 byte)

The action word is only required when the INPFIL EXIT parameter is specified. You must put the return code in the rightmost byte of the action word. The valid codes are:

X'08' No more records to come from a specified file
X'0C' New record inserted
X'10' Terminate merge

Parameter 4 gives the address of a 20-byte area containing the record's length parameters l_1 through l_5.

Parameter 5 gives the address of a byte containing the record type code. The codes are X'80' for fixed-length records and X'40' for variable-length records.

Procedure Without INPFIL EXIT

When INPFIL EXIT is not specified only parameters 1, 4, and 5 are passed. The merge reads the first record from the first input file and then passes control to you, with the record's address in parameter 1. You can accept the record; or you can substitute a new one of the same length by changing parameter 1 to point to the new record. You then return control to the merge. No return codes are passed back. This process is repeated until the input is exhausted.

Procedure With INPFIL EXIT

However, if you have specified INPFIL EXIT the procedure is different. Then, when you first receive control, parameter 1 contains zeros and parameter 2 contains a pointer to a word containing a hexadecimal code in the rightmost byte which specifies from which input file the next record should be obtained. The codes are:

X'00'	File 1	X'14'	File 6
X'04'	File 2	X'18'	File 7
X'08'	File 3	X'1C'	File 8
X'0C'	File 4	X'20'	File 9
X'10'	File 5		

You should then:

1. Open all input files and do any necessary label processing.

2. Read the first block of records from the first file.

3. Put the address of the first record in parameter 1.

4. Put X'0C' ('new record inserted') in the action word pointed to by parameter 3.

5. Return control to the merge.

On each subsequent entry you pass a record to the merge in the same way, from the file requested in parameter 2.

When you have no more input on the requested file, you close the file (processing labels as necessary) and return with zeros in parameter 1 and X'08' in the action word. SM2 will then request input from a different file, until you have returned a code of 'X08' for each of the input files. If you need to terminate the merge before end of input (abnormal termination) you return a code of X'10'.

An example of a routine for use at exit E32 when the INPFIL EXIT is specified is shown below.

```
PH3RTN  CSECT
        PRINT NOGEN
        USING *,15
START   DC    A(0)              E31 NOT USED
        B     E32               ENTRY POINT FOR EXIT E32
        DC    A(0)              E35 NOT USED
        DC    A(0)              E37 NOT USED
        DC    A(0)              E38 NOT USED
        DC    A(0)              E39 NOT USED
        DROP  15
        USING START,12
E32     SAVE  (14,12)           SAVE REGISTERS
        LR    12,15             LOAD BASE REGISTER
        ST    13,SAVE13
        LM    2,3,4(1)          LOAD PARAMETERS
        LR    4,1               SAVE PARAMETER POINTER
*
*       REGISTER 2 NOW POINTS TO FILE NUMBER INDICATOR
*       REGISTER 3 NOW POINTS TO ACTION WORD
*       REGISTER 4 NOW POINTS TO PARAMETER LIST
*
        CLI   FIRST,X'FF'       IS THIS THE FIRST TIME?
        BE    FILENO            NO, BRANCH
        OPEN  MASTER,WEEK       YES, OPEN FILES
        MVI   FIRST,'X'FF'      SET 'FILES OPEN' INDICATOR
*
FILENO  CLI   3(2),X'00'         FILE 1?
        BE    GETMAST           YES, READ FROM MASTER FILE
        CLI   3(2),X'04'        FILE 2?
        BE    GETWEEK           YES, READ FROM WEEKLY UPDATE FILE
*       IF NO BRANCH WAS TAKEN, THIS IS AN ERROR, FORCE DUMP
ERROR   DUMP
*
GETMAST CLI   MASTOUT,X'FF'     FILE ALREADY CLOSED?
        BE    ERROR             YES, ERROR
        GET   MASTER            ELSE, READ A RECORD
        B     INSERT            GO TO SEND IT TO MERGE
*
GETWEEK CLI   WEEKOUT,X'FF'     FILE ALREADY CLOSED?
        BE    ERROR             YES, ERROR
        GET   WEEK              ELSE GET RECORD
*
```

```
INSERT   ST     5,0(,4)              STORE ADDRESS OF RECORD
         MVC    3(1,3),ACCEPT        SET ACTION WORD
         B      RETURN               RETURN TO MERGE
*
ENDMAST  MVC    3(1,3),END           SET ACTION WORD
         CLOSE  MASTER               CLOSE MASTER FILE
         MVI    MASTOUT,X'FF'        SET FILE CLOSED INDICATOR
         B      RETURN               RETURN TO MERGE
ENDWEEK  MVC    3(1,3),END           SET ACTION WORD
         CLOSE  WEEK                 CLOSE WEEKLY UPDATE FILE
         MVI    WEEKOUT,X'FF'        SET FILE CLOSED INDICATOR
*
RETURN   L      13,SAVE13            RESTORE REG 13
         RETURN  (14,12)             RESTORE REGISTERS AND RETURN TO MERGE
*
*        RETURN CODES FOR MERGE
ACCEPT   DC     X'0C'                INSERT RECORD (ACCEPT)
END      DC     X'08'                END OF FILE
*
MASTOUT  DC     X'00'                MASTER CLOSED INDICATOR
WEEKOUT  DC     X'00'                WEEK CLOSED INDICATOR
FIRST    DC     X'00'                FIRST TIME INDICATOR
SAVE13   DC     F'0'
INBUFM   DS     100F                 INPUT BUFFER FOR MASTER
INBUFW   DS     100F                 INPUT BUFFER FOR WEEK
*
MASTER   DTFSD  BLKSIZE=400,DEVADDR=SYS020,RECFORM=FIXBLK,
                RECSIZE=80,IOREG=(5),ERROPT=SKIP,DEVICE=3340,
                IOAREA1=INBUFM,EOFADDR=ENDMAST
WEEK     DTFSD  BLKSIZE=400,DEVADDR=SYS021,RECFORM=FIXBLK,
                RECSIZE=80,IOREG=(5),ERROPT=SKIP,DEVICE=3340,
                IOAREA1=INBUFW,EOFADDR=ENDWEEK
         END
```

E35 Coding Instructions

Parameter List

1. Addr of current record	Current record
2. Addr of previous record	Previous record now in the output buffer
3. Addr of action word	Action word
4. Addr of sequence check word	Sequence check word
5. Addr of record length vector	l_1 l_2 l_3 l_4 l_5 (5 words)
6. Addr of record type	Record type switch (1 byte)

Parameter 1 points to the record currently selected for output. When output is exhaused it contains all zeros.

Parameter 2 points to the record most recently moved to the output buffer. Until the first record has been moved out it contains zeros.

Parameter 3 is for your return information to SM2. You must put a return code in the rightmost byte of the action word. Valid codes are:

X'00'	Process current record normally
X'04'	Delete current record
X'08'	Do not return to this exit
X'0C'	Insert a record
X'10'	Terminate sort/merge

Parameter 4 contains zeros; it is for use if you want to insert records which are out of sequence.

Parameter 5 gives the address of a 20-byte area containing the record's length parameters l_1 through l_5.

Parameter 6 gives the address of a byte containing the record type code. Bit 0 is on and bit 1 is off for fixed length records and bit 1 is on and bit 0 is off for variable-length records. Other bits may be used by sort for flags.

Procedure

When your E35 routine first receives control, parameter 1 will normally point to the first output record. Parameters 2 and 4 will contain zeros.

a. If the current record is acceptable you return a code of X'00'.

b. If you want to delete it you return a code of X'04'.

c. If you want to insert a record which you have yourself read in, you check it against the current record (parameter 1). If it collates ahead of the current you put its address in parameter 1 and return a code of X'0C'. Next time, parameter 1 will have been restored to its former value. You can insert a record which is out of sequence; if you do so, you must inform SM2 by putting a nonzero value in the sequence check work pointed to by parameter 4.

d. If you want to change the current record you move it to a work area, change it, put its new address in parameter 1, and return a code of X'00'.

This process is repeated for every output record until you return a code of X'08' or X'10'.

If you have specified OUTFIL EXIT you return a code of X'04' every time, until parameter 1 contains zeros: then output is exhausted and, after closing your output file, you return a code of X'08'. A code of X'10' is also valid with OUTFIL EXIT.

Example of Coding at E35

The routine shown in the following example is self-relocating and prints all sorted records on SYSLST using physical IOCS.

```
PH3RTN   CSECT
         PRINT NOGEN
         USING *,15                    REG 15 POINTS TO START OF EXIT
EXITS    DC    A(0)                    EXIT E31 NOT USED
         DC    A(0)                    EXIT E32 NOT USED
         B     EXIT35                  EXIT E35 ENTRY POINT
         DC    A(0)                    EXIT E37 NOT USED
         DC    A(0)                    EXIT E38 NOT USED
         DC    A(0)                    EXIT E39 NOT USED
EXIT35   STM   14,12,12(13)            SAVE REGISTERS
         LR    4,1                     REG 4 POINTS TO LIST OF ADDRESS
*                                       CONSTANTS PASSED BY SORT
         SR    5,5                     CLEAR REG 5
         C     5,0(,4)                 1ST ADDR CONSTANT ZERO?
         BC    8,EOF                   YES, NO MORE RECORDS FROM MERGE
         L     6,0(,4)                 LOAD ADDR OF REC LEAVING MERGE
         MVC   OUTAREA,0(6)            MOVE RECORD TO PRINTAREA
         LA    1,PRINTCCB              GET ADDR OF PRINTCCB
         CLI   SW1,X'00'               CHECK IF FIRST TIME THROUGH
         BNE   PRINT                   NO, PRINT RECORD
         MVI   SW1,X'FF'               SET FIRST TIME SWITCH
         BAL   3,CHA12                 GO SKIP TO CHANNEL 1
PRINT    EXCP  (1)                     PRINT A RECORD
         WAIT  (1)                     WAIT FOR COMPLETION
         LA    6,CHA12                 GET ADDR OF CHANNEL 12 ROUTINE
         BAL   3,CHECK                 CHECK IF CHANNEL 12 REACHED
         L     6,8(,4)                 LOAD ADDR OF ACTION WORD
         MVI   3(6),X'04'              INSERT RETURN CODE (DELETE)
         B     RETURN                  GO TO RETURN TO SORT
EOF      L     6,8(,4)                 LOAD ADDRESS OF ACTION WORD
         MVI   3(6),X'08'              SET RETURN CODE (DO NOT RETURN)
RETURN   LM    14,12,12(13)            RESTORE REGISTERS
         BR    14                      RETURN TO SORT/MERGE
CHECK    TM    4(1),X'01'              TEST FOR UNIT EXCEPTION
         BCR   1,6                     YES, GO TO CHANNEL 12 ROUTINE
         BR    3                       NO, LINK BACK
CHA12    MVI   PRINTCCW,X'8B'          MODIFY PRINTCCW TO SKIP TO CH1
         EXCP  (1)                     SKIP TO CHANNEL 1
         WAIT  (1)                     WAIT FOR COMPLETION
         MVI   PRINTCCW,X'09'          RESTORE OP CODE IN PRINTCCW
         BR    3                       BRANCH BACK TO SET RETURN CODE
PRINTCCW CCW   X'09',OUTAREA,X'20',L'OUTAREA
PRINTCCB CCB   SYSLST,PRINTCCW         CCB
SW1      DC    X'00'                   FIRST TIME SWITCH
OUTAREA  DC    CL100' '
         END
```

E37 Coding Instructions

Parameter list

1. Addr of block count list	Block count list (4-byte entries)	
	Output file block count	
	Input file 1 block count	
	.	only for
	.	a merge
	.	
	Input file n block count	

Your routine at E37 receives control only once. It must:

- Process the trailer label of the last volume of output, using the first entry in the block count list passed as a parameter, and close the output file.
- For a merge, carry out the same processing for the last volume of each input file.

See the diagram and coding example supplied with the E31 coding instructions.

E38 Coding Instructions

Parameter List

1. Type indicator
2. Zeros
3. Addr of action word — Action word

The only valid return codes are:

X'00' No reply
X'04' Reply provided
X'08' Do not return

Procedure

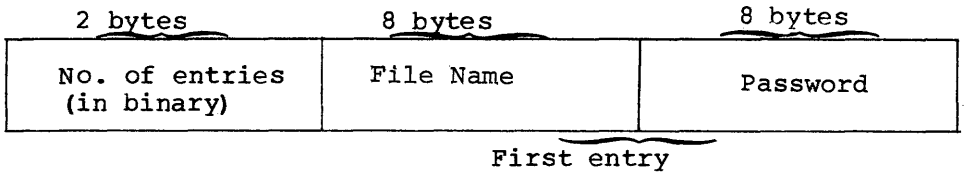
Your E38 routine is entered twice. The first time, parameter 1 contains C'PWD' (plus a padding byte). This means that passwords are requested. The second time it contains C'EXL', which is a request for an exit list.

If you have no reply, return with a code of X'00' in the action word.

Otherwise put the address of the password list or exit list (whichever has been requested) in parameter 2, and return with a code of X'04'.

Password List

The list must begin with a 2-byte entry count, and continue with a sixteen byte entry for each password-protected merge input file:



Exit List

The exit list must be built according to the rules laid down in DOS/VS Supervisor and I/O Macros or Using VSE/VSAM Commands and Macros or VSE/VSAM Documentation Subset. All routines pointed to by the list must use standard VSAM linkage to return to VSAM. They must not use Register 13, which is in use by VSAM: instead they must provide their own save area.

For end-of-file processing do NOT use the EODAD exit, as the sort will not then be able to detect end-of-file. Instead supply a LERAD exit and test the FDBK code for X'04', which indicates EOF. Do not change the code, as the sort program also uses it.

Note: The same exit list must be valid for all SORTIN files.

E39 Coding Instructions

Parameter List

1. Type indicator
2. Zeros
3. Addr of action word — Action word

The valid return codes that can be entered in the rightmost byte of the action word are:

X'00' No reply
X'04' Reply provided
X'08' Do not return

Procedure

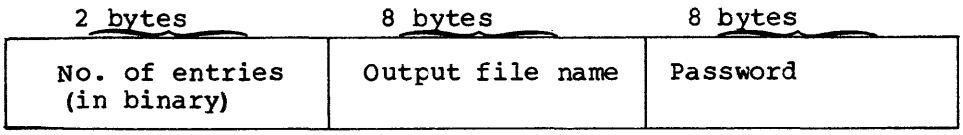
Your E39 routine is entered twice. The first time, parameter 1 contains C'PWD' (plus a padding byte). This means that passwords are requested. The second time it contains C'EXL', which is a request for an exit list.

If you have no reply, return with a code of X'00' in the action word.

Otherwise put the address of the password list or exit list (whichever has been requested) in parameter 2, and return with a code of X'04'.

Password List

The list is 18 bytes long: an entry count (2 bytes), followed by the name of the output file and its password.



Exit List

The exit list must be built according to the rules laid down in DOS/VS Supervisor and I/O Macros or Using VSE/VSAM Commands and Macros or VSE/VSAM Documentation Subset. All routines pointed to by the list must use standard VSAM linkage to return to VSAM. They must not use Register 13, which is in use by VSAM; instead they must provide their own save area.

Chapter 6. Factors of Importance for Performance

This chapter discusses performance under four main headings:

- The effect of the environment
- Choices of program function than can affect performance positively
- Those which can affect performance negatively
- Use of the DIAG option.

Effect of the Environment

The major environmental considerations are those of storage, and the characteristics of input and output.

SM2 MODULES IN THE SVA

If SM2 was installed in the system core image library most of its phases can be loaded in the SVA, as described in the DOS/VS Sort/Merge Version 2 Installation Reference Manual.

SM2 should be executed from the SVA to get maximum storage utilization and performance.

MAIN STORAGE (REAL AND VIRTUAL)

In general the more main storage available to the program the better its performance, but overcommitment must be avoided. Overcommitment will occur if virtual storage allocated to SM2 is much greater than the real storage available, resulting in heavy paging and/or deactivation taking place.

In common with other sort/merge programs, SM2 generally uses all the main storage (real or virtual) available to it.

Figure 23 shows how various parameters affect SM2's use of main storage.

In System/370 Mode

Performance is improved if the DOS/VS (DOS/VSE) system is generated with the PFIX=YES and ECPREAL=YES options, and SM2 is run in a virtual partition with a sufficiently large associated real partition (defined by the ALLOCR statement). SM2 can then utilize the page fixing and private CCW translation features to perform its I/O in real mode. SM2 will not page-fix merge applications.

An adequate size for the associated real partition allocated through the ALLOCR JCL statement can be determined as follows:

- Fixed-length record sorts: about 50% of available main storage
- Variable-length record sorts: about 100% of available main storage

In ECPS:VSE Mode

Performance is improved if SM2 is allowed to fix the buffer pages. This is controlled by the ALLOCR JCL statement, which is determined in the same way as for System/370 mode (described above).

Storage Use

The amount of main storage actually used by SM2 (whether virtual or real) is the smaller of:

SIZE - from the EXEC statement or SIZE statement
and
STORAGE - from the SM2 OPTION statement, or the installed default

The use of the STORAGE parameter is a good way to avoid overcommitment. If neither STORAGE nor SIZE is specified SM2 has access to the entire partition, and will use the amounts shown in Figure 22.

SYSTEM	VIRTUAL EXECUTION	REAL EXECUTION
DOS/VS Rel 33 and 34	Max (64K; ALLOCR + 12K) [1],[2]	ALLOCR
DOS/VSE	Up to partition GETVIS area	ALLOCR, or up to partition GETVIS area if any

[1] If execution is not possible in this environment SM2 will use as much as necessary (up to the full partition) to enable it to continue processing.
[2] If no work files have been allocated for a sort (WORK=0) SM2 will use all space available to it.

Figure 22. Default Storage Value Used by SM2

If both DIAG and PRINT=ALL have been specified, SM2 will print diagnostic messages which will tell you whether buffer pages have been fixed, whether real I/O has been used, and how storage has been used.

WORK STORAGE

Best performance can be expected when one work file is allocated on a device which is separate from the input and output devices. Allocation of more than one device for work storage does not improve performance.

With a small input file it may be possible to sort the file in the available virtual storage, without the need for any work files. Then WORK=0 can be specified. For more details see Appendix B.

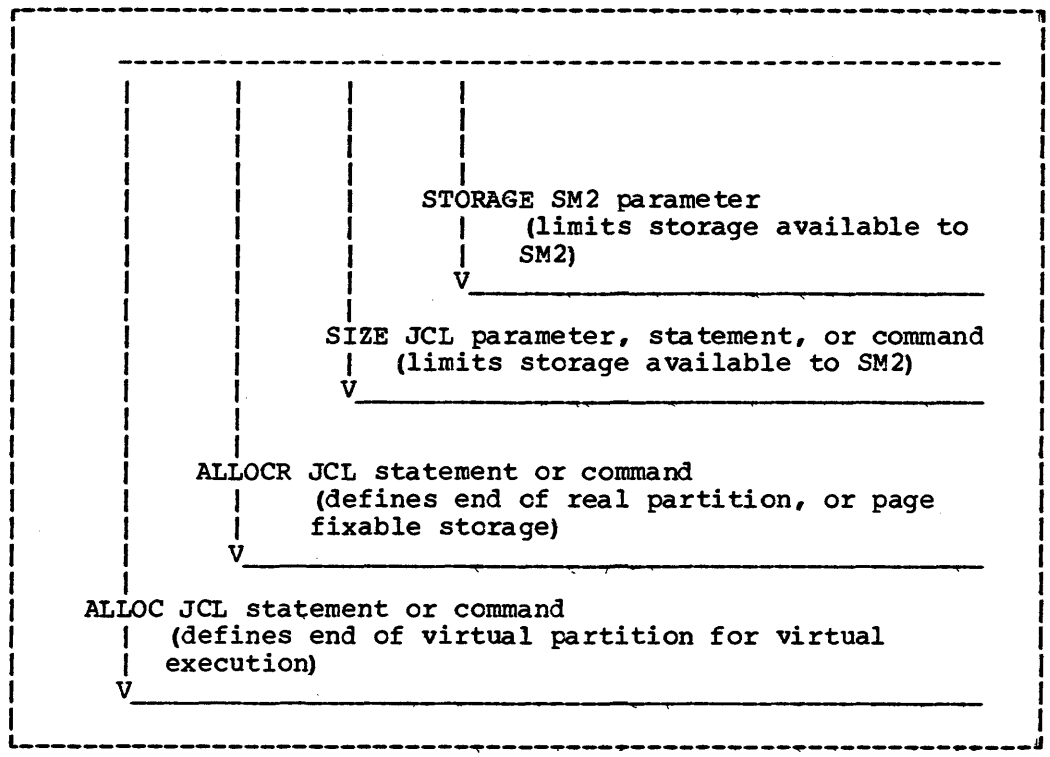


Figure 23. SM2 Storage Allocation Map

Data transfer rate can also be important. In general, the faster the data transfer rate of the work device used, the faster SM2 will run.

INPUT AND OUTPUT FILES

With tape and CKD devices performance is generally better when the input and output data is blocked. Large blocks give better performance than small ones.

With FBA devices, a large CI value gives better performance than a small one.

SPECIFICATION OF RECORD LENGTH

When sorting variable-length records you can aid performance by careful specification of l_5 in the RECORD statement.

The l_5 parameter gives the record length that occurs most frequently in the input file (modal length). If you do not specify a value for l_5 it is assumed to be equal to the average of the maximum and minimum (l_2 and l_4) record lengths in the input file. For optimum performance, both l_4 and l_5 should be specified. Specifying an inaccurate modal length may cause reduced performance.

Functions that May Affect Performance Positively

INCLUDE/OMIT

You can use the INCLUDE/OMIT statement to select for sorting or merging only those records which are needed in the output file. If only a subset of the file is selected, CPU time and data transfer time will be reduced.

SUM

You can use the SUM statement to cause records to be summarized: whenever two records with equal control fields are found, the contents of fields defined in the SUM statement are added, the result is placed in one record, and the other is deleted; any resulting reduction in the number of records to be processed by SM2 will save CPU time and data transfer time.

OUTREC

You can use the OUTREC statement to reduce the size of sorted or merged records, removing fields not needed in the output records. This will usually save data transfer time.

NOCHAIN

If the input file is on tape, and is declared as fixed format, and contains many blocks which are shorter than the specified maximum block size, you will be getting the overhead of chaining without its benefits. This is because fixed format tape input command chains cannot read past a short input block. In this case (and only in this case) specify NOCHAIN to prevent performance deterioration. However, if possible it is better to alter the BLOCKSIZE parameter to be accurate and try to avoid short blocks in the input file.

Functions that May Affect Performance Negatively

CHECKPOINT/RESTART

If SM2 is invoked from a user program, or if user-written routines are in use, the entire virtual partition in which SM2 is running is checkpointed. Otherwise only the main storage being used by sort/merge is checkpointed. Checkpointing the whole partition may take longer than checkpointing the main storage used by sort/merge; SM2 performance may therefore suffer.

VERIFY, BYPASS, ERASE, DIAG, EQUALS, DUMP, AND WORK=0.

The VERIFY option will affect performance negatively, since it involves an extra read operation of the written output. It also precludes the use of command chaining when writing output files.

BYPASS precludes the use of command chaining when reading input files. Command chaining usually provides a good performance improvement.

ERASE involves additional writes on the work files.

DIAG produces additional messages but is useful when tuning SM2.

The EQUALS option causes an additional field of four bytes to be added to each record, which increases the time needed for comparison of records and for data transfer.

When the DUMP option is in force SM2 maintains a trace table, which costs a little CPU time.

SM2 does not use command chaining on input and output when no work files are provided (WORK=0).

EFFECT OF USER ROUTINES

When user routines are included in a sort or merge application, the time required to run the job is usually increased.

The execution time required by most user routines is generally small, but the routines at exits E15, E32, and E35 are entered for each record of the file(s). For large input files, the total execution of these routines can be relatively large.

User routines also occupy main storage that could otherwise be used by SM2 to improve its performance. Depriving the program of this main storage is particularly detrimental to performance when the program is running in a small partition, or when the input file size is very large.

Using the DIAG Option

The DIAG option can be used for tuning purposes, to investigate how well SM2 is performing in its current environment and to discover whether and how improvements could be made.

If you specify DIAG in the OPTION statement you receive extra messages concerning SM2's storage use, optimization parameters, data handling, and time required, as shown in the tuning table below.

The table also shows the interpretation which can be placed on the various items of information, and how they can be used to improve performance.

Message Concerning:	Interpretation:
MAIN STORAGE USE	
Whether SM2 is executed from the SVA	Execution from the SVA gives better storage utilization and performance.
Real storage available for page fixing	If too little real storage is available you will receive one or more messages saying that input, work, or output buffer pages have not been fixed. To improve performance, increase real storage.
Virtual storage used by SM2	Determined by the partition allocation, the EXEC SIZE parameter, or SM2's STORAGE parameter.
Buffer allocation in phases 1-3	Performance improves when sort/merge uses double buffering on its files. For fixed-length records the optimum number of data buffers in the respective phases are Ph1; two input and two output; Ph2; three buffers which are rotated; Ph3, two input and two output. For variable-length records sort optimum is two input and two output in Ph1 while in the other phases sort uses M+1 on input and two output buffers (where M is merge order for the phase). If fewer have been allocated, increase virtual and preferably also real storage size.
Modal record length	It is vitally important for performance when sorting variable-length records that modal record length (l_s on the RECORD statement LENGTH parameter) is specified, and is reasonably accurate.
WORK SPACE USE	
	You can use information on work space utilization to make sure you are not being over-generous or unduly mean with work file space. If you have an application which will be run regularly, and where the amount of data is likely to grow, you will probably want to allow a generous amount of space. If however you have a shortage of direct-access space you may want to trim the allocation to the minimum.
Sort capacity	You can compare SM2's estimate of how many records it could handle with the number of records actually sorted.
Work space used	You can compare the amount of space actually used with the amount allocated.

Message Concerning:	Interpretation:
OPTIMIZATION PARAMETERS	
Block size for index and work buffers	With CKD devices work block size can be up to full track length. If it is less, performance will probably be improved if you increase real storage size. An increase in virtual storage size will also usually give an improvement as long as the relationship between real and virtual does not become excessive.
Fixable storage Real I/O Buffer area fixed	SM2 should be given enough real storage to be able to page fix the buffer areas at the beginning of each phase. This avoids the need to page fix and translate channel program addresses for each EXCP. Even in ECPS;VSE, where there is no channel program translation (no EXCPREAL option), the CPU time saving is considerable.
Merge order (M) in phases 2 and 3	The higher the merge order, the greater SM2's efficiency. If it is less than (say) 8, performance should be improved if you increase it by increasing virtual storage size.
Internal record length	If internal record length is very large (say, close to track size) you are approaching the limit that this application can handle. Internal record length can sometimes be reduced by respecifying control fields or other fields, as described elsewhere in this manual.
RSA bin size	With variable-length records the size of the root bin is affected by modal record length (RECORD statement LENGTH 1^5 parameter). Correct specification of 1^5 can be critical for performance.
DATA HANDLING	
Number of data and index blocks handled by phases 1-3	Increased storage allocation can lead to reduced data handling.
Number of physical and logical strings handled by the partitioning part of phase 2	The amount of data handling in phase 2 can give an idea of how well SM2 is performing, as can the number of physical strings handled in partitioning. If, in partitioning, the number of logical strings is lower than the number of physical strings, your input file is not random. The amount of difference gives an indication of the degree of non-randomness.
TIME REQUIRED	
CPU and elapsed time for each of phases 1-3	Enables you to study the effect of your tuning measures, if you have job accounting in your supervisor.

Appendix A. Sample of Job Streams, with Statement Format Rules

Statement Format

This appendix first gives the full format rules for program control statements, and then supplies several examples of complete job streams for executing a sort or a merge.

An example of control statement format is given in Figure 24.

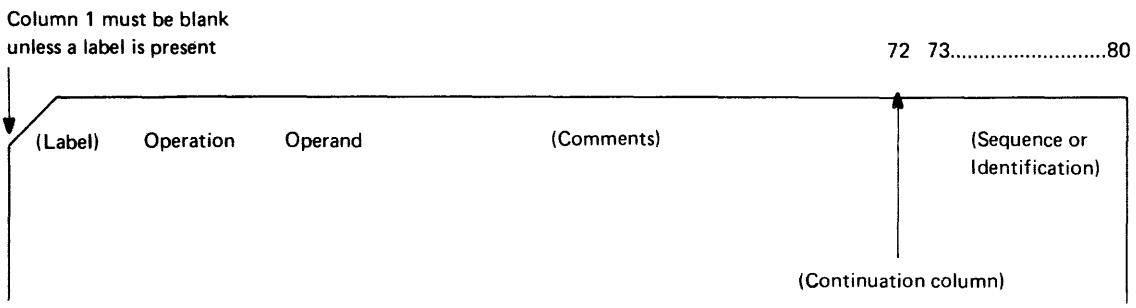


Figure 24. Control Statement Format Example

The control statements are free-form. The operation definer, operand(s), and comments may appear anywhere in a statement, as long as they appear in the proper order, and are separated by one or more blank characters. Column 1 of each control statement must be blank, unless the first field is a label, in which case it must begin in column 1.

Label Field

If present, the label must appear first on the card. It must begin in column 1, and conform to Assembler label format rules.

Operation Field

This field must not extend beyond column 70 of the first card. It contains a word (SORT, MERGE, RECORD, MODS, etc.) that identifies the statement type to the program. It must not begin in column 1 and it must be separated by at least one blank from a label field.

Operand Field

The operand field is made up of one or more operands separated by commas. This field must follow the operation field, and be separated from it by at least one blank. If the statement occupies more than one card, this field must begin on the first card.

Each operand has an operand definer, or keyword (a group of characters that identifies the operand type to the sort/merge program). A value or values may be associated with a keyword. The three possible operand formats are:

- keyword (operand 3)
- keyword=value (operand 2)
- keyword=($value_1$,$value_2$...,$value_n$) (operand 1)

When an operand of the type keyword=($value_1$,$value_2$...,$value_n$) is used, values may be omitted if they are equal to those assumed by the program. The following rules apply to omitting values from such an operand:

- Values can be dropped from right to left. Thus, if all values after $value_2$ are equal to those assumed by the program, the operand may be written: keyword=($value_1$,$value_2$).
- If values are dropped from the middle, commas must be used to signify their omission. Thus, if value is equal to the value assumed by the program, the operand may be written: keyword=($value_1$,,$value_3$).
- If only the first value of a series is needed, the parentheses are optional. An operand of this type may be written as either keyword=value or keyword=(value).

Comments Field

This field may ccntain any information you desire. It is not required, but if it is present, it must be separated from the operand field by at least one blank.

A comments field may appear on the first statement or on any continuation statement as long as there is a blank between the comma (,) of the operand to be continued and the desired comment. This allows the following layouts:

```
SORT FIELDS=(.,.,.),      SORT KEYS
WORK=1,                   NO. OF WORK AREAS
FILES=2                   NO. OF INP FILES
```

Continuation Column (72)

Any character other than a blank in this column indicates that the present statement is continued on the next card.

Columns 73-80

This field may be used for any purpose you desire.

CONTINUATION CARDS

A continuation card is treated as a logical extension of the preceding card. Either an operand or a comments field may begin on one card and continue on the next. You can indicate that a statement continues on the next card in two ways:

1. By placing any nonblank character in column 72. This method must be used when comments fields are to be continued.

2. By a comma followed by at least one blank (before column 72). When you do this, SM2 adds an asterisk (*) in column 72--unless the statement is INCLUDE or OMIT. This method cannot be used to indicate continuation of comments fields.

When preparing continuation cards the following rules apply:

- The continuation must begin in one of columns 2-16.
- The continuation column (72) and columns 73-80 of continuation card fulfill the same purpose as they do on the first card of a control statement.
- If an operand is broken at column 71, column 72 must contain a nonblank character. The continuation must then begin in column 16.

SUMMARY OF RESTRICTIONS

The following rules apply to control statement preparation:

- Unless a label is present, column 1 of each control statement must be blank.
- Labels must begin in column 1.
- The whole operation definer must be contained on the first card of a control statement and it must be separated by at least one blank from a label field.
- The first operand must begin on the first card of a control statement. The last operand in a statement must be followed by at least one blank.
- Each type of program control statement may appear only once for each execution of the sort/merge program.

The following restrictions do NOT apply to a self-defining term enclosed in quotation marks:

- Embedded blanks are not allowed in an operand Anything following a blank is considered part of the comments field.
- Values may contain no more than eight alphameric characters.
- Commas, equal signs, parentheses, and blanks can be used only as delimiters. They must not be used in values.

CONTROL STATEMENT NOTATION

In this publication, the descriptions of sort/merge program control statements use the following notation:

- Uppercase characters designate keywords or operators which should be written exactly as shown.
- The comma ',', left and right parentheses '()', and the equal sign '=' should be written exactly as shown.
- Braces, brackets, ellipses, lowercase characters, and subscripts are used to define control statements; they should not be written in a statement. They are used as follows:
  - Braces '{}' designate alternatives, only one of which should be selected.
  - Brackets '[]' designate optional parameters which may be omitted. Only one item from each bracket may be used if there is a choice.
  - Ellipses ... indicate that the preceding variable (or group of variables) is the first in a series.
  - Subscripts define the sequence of an item in a series.
  - Lowercase names and letters represent variables for which specific information must be substituted.
- An underlined parameter indicates the standard default setting supplied with the program. Note, however, that some defaults can be changed.

Examples

This appendix supplies the following complete examples:

1. Merge two tape and two disk files to a tape output file.
2. Sort a tape file using four disk work files.
3. Sort a tape file using two disk work files on four extents.
4. Sort a disk file onto tape, using five single-extent disk work files, and exits E31 and E37 for label handling.
5. Sort a VSAM file, with output in the form of disk addresses only, using a disk work file on three extents.
6. Sort parts of a tape file (INCLUDE specified), and reformat the output records (OUTREC) on a tape file, using two disk work files of two extents each.
7. Sort a tape file, summarizing equal records, and collating national characters at the end of the alphabet. One disk work file of 4 extents.

This page intentionally left blank

1. Merge four files (tape + disk); tape output

```
 1  // JOB EXAMPLE1
 2  // ASSGN   SYS001,X'282'   MERGE OUTPUT
 3  // ASSGN   SYS002,X'284'   MERGE INPUT
 4  // ASSGN   SYS003,X'191'   MERGE INPUT
 5  // ASSGN   SYS004,X'283'   MERGE INPUT
 6  // ASSGN   SYS005,X'192'   MERGE INPUT
 7  // DLBL    SORTIN2
 8  // EXTENT  SYS003,191191,1,0,20,70
 9  // DLBL    SORTIN4
10  // EXTENT  SYS005,192192,1,0,20,70
11  // EXEC    SORT,SIZE=64K
12     OPTION LABEL=(U,U,S,U,S)
13    MERGE    FIELDS=(21,4,ZD,D,9,8,PD,A,30,4,BI,A,40,4,CH,D,35,4,
14              CH,A,70,4,FL,A,90,14,CH,D,79,5,CH,A,86,2,BI,A,5,
15             4,PD,D,25,3,CH,D,130,4,BI,A),FILES=4
16    RECORD TYPE=V,LENGTH=(154,,154)
17     INPFIL BLKSIZE=1544
18      OUTFIL BLKSIZE=1544
19  /*
20  /&
```

2 Assigns the output unit, SYS001, to the tape unit at address 282.

3-6 Assigns tape units SYS002 and SYS004 and disk units SYS003 and SYS005 as input units.

7-10 Assigns two extents on the two disk input volumes. A total of 140 tracks allocated.

11 Initiates the program and specifies the SIZE parameter to restrict the amount of virtual storage available to SM2.

12 Specifies that the output tape SYS001 and input tape units SYS002 and SYS004 are unlabeled. Input disk units SYS003 and SYS005 use standard labels.

13-15 Specifies that a merge based on 12 control fields is to be executed. There are four input files to be merged.

16 Specifies that variable-length records are to be merged; the maximum length of both input and output records is 154 bytes.

17-18 Specify that input and output block size is 1544 bytes.

2. Sort: tape input and output

```
 1  // JOB EXAMPLE2
 2  // ASSGN   SYS001,X'283'   SORT OUTPUT
 3  // ASSGN   SYS002,X'282'   SORT INPUT
 4  // ASSGN   SYS003,X'162'   SORT WORK 1
 5  // ASSGN   SYS004,X'162'   SORT WORK 2
 6  // ASSGN   SYS005,X'163'   SORT WORK 3
 7  // ASSGN   SYS006,X'163'   SORT WORK 4
 8  // TLBL    SORTOUT
 9  // DLBL    SORTWK1,,0
10  // EXTENT  SYS003,,1,0,760,38
11  // DLBL    SORTWK2,,0
12  // EXTENT  SYS004,,1,0,380,38
13  // DLBL    SORTWK3,,0
14  // EXTENT  SYS005,,1,0,380,38
15  // DLBL    SORTWK4,,0
16  // EXTENT  SYS006,,1,0,760,38
17  // EXEC    SORT,SIZE=32K
18    OPTION PRINT=ALL,LABEL=(N,U),ROUTE=LST
19    SORT FIELDS=(5,4,CH,D,20,12,BI,A,50,3,CH,A),
20           WORK=4,FILES=1
21    RECORD TYPE=V,LENGTH=(158,,,54,100)
22     INPFIL BLKSIZE=1544,CLOSE=RWD
23      OUTFIL BLKSIZE=1544,CLOSE=RWD
24        MODS     PH3=(SAEXIT,L2000,E31,E37)
25  /*
26  /&
```

2-7 Assign I/O devices to be used as output, input and work units. The output and input units, SYS001 and SYS002 respectively, are assigned to tape units at addresses 283 and 282. Four work units, SYS003 through SYS006, are assigned to disk.

8 Allocate a tape file for output.

9-16 Allocate four extents on the two disk work volumes; 152 tracks are allocated in all.

17 Specifies that the program is to have 32K bytes of virtual storage available for its use.

18 Indicates that the program is to print all messages. The input is unlabeled and the output file has standard labels with additional user labels that will be processed by the user.

19-20 Specifies that a sort based on three control fields will be executed. There are four work areas available; and the input file is contained on one file.

21 Specifies that variable-length records are to be sorted. The maximum input record length is 158 bytes, as indicated by the l_1 value of the LENGTH operand. By default, the values for l_2 and l_3 are the same as l_1. The l_4 value indicates that the minimum record length is 54, and the most common (modal) record length is 100 bytes, as indicated by the l_5 parameter.

22 Specifies that the input block size is 1544 bytes, and the program is to rewind the input volume at end-of-file.

23 Specifies that the output block size is 1544 bytes,and the program is to rewind the output volumes at end-of-job.

24 Specifies that the sort program is to load the user routine SAEXIT for execution during phase 3. These routines have a length of 2000 bytes and are relocatable. The 2000 bytes are to be included within the storage size that the sort is to operate in. The user exits to be activated during phase 3 are E31 and E37 which are used for label processing of the output file.

3. Sort: tape input and output

```
 1  // JOB EXAMPLE 3
 2  // ASSGN   SYS001,X'284'   SORT OUTPUT
 3  // ASSGN   SYS002,X'282'   SORT INPUT
 4  // ASSGN   SYS003,X'191'   SORT WORK
 5  // ASSGN   SYS004,X'192'   SORT WORK
 6  // DLBL    SORTWK1,,1,DA
 7  // EXTENT  SYS003,191191,1,0,760,38
 8  // EXTENT  SYS003,191191,1,1,380,38
 9  // EXTENT  SYS004,192192,1,2,380,38
10  // EXTENT  SYS004,192192,1,3,760,38
11  // EXEC    SORT,SIZE=32K
12     OPTION PRINT=ALL,LABEL=(U,U)
13    SORT FIELDS=(5,4,CH,D,20,12,BI,A,50,3,CH,A),
14    FILES=1
15    RECORD TYPE=V,LENGTH=(158,,,54,100)
16     INPFIL BLKSIZE=1544,CLOSE=RWD
17      OUTFIL BLKSIZE=1544,CLOSE=RWD
18  /*
19  /&
```

This example is the same as Example 2, but illustrates the use of a multiextent work file.

6 One DLBL statement is provided for the work file. Code DA is specified, indicating that there is one multiextent work file, and a retention period of one day is requested.

7-10 Four EXTENT statements describe the four work extents.

11 32K bytes of virtual storage are available to the program.

12 The SORTWK parameter on the OPTION statement is not specified. The symbolic unit name SYS003 is chosen by default for the first extent. The symbolic unit names for the other extents are chosen according to the rules explained in the topic 'Work File Statements' in Chapter 3.

13 The WORK parameter of the SORT statement need not be specified, as the default (WORK=DA) applies.

4. Sort: input on disk, tape output

```
 1  // JOB EXAMPLE 4
 2  // ASSGN   SYS001,X'284'   SORT OUTPUT
 3  // ASSGN   SYS002,X'190'   SORT INPUT
 4  // ASSGN   SYS003,X'191'   SORT WORK 1
 5  // ASSGN   SYS004,X'192'   SORT WORK 2
 6  // ASSGN   SYS005,X'192'   SORT WORK 3
 7  // ASSGN   SYS006,X'191'   SORT WORK 4
 8  // ASSGN   SYS007,X'191'   SORT WORK 5
 9  // TLBL    SORTOUT
10  // DLBL    SORTIN1
11  // EXTENT  SYS002,190190,,,1900,500
12  // DLBL    SORTWK1
13  // EXTENT  SYS003,191191,,,3800,10
14  // DLBL    SORTWK2
15  // EXTENT  SYS004,192192,,,3838,12
16  // DLBL    SORTWK3
17  // EXTENT  SYS005,192192,,,5700,700
18  // DLBL    SORTWK4
19  // EXTENT  SYS006,191191,,,380,99
20  // DLBL    SORTWK5
21  // EXTENT  SYS007,191191,,,760,99
22  // EXEC    SORT
23       OPTION PRINT=ALL,LABEL=(N),ADDROUT,STORAGE=64K
24    SORT FORMAT=FL,FIELDS=(86,19,A,106,8,D,415,08,D,391,33,A),
25                 WORK=5
26    RECORD LENGTH=(1641,,,547,900),TYPE=V
27     OUTFIL OPEN=NORWD,CLOSE=RWD,BLKSIZE=80
28      MODS PH3=(PHASE3A,L19000,E31,E37)
29  /*
30  /&
```

2-8 Assign I/O devices to be used as sort output, input, and work units. The output unit, SYS001, is assigned to a tape unit at address 284. The input unit, SYS002, is assigned to a disk unit at address 190. The work units, SYS003 through SYS007, are assigned to disk units at addresses 191 and 192.

9 Specifies the tape label information for the output file.

10-21 Allocate six extents: one extent containing 500 tracks for sort input, and five extents containing a total of 920 tracks on three volumes for sort work storage.

23 Specifies that the sort program is to print all messages. The output volume will have nonstandard labels, the input volumes will use standard labels. The sort program is to operate within 65,536 bytes of virtual storage; the user routines at PHASE3A are to be included within this operating space. The ADDROUT option has been specified, so each output record is to consist of a 10-byte disk address.

24-25 Specify that a sort based on four control fields, all containing floating point data, will be executed.

26 Specifies that variable-length records are to be sorted. The maximum input record length is 1641 bytes. The minimum input record length is 547 bytes, and the most frequent (modal) input record length is 900 bytes. Output record length is 10 bytes; this parameter could be omitted as it is the default when ADDROUT is specified with non-VSAM input (see line 28).

27 Specifies that the output blocksize is 80 bytes. SM2 is not to rewind the output volume before opening the volume; it will rewind the output volume at end-of-job.

As there is no INPFIL statement, the input blocksize is 1645 bytes by default.

28 Specifies that the sort program is to load the user routines at PHASE3A for execution during phase 3. These routines have a length of 19,000 bytes and are relocatable. The 19,000 bytes are to be included within the storage size that the sort is to operate in. The user exits to be activated during phase 3 are E31 and E37. They are needed because the output file will have nonstandard labels (see statement 23).

```
5. Sort: VSAM input and output, ADDROUT specified.
 1  // JOB EXAMPLE 5
 2  // ASSGN   SYS001,X'160'   SORT OUTPUT
 3  // ASSGN   SYS003,X'163'   SORT WORK
 4  // ASSGN   SYS006,X'160'   SORT INPUT
 5  // DLBL    INPUT,'NAME.DEFINED.BY.AMS',,VSAM
 6  // EXTENT  SYS006,DISK01
 7  // DLBL    SORTWK1,,0
 8  // EXTENT  SYS003,,,,150,6
 9  // DLBL    SORTOUT,'ALSO.DEFINED.BY.AMS',0,VSAM
10  // EXTENT  SYS001,DISK01
11  // EXEC SORT,SIZE=32K
12    OPTION ROUTE=LST,DUMP,ADDROUT,FILNM=(,INPUT)
13    SORT FIELDS=(1,56,BI,A),WORK=1
14    RECORD TYPE=F,LENGTH=(80,,5)
15    INPFIL VSAM
16    OUTFIL ESDS
17  /*
18  /&
```

2-4 Assigns I/O devices for input, output, and work files. All files are on disk devices.

5-6 Specifies a VSAM file as input. Extent card not needed under DOS/VSE Release 2.

7-8 Specifies a work file on a disk; only 6 tracks are required.

9-10 Specifies a VSAM file as output. The file must have been created previously. Extent card not needed under DOS/VSE Release 2.

11 Specifies the SIZE parameter to restrict the amount of virtual storage available to sort/merge. There must be sufficient virtual storage left in the partition for VSAM use.

12 Specifies that messages are to be routed to SYSLST, that a dump is always to be taken in case of a critical message, that the name used for the input file is not 'SORTIN1', but 'INPUT', and that the output records are to consist of VSAM disk addresses only.

13 Specifies a sort based on one control field, 56 bytes long; there is one work file available.

14 Specifies that fixed-length records are to be sorted. The input record length is 80 bytes; in the output the records are only 5 bytes long (disk address).

15 Specifies that the input file is a VSAM file.

16 Specifies that the output file is an entry-sequenced VSAM file.

6. Sort: Tape input and output, INCLUDE and OUTREC specified

```
 1  // JOB EXAMPLE 6
 2  // ASSGN   SYS001,X'284'   SORT OUTPUT -- TAPE
 3  // ASSGN   SYS002,X'282'   SORT INPUT -- TAPE
 4  // ASSGN   SYS003,X'161'   SORT WORK -- DISK
 5  // ASSGN   SYS004,X'162'   SORT WORK -- DISK
 6  // DLBL    SORTWK1,,1,DA
 7  // EXTENT  SYS003,111111,1,0,760,38
 8  // EXTENT  SYS003,111111,1,1,380,38
 9  // EXTENT  SYS004,222222,1,2,380,38
10  // EXTENT  SYS004,222222,1,3,760,38
11  // EXEC    SORT,SIZE=36K
12     OPTION LABEL=(U,U)
13     SORT FIELDS=(1,4,A,6,12,A),FORMAT=CH
14     RECORD TYPE=F,LENGTH=80
15     INPFIL BLKSIZE=800,BYPASS
16     OUTFIL BLKSIZE=800
17     INCLUDE COND=(6,1,GE,C'M'),FORMAT=CH
18     OUTREC FIELDS=(1,4,6,16)
19  /*
20  /&
```

2-11 These statements are the same as in Example 3 except different disk volumes are in use. Line 12 specifies the SIZE parameter to restrict the amount of virtual storage available to SM2.

12 Specifies that the input and output tapes are unlabeled.

13 Specifies a sort based on two control fields.

14 Specifies that the records are fixed-length, 80 bytes long. All other length specifications are defaulted.

15-16 Specifies that the input and output block sizes are 800 bytes, and that input blocks causing I/O errors are to be bypassed.

17 Specifies that the sixth byte of every record is to be examined. Records whose sixth byte contains a character collating greater than or equal to M are to be included in the sort; all others are to be discarded.

18 Specifies that the output records are to consist of two fields taken from the input records; the first field is 4 bytes long and begins at byte 1 of the input record; the second field begins at byte 6 of the input record and is 16 bytes long. The effective output record length is thus 20 bytes.

7. Sort: Tape input and output, disk work areas, ALTSEQ and SUM specified

```
 1  // JOB EXAMPLE 7
 2  // ASSGN   SYS001,X'284'   SORT OUTPUT -- TAPE
 3  // ASSGN   SYS002,X'282'   SORT INPUT -- TAPE
 4  // ASSGN   SYS003,X'161'   SORT WORK -- DISK
 5  // ASSGN   SYS004,X'162'   SORT WORK -- DISK
 6  // DLBL    SORTWK1,,,DA
 7  // EXTENT  SYS003,111111,1,0,760,38
 8  // EXTENT  SYS003,111111,1,1,380,38
 9  // EXTENT  SYS004,222222,1,2,380,38
10  // EXTENT  SYS004,222222,1,3,760,38
11  // EXEC    SORT,SIZE=36K
12    OPTION LABEL=(U,U)
13    SORT FIELDS=(6,12,AQ,A)
14    RECORD TYPE=F,LENGTH=80
15    INPFIL BLKSIZE=800,BYPASS
16    OUTFIL BLKSIZE=800
17    SUM FIELDS=(51,6,ZD)
18    ALTSEQ CODE=(5BEA,7BEB,7CEC)
19  /*
20  /&
```

2-11 These statements are the same as in Example 3, except that different disk volumes are in use, and the SIZE parameter has been specified on the EXEC statement (line 11).

12 Specifies that the input and output tapes are unlabeled.

13 Specifies a single control field 12 bytes long, beginning in byte 6 of the record. An alternative collating sequence (specified in the ALTSEQ statement, line 18) is to be used.

14-16 These statements are the same as in Example 6.

17 Specifies a 6-byte-long zoned decimal summary field, beginning in byte 51 of each record.

18 Specifies that X'5B' is to collate as X'EA', X'7B' is to collate as X'EB', X'7C' is to collate as X'EC'--in other words, that national characters are to collate at the end of the alphabet.

8. Sort: VSAM managed SAM input, output, and work files.

```
 1  // JOB EXAMPLE 8
 2  *  VSAM MANAGED SAM EXAMPLE
 3  // DLBL SORTIN1,'INPUT.FILE',,VSAM
 4  // DLBL SORTOUT,'IMPLICIT.DEFINE',10,VSAM,RECORDS=1000,          c
 5                  RECSIZE=500,DISP=(,KEEP)
 6  // EXTENT ,DISK01
 7  // DLBL SORTWK1,,,VSAM,DISP=(,DELETE)
 8  // EXEC SORT,SIZE=40K
 9    SORT FIELDS=(10,20,CH,A)
10    RECORD TYPE=V,LENGTH=(500,,,100,400)
11    INPFIL BLKSIZE=4000
12    OUTFIL BLKSIZE=4000
13 /*
14 /&
```

3 Defines sort input as file 'INPUT.FILE' managed by VSAM. This file was defined by IDCAMS and previously loaded by another program.

4-5 Defines sort output as managed by VSAM. This file will be implicitly defined to be able to contain, 1000 500-byte records. It has a retention period of 10 days.

6 Indicates to VSAM that output file must be defined on disks with serial number DISK01. Can be omitted if there is a default model cataloged which indicates the required disk.

7 Defines sort work file as managed by VSAM. This file is assumed to have been previously defined by IDCAMS with the NOALLOCATE(NAL) and REUSE attributes. Its space will be allocated when sort opens it and deallocated when sort closes it.

9 Specifies a sort based on one control field 20 bytes long. The output will be in ascending order. By default one work file and one input file are assumed.

10 Specifies that variable length format records will be sorted whose maximum length is 500 bytes, minimum length 100 bytes and modal length 400 bytes.

11 Specifies that the input file will have a maximum logical blocksize of 4000 bytes. VSAM is not specified, so that sort will use SAM access.

12 Specifies that the output file will have a maximum logical blocksize of 4000 bytes. ESDS is not specified, so that sort will use SAM access.

Note: All VSAM definitions are assumed to be in the Master Catalogue.

Appendix B. Storage Requirements

A merge operation needs main storage only. A sort usually also needs work storage space on a direct-access device.

If there is plenty of main storage and the input file is small, a sort may be able to run without work storage. If there is less than, say, 64K of main storage, and work files are needed, then there is a relation between the two requirements; the less main storage available, the more work space you will need.

You can always find out in advance how much storage space a given application would need by submitting the complete control statement set, with the addition of an ANALYZE statement. As described in Chapter 2, SM2 will then analyze all the control statements and make the usual optimization calculations. It will issue all the usual messages (including those specifying how many records can be sorted with the given configuration; how much more main storage should be allocated, if the allocation was insufficient; and the internal length of the records to be sorted or merged), and then terminate without sorting or merging.

This appendix does not therefore describe in detail how to calculate space requirements. However it does give rules of thumb for making reasonable estimates of:

1. Minimum main storage requirements for a sort or merge.
2. Main storage requirements for a sort with no work files.
3. Work file size for a sort using at least minimum main storage.

Minimum Main Storage

SM2 needs a minimum of 32K bytes of virtual storage.

The minimum requirement for a given application can be more than 32K bytes. The major factors affecting the requirement are:

- Use of the SVA
- Input and output buffer sizes
- Size of user routines at program exits
- Use of special functions
- Internal record length

These factors are considered in turn below.

USE OF THE SVA

If eligible SM2 modules were not put in the SVA when your system was IPLed, they will instead have to be loaded into your partition. They need about 12K bytes of storage.

INPUT AND OUTPUT BUFFER SIZES

The sort needs at least one input buffer in Phase 1. The size of the buffer is the (maximum) block size or CI size, plus CCW size, plus the size of any IDALs required (Release 33 or 34), or the size of the page fix list, if any (DOS/VSE). The record area is rounded up to the nearest whole number of fullwords.

The elements and their sizes are shown in Figure 25. IDAL size is not shown because it varies in proportion to the size of the record area (plus 8 byte count field if DASD). It should seldom exceed 100 bytes.

It is given by

$$\left[\frac{\text{Record Area} - 2}{\text{Page Size}} + 2\right] \times 4 \text{ bytes}$$

Page Size is usually 2K for DOS/VS

Device	Record area	Input CCW	Fix list	Output CCW
CKD disk	Block size +8	40	20	32*
- with RPS	Block size +8	56	20	48*
FBA disk	CI size	24	20	24
Tape	Block size	8	12	8
*If VERIFY is used, add 24 (32 if RPS)				

Figure 25. Input and Output Buffer Element Sizes, in Bytes

Note: If command chaining is used for tape or CKD disk I/O, for each block over 1 chained, add 8 bytes to the CCW length and block size (+8 bytes if DASD) to the record area.

In Phase 3 (or Phase 1, if output is from that phase) SM2 needs one output buffer made up in the same way.

SIZE OF USER ROUTINES AT PROGRAM EXITS

The size of any routines to be used at program exits must be included in the storage allocated for SM2, unless the routines are preloaded and SM2 is called from another program.

USE OF SPECIAL FUNCTIONS

Some of the special functions provided by SM2 require that special routines be generated at execution time. They are INCLUDE, OMIT, and SUM, all of which need to handle the logical relations specified by you in the appropriate program control statement.

The extra space required by these generated routines is usually so small that it can for all practical purposes be ignored.

In any case it is never more than 4K bytes per generated routine, and (since INCLUDE and OMIT are mutually exclusive) can thus never exceed 8K for any one application.

If you use the DELBLANK parameter of the RECORD statement, a routine is generated of the same length as if you had coded an equivalent INCLUDE statement.

INTERNAL RECORD LENGTH

If the records handled internally by a sort are longer than about 1K bytes, you will probably need more main storage.

Internal record length is usually input record length (l_4 on the RECORD statement) rounded up to a whole number of fullwords, as long as all control fields have either EBCDIC character or binary format. Other formats, and use of EQUALS or ADDROUT, will cause a change. Use of SUM or OUTREC may also do so.

If you run SM2 with the DIAG option or ANALYZE statement, you will receive a message telling you the length:

```
7C12I INTERNAL RECORD LENGTH = xxxx BYTES.
```

Sort Main Storage Without Work Files

If your input file is not very big you may not need work files.

As a rule of thumb, you can sort an input file of about 100K bytes with 20K bytes or more of virtual storage. This presupposes that SM2 is executed from the SVA, and that input block size or CI size is moderate.

Input file size is simply input record length multiplied by the number of records to be processed (after any INCLUDE or OMIT statement has been used to select a subset of the input).

Work Files

As a rule of thumb, if work files are needed they should be the same size as the input file, plus about 25%. In the worst possible case you might need to add 80% instead of 25%; this would be if all the following conditions were met:

- Control fields were neither binary nor character format
- The sum of control field lengths were close to the maximum
- Input records were variable-length
- Input records were not much longer than total control field length.

With CKD work files you must allocate a minimum of four tracks. With FBA work files you must allocate a minimum of 64 blocks.

Appendix C. Conversion Aids

This appendix describes the changes that will need to be made to your existing IBM sort/merge applications if they are to run under SM2.

The first section concerns the comparatively minor changes needed to convert from programs which are closely related to SM2:

DOS Tape and Disk Sort/Merge, 360N-SM-483
DOS Tape and Disk Sort/Merge Program Product 5736-SM1
DOS Sort/Merge Program Product 5743-SM1
DOS/VS Sort/Merge Program Product 5746-SM1

The second section covers conversion from less similar programs 'Unrelated Programs', discussing seprately the three topics of JCL statements, program control statements, and user routines at program exits. The programs are:

DOS/TOS Tape Sort/Merge 360N-SM-400
DOS Disk Sort/Merge 360N-SM-450
Model 20 Disk Sort/Merge

The third section deals with conversion from a completely different sort, the System/3 Disk Sort (5702-SM1 and 5703-SM1).

Because of differences in techniques storage requirements may be different. Also, there are differences in maximum record length that can be sorted. For more details see Appendix B. User routines or programs relying on internals of previous sort/merge will not work.

Related Programs

Figure 26 summarizes the differences between SM2 and those programs which are closely related to it. As shown, in most cases if a nonsupported function or parameter is specified it will simply be ignored.

Figure 27 shows how the incompatibilities can be dealt with. Note however that any user routine or program relying on the internals of previous sort/merge programs will not execute correctly with SM2.

NONSUPPORTED FUNCTIONS

Tape work files*
Pooling of input or output with work files*
Use of ALTSEQ when input is in ASCII form*

Exit Use:

VSAM I/O error handling not using Exit List at E18, E38, or E39
Processing SAM read errors at E18 or E38
Processing SAM write errors at E39
Using Phase 2 program exits

CONTROL STATEMENTS AND PARAMETERS WHICH ARE NOT USED

Statement	Parameter
	ADDROUT=D*
	ALTWK
OPTION	CALCAREA
	KEYLEN
	TP
INPFIL	PRESEQ
SORT	SIZE
END	--

*These cause the program to terminate if specified. All other listed functions and parameters are accepted but ignored by SM2.

Figure 26. Differences from 5746-SM1 and Similar Programs

PREFERRED STATEMENTS AND PARAMETERS

Some of the parameters listed in the control statements of earlier versions of DOS and DOS/VS sort/merge program products have become out-of-date with present-day systems, usage, and standards. Most of these parameters are still accepted for compatibility reasons but as they only add unnecessary double choices they have been removed from the list of parameters given with the control statements in Figure 26.

In addition to the invalid parameters previously listed, other parameters which should be avoided in new applications are shown together with the preferred alternative parameter in Figure 27.

Statement	Old Form of Parameter	Preferred Form
SORT	CHKPT	CKPT
MERGE	ORDER=n	FILES=n
RECORD	DELBLANK	OMIT Statement
	PRINT	PRINT=ALL
OPTION	ADDROUT=A	ADDROUT
	FILNM=(,,,,,,,,,,work)	WORKNM=work

Figure 27. Preferred Parameters

Conversion

The table in Figure 28 recommends solutions for conversion problems. Users converting from DOS to DOS/VS may want to relink-edit those of their user routines that are not self-relocating, in order to make them eligible for relocation by the relocating system loader. (Relocatability is discussed in the DOS/VS System Management Guide.) MODS statements referring to re-link-edited routines may need to be changed.

INCOMPATIBILITY	CONVERSION SOLUTION
Tape work files	Change your JCL statements
Pooling of input or output files with work files	Change your JCL statements
I/O error checking at user exits E18, E38 and E39	SAM exits are ignored. VSAM I/O error can be handled by programming the VSAM Exit List facility at these exits
ADDROUT=D output records	Write your own routine and insert records containing direct access storage address and sort keys at user exit E15
Output records with separate keys (KEYLEN)	If such output data is required where the key area is separate from the data area, exit E35 can be used to insert a routine to separate the key from the data. Note that SM2 can read keyed data provided it is fixed-length, unblocked, and sequential.
Requirement for CALCAREA option	Use ANALYZE statement

Figure 28. Incompatibility and Conversion

Unrelated Programs

Conversion from the programs listed above as 'unrelated' is dealt with in this section, under three topics:

- JCL statements
- Program Control Statements
- Routines at Program Exits

The following differences in I/O device assignment should be noted:

1. File names on DLBL/TLBL cards must be changed to agree with those specified in Chapter 3, or must be specified in the FILNM operand of the OPTION statement.

2. For checkpoint facilities, the unit assignment must be given for SYS000. The file name must be SORTCKP, and the volume must have a standard label.

PROGRAM CONTROL STATEMENTS

Described below are the changes which may need to be made to your existing statements. This section does not cover how to add parameters to take advantage of the additional features of SM2.

DOS/TOS TAPE SORT/MERGE, 360N-SM-400

Statement	Parameter	Action
SORT/MERGE	FIELDS	Check that all control fields are within the first 4092 bytes of the record.
RECORD	l_4 and l_3	Check that these values agree with the block size specified in INPFIL and OUTFIL: You cannot use these parameters to lengthen or shorten records--use exit E15 or E35 instead, and add l_2 if you use E15.
INPFIL	BYPASS	You will no longer get a count of the records bypassed.
MODS	EXIT	Check that only valid exits are specified: in phase 1, E11, E15, E17, and E18; in phase 3, E31, E32, E35, E37, E38 and E39.
OPTION	FILES	Move to SORT statement.
	PRINT	Add a value (ALL, CRITICAL, or NONE).
	LABEL	Change the order to: output,$input_4$,...$input_n$
	DENSITY	Remove.
END		Optionally, remove.

DOS DISK SORT/MERGE, 360N-SM-450

Statement	Parameter	Action
SORT	(WORK)	If you are using single-extent work files, add the WORK parameters.
RECORD	l_1 and l_3	Check that these values agree with the block size specified in INPFIL and OUTFIL: You cannot use these parameters to lengthen or shorten records -- use exit E15 or E35 instead, and add l_2 if you use E15.
INPFIL	INPUT	Remove.
	BYPASS	You will no longer get a count of the records bypassed.
OUTFIL	OUTPUT	Remove.
MODS	EXIT	Check that only valid exits are specified: in phase 1, E11, E15, E17, and E18; in phase 3, E31, E32, E35, E37, E38 and E39.
OPTION	RESTART	Remove; can be replaced by a RSTRT JCL statement.
END		Optionally, remove.

MODEL 20 DISK SORT/MERGE

Statement	Parameter	Action
SORT	(WORK)	If you are using single-extent work files, add the WORK parameters.
	(FILES)	If you have more than one input file, add a FILES parameter.
INPFIL	SKIPBYTE	Ignored; optionally, remove.
MODS	EXIT	Check that only valid exits are specified: in phase 1, E11, E15, E17, and E18; in phase 3, E31, E32, E35, E37, E38 and E39.
OPTION	RESTART	Remove; can be replaced by a RSTRT JCL statement.
	LABEL	Change the order to: output,$input_a$,...$input_n$
	TIME	Remove; the program will terminate if it is retained.
END		Optionally, remove.

ROUTINES AT PROGRAM EXITS

If you want to use your existing routines at SM2 exits you will need to check, first, that its function is still supported; second, at which exit; and third, that your interface meets SM2's requirements.

Figure 29 gives an overview of the relationship in function of old exits to new.

The Model 20 Disk Sort/Merge's exits are entirely incompatible with

those of SM2 because the Model 20 Assembler language is incompatible with that of DOS/VS.

For the other two programs, a conversion guide follows Figure 29.

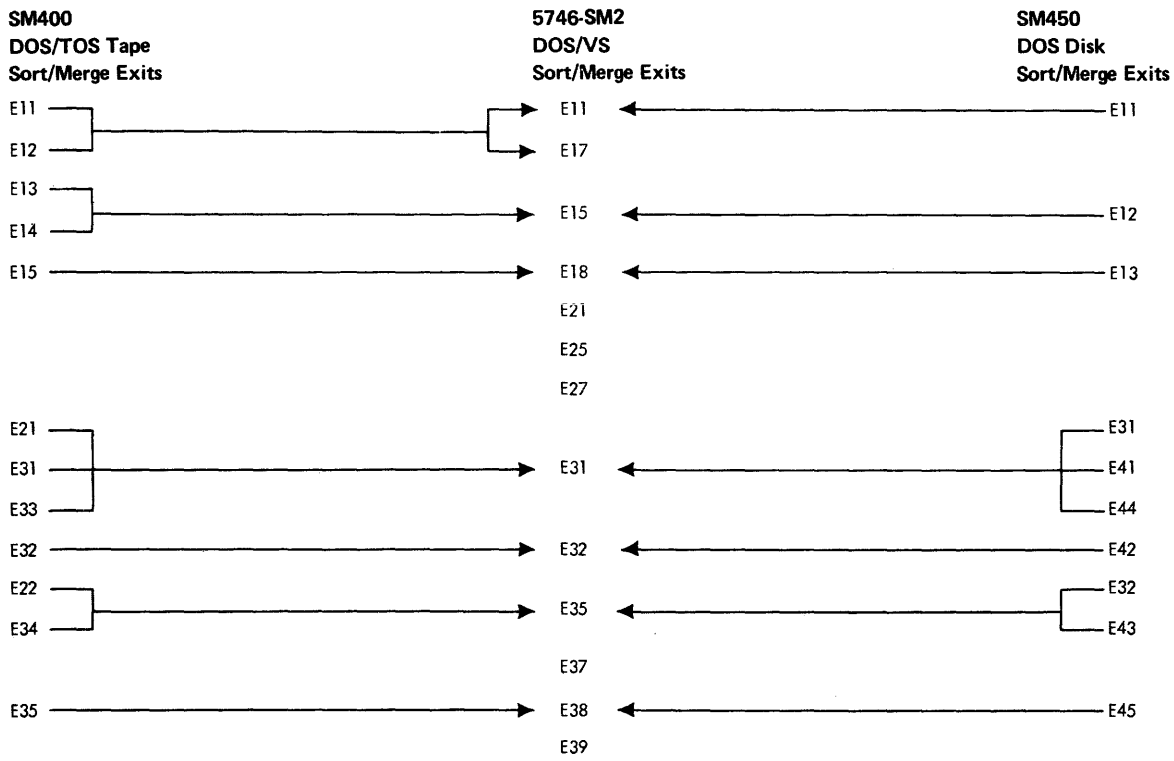


Figure 29. Correspondence of Old Exits to SM2 Exits

DOS/TOS Tape Sort/Merge, 360N-SM-400

The 5746-SM2 program exits differ from the exits of the DOS/TOS program in the ways listed below. Most of the changes affect parameters and interfaces.

E11 and E12 open functions must be combined in to the SM2 E11. E11 and E12 must be rewritten to the SM2 specifications for E11 and E17. The SM2 E17 has been added for handling the close functions of nonstandard labeled tapes and other phase termination activities.

E13 and E14 E13 and E14 functions are available in SM2 E15. E15 has also added the 'insert records' feature, and the exit is made available for each record before sort processing. Exit E15 may be used to read the entire input file, one record at a time. Present exit functions may be used with slight modifications to the interfaces. The SM2 sort interfaces are standardized -- a parameter list address is passed in register 1. User return is always BR 14 with register 1 pointing to the modified parameter list.

E15 is not supported. Above it is shown as corresponding to E18.

E21 must be changed to the SM2 E31. The SM2 sort is divided into three phases -- internal sort, external sort, and final merge. The SM2 E31 handles the functions of the present E21 and E31. E31 will have to be

rewritten to encompass E21 functions and the standard interfaces set up by the SM2 program. SM2 E31 has been expanded to include the present E21 functions in a sort application and E33 functions in a merge application. E31 will have to be rewritten to encompass these extra functions and the standard interfaces set up by the SM2 program.

E22 has to be replaced by SM2 E35. The new exit is activated just before records are moved to the output buffer. Exit E35 may be used to write the entire output file one record at a time. The user controls record modification through a standard interface. Most of the changes to the old exit will be in the interfaces and handling of the parameter list. The option to suppress sequence checking is also provided at this exit.

E32 may require changes to interfaces.

E33 functions are incorporated in SM2 E31. See also above under E21.

E34 has to be replaced by a SM2 E35. Record modification coding may be used with slight changes to the interfaces and handling of a parameter list. E34 also passed an input table. This function has been deleted. The SM2 E35 offers an option to suppress sequence checking.

E35 is not supported. Shown above as corresponding to E38.

DOS Disk Sort/Merge, 360N-SM-450

The present program exits differ from the exits of the 360N-SM-450 DOS Disk Sort/Merge program in the ways listed below. Most of the changes affect parameters and interfaces.

E11 must be rewritten to the new specifications and be split into E11 and E17. The user must handle the opening, label checking, and positioning of nonstandard or user-standard labels for initial input and intermediate storage.

E12 has to be changed into SM2 E15 and several new functions have been added. E15 allows the user to insert, delete, lengthen, shorten, or alter his records. The user can also read the entire input file at this exit. Renaming the exit and the branch table entry should be all that is needed to convert.

E13 is not supported. Shown above as corresponding to E18.

E31 must be rewritten to the new specifications. The functions of the former E31, E41, and E44 have been combined into this one exit. The user must handle opening, label checking, and positioning of nonstandard or user-standard labels for final output from a sort or merge, and initial input to a merge.

E32 is replaced by SM2 E35. The user controls record modification through a standard interface. Most of the changes to the old exit will be in the interfaces and handling of the parameter list. It should be noted that E35 must be used to lengthen or shorten records. SM2 does not truncate records. Exit E35 may be used to write the entire output file, one record at a time.

E41, see E31 above.

E42 has to be changed to SM2 E32. Changes to interfaces may be required.

E43 has been combined with E32 and given the name E35. Most of the changes to the old exits will be in the interfaces and handling of a parameter list.

E44, see E31 above.

E45 is not supported. Shown as corresponding to E38.

Converting from System/3 Disk Sort

Sort applications written for the System/3 Disk Sort programs (5702-SM1 and 5703-SM1) must be completely rewritten when converting to DOS/VS, as neither System/3 OCL statements nor System/3 Sort Sequence Specifications are compatible with DOS/VS Job Control Statements or DOS/VS Sort/Merge control statements.

The sections 'Converting Sequence Specifications' and 'Converting OCL Statements' discuss some of the points you should consider when converting a System/3 Disk Sort application into a DOS/VS Sort/Merge application. The discussions are not intended to be exhaustive, since those features of the DOS/VS Sort/Merge for which there exists no equivalent in System/3 Disk Sort are not discussed.

Most of the facilities available to the user of the System/3 Disk Sort are also available in the DOS/VS Sort/Merge Program Product 5746-SM2. SM2 also provides additional features.

The major differences are:

Additional Features

- SM2 can merge sequenced input files as well as sort.
- SM2 can link to user-written routines at points in the sort/merge called program exits. At these exits, the user-written routines may write or check labels, open or close files, take checkpoints, insert, modify, or delete records, read the input file, write the output file, or process I/O errors.
- SM2 allows use of more than one work file, and allows use of a multi-extent work file; Job Control statements specifying a work file must always be present.

Nonsupported Features

- SM2 does not support forced control fields. If this function is required it must be performed by a user-written routine at a program exit.
- SM2 allows only one INCLUDE/OMIT statement, but the statement can contain many selection conditions (is equivalent to many Record Type Specifications).
- SM2 does not support multiple record types directly control fields must have the same relative position in each record. However, any record type (or types, if the control fields are equivalent) can be accessed by selecting that type (or types) with an INCLUDE/OMIT statement.

- SM2 does not support comparison of zone or digit parts only of a field. All fields specified must be a whole number of bytes long and must begin and end on a byte boundary.

Examples of complete SM2 job streams are given in Appendix A.

Converting Sequence Specifications

This section discusses some of the points you should consider when converting System/3 Disk Sort Sequence Specifications to SM2 control statements.

1. The information on the Header Specification line and the Field Description lines for control fields is used on the SORT or MERGE control statement. Here you specify your control fields and their formats. You will get a tag-along sort (with the output records identical to the input records) unless

   - You specify the ADDROUT option on an OPTION statement (to get an addrout sort).
   - You specify summary fields on a SUM statement (to get a summary sort).
   - You specify record reformatting by means of the OUTREC statement (partial tag-along).

2. The information on the Field Description lines about data fields need not be specified if you want a tag-along sort where the output records are identical to the input records.

3. To specify a tag-along sort with record reformatting, where the output records are not identical with the input records, code an OUTREC statement, specifying which fields from the input record are to appear in the output record, and specifying how the fields are to be aligned within the records.

4. To drop control fields from records, code an OUTREC statement, specifying which fields from the input record are to appear in the output record, and specifying how the fields are to be aligned within the records.

5. To select records for inclusion in or omission from the sort, code an INCLUDE/OMIT statement. Even where the System/3 Disk Sort requires more than one Record Type specification, all the selection conditions can be coded on one INCLUDE/OMIT statement. Note that SM2 accepts only one INCLUDE or one OMIT statement.

6. To specify an addrout sort, code an OPTION statement with the keyword ADDROUT. Note that disk addresses are 10 bytes long for DOS/VS SAM files, and 5 bytes long for DOS/VS VSAM files.

7. To specify a summary sort, code a SUM statement to define the fields to be summarized. Note that you do not define summary overflow fields; if SM2 detects an overflow condition, the two records involved remain unsummarized.

8. To handle signed control fields, you need only specify the correct format parameter on the SORT statement when the field is defined as a control field. No special coding is involved.

9. You must always code a RECORD statement to describe your records; you will need to code an INPFIL and an OUTFIL statement to describe your input and output files respectively, unless the default values assumed will be satisfactory.

Converting OCL Statements

This section discusses some of the points you should consider when converting the System/3 OCL statements necessary for a sort to the DOS/VS Job Control statements necessary for sort/merge.

1. The // LOAD statement is replaced by the // EXEC statement, which must appear last of the Job Control statements, immediately before the program control statements.

2. Each // FILE statement must be expanded into a // ASSGN statement, a // DLBL statement, and one or more // EXTENT statements. If tape files are to be used, // TLBL statement may be necessary. You may not need the // ASSGN statement if your installation has a suitable standard assignment for the file in question.

3. The // RUN statement is not used.

4. A // JOB statement must be coded and placed first, before all other statements. This is then followed by all the necessary // ASSGN, // DLBL, and // EXTENT statements. The // EXEC statement must appear next, followed by the program control statements. A /* statement must follow the END statement, and a /& statement must appear last of all.

Appendix D. Permitted Data Formats

The format descriptions refer to the assembled data formats as used with IBM System 360/370. If for example, a data variable is declared in PL/I as FIXED DECIMAL it is the compiled format of the variable that must be given in the 'f' field of the Sort Statement, not the PL/I declared format. In this case the 'f' field would be PD (packed decimal) because the PL/I compiler converts fixed decimal to packed decimal form.

DATA FORMAT EXAMPLES

Format	Description

CH (character EBCDIC, unsigned). Each character is represented by its 8-bit EBCDIC code.

```
Example:   AB7 becomes
   C1        C2       F7        Hexadecimal
11000001   11000010  11110111 Binary
```

ZD (zoned decimal, signed). Each digit of the decimal number is converted into its 8-bit EBCDIC representation. The sign indicator replaces the first four bits of the low-order byte of the number.

```
Example:  -247 becomes
   2         4       -    7       Decimal
   F2        F4       D7         Hexadecimal
11110010   11110100  11010111 Binary

The number +247 becomes
   F2        F4       C7
   11110010 11110100 11000111
```

PD (packed decimal, signed). Each digit of the decimal number is converted into its 4-bit binary equivalent. The sign indicator is put into the rightmost four bits of the number.

```
Example:  -247 becomes
 2   4    7                    Decimal
   24       7D                 Hexadecimal
00100100 01111101              Binary

The number +247 becomes 247C in hexadecimal.
```

FI (fixed point, signed). The complete number is represented by its binary equivalent in either halfword or fullword format. The sign indicator is placed in the most significant bit position. 0 for + or 1 for -. Negative numbers are in 2's complement form.

```
Example:  +247 becomes in halfword form
      00F7                     Hexadecimal
0000000011110111               Binary

The number -247 becomes
      FF09                     Hexadecimal
1111111100001001               Binary
```

Format	Description

BI (binary unsigned). Any bit pattern.

FL (floating point, signed). The specified number is in the two-part format of character and fraction with the sign indicator in bit position 0.

```
Example:  +247 becomes
0 1000010 11110111000000000.....
+ chara.     fraction
```

-247 is identical except that the sign bit is changed to 1. SM2 assumes that the numbers are normalized, i.e. that the high-order hexadecimal fraction is nonzero (unless the whole fraction is 0).

AC (character ASCII, unsigned). This is similar to format CH but the characters are presented with ASCII code.

```
Example:  AB7 becomes
   41       42       37     Hexadecimal
01000001 01000010 00110111  Binary (ASCII code)
```

CSL (signed number, leading separate sign). This format refers to decimal data as punched into cards, and then assembled into EBCDIC code.

```
Example:  +247 punched in a card becomes
   +        2        4        7      Punched numeric data
   4E       F2       F4       F7     Hexadecimal
01001110 11110010 11110100 11110111  Binary EBCDIC code

-247 becomes
   -        2        4        7      Punched numeric data
   60       F2       F4       F7     Hexadecimal
01100000 11110010 11110100 11110111  Binary EBCDIC code
```

CST (signed numeric, trailing separate sign). This has the same representation as the CSL format except that the sign indicator is punched after the number.

```
Example:    247+ punched on the card becomes
F2  F4  F7  4E   Hexadecimal
```

CLO (signed numeric, leading overpunch sign). This format again refers to decimal data punched into cards and then assembled into EBCDIC code. The sign indicator is, however, overpunched with the first decimal digit of the number.

```
Example:  +247 with + overpunched on 2 becomes
  +2       4        7       Punched numeric data
  C2       F4       F7      Hexadecimal
11000010 11110100 11110111   Binary EBCDIC code
```

Similarly -247 becomes D2 F4 F7.

Note: The overpunched sign bit is always hex 'C' for positive and hex 'D' for negative.

CTO (signed numeric, trailing overpunch sign). This format has the same representation as for the CLO format except that the sign indicator is overpunched on the last digit of the number.

```
Example:  +247 with + overpunched on 7 becomes
F2 F4 C7 hexadecimal.
```

Format	Description

ASL (signed numeric, ASCII, leading separate sign). Similar to the CSL format but with decimal data assembled into ASCII code.

```
Example:  +247 punched into card becomes
   +        2        4        7       Punched numeric data
   2B       32       34       37      Hexadecimal
00101011 00110010 00110100 00110111   Binary ASCII code
```

Similarly -247 becomes 2D 32 34 37 hexadecimal.

AST (signed numeric, ASCII, trailing separate sign). This gives the same bit representation as the ASL format except that the sign is punched after the number.

```
Example:   247+ becomes
32 34 37 2B hexadecimal
```

A detailed description of CH, ZD, PD, FI, BI and FL data formats can be found in OS/VS-DOS/VSE-VM/370 Assembler Language, GC33-4010, Section G.

Appendix E. Program Messages

This section lists, explains, and suggests appropriate responses to messages produced by the program.

A serious statement error does not immediately stop program execution. Each statement is checked until one critical error is found then the rest of the statement is skipped. Usually, any continuation statements are also skipped, and are printed with the word 'FLUSHED' in columns 74-80. The next statement is then scanned.

Different Types of Message

Four different types of message are produced by SM2:

- General messages containing information or a warning
- Critical messages giving information or a warning
- Program error messages which are designed to help in tracing any error in the SM2 code
- Diagnostic messages producing information which can be used for tuning purposes

All messages have the same numbering and format, and are described sequentially in this appendix. After the message descriptions follows a section giving more details of the program error messages.

Chapter 6 describes briefly how diagnostic messages can be used, under the heading 'Using the DIAG Option'.

The messages begin with a 5-digit code. For SM2 the first character of this code is always 7. The second character indicates the module or phase in which the message was produced:

```
A-D = phase 0
E   = phase 1
F,G = phase 2
H,J = phase 3
L   = merge-only
M   = input/output modules
K   = debug module
V   = VSAM special error module, called
      by VSAM input or output module
```

The third and fourth characters are the message number. The fifth character is always I.

Messages displayed on the console log will have the job name inserted between the message number and the message text. When SM2 is subtasked then the subtask name appears in all messages (LOG or LST).

When and Where Messages are Produced

The standard default is that all messages except for diagnostic messages are produced, and are directed to the SYSLST printer. In addition, critical messages are routed to the SYSLOG printer, as is the 'SORT|MERGE COMPLETE' message.

These defaults can be changed at any time after the program has been installed, as described in the DOS/VS Sort/Merge Version 2 Installation Reference. You can also change them at execution time by use of the OPTION statement:

Required action	What to specify
Route all messages to the console	ROUTE=LOG
Produce diagnostic messages	DIAG
Print only critical messages	PRINT=CRITICAL*
Suppress all messages	PRINT=NONE*
Route all messages to a device of your choice (only when SM2 is invoked from another program)	ROUTE=xxx
*Will also suppress diagnostic messages, even if DIAG is specified	

If you include an ANALYZE CALC statement the effect will be the same as if you had specified:

```
OPTION  DIAG,NODUMP,ROUTE=LST|xxx,PRINT=ALL
```

*** SORT/MERGE 5746-SM2, REL n, MOD n, PTF nn DATE xx/xx/xx

Explanation: This heading is printed before all other messages if PRINT=ALL is in effect, and gives status information on which release, modification level, and PTF update is in use. Date is current date of execution. If PRINT=CRITICAL is specified the heading is printed only if a critical message is produced.

System Action: None.

Programmer Action: None.

7A01I INSUFFICIENT STORAGE

Explanation: Less than 32K bytes of main storage were available for the program. The minimum storage requirement for SM2 is 32K bytes.

System Action: SM2 terminates.

Programmer Action: Increase the EXEC SIZE parameter or the SORT or MERGE STORAGE parameter. If sort/merge is called ensure that all exit points and return point are below SM2's entry point, or that they are more than 32K bytes above this.

7A02I ILUSMANS HAS WRONG SVA STATUS

Explanation: ILUSMANS, which is used for restart if checkpoints are requested and SM2 modules are in the SVA, must be in the SVA if used. If SM2 modules are in the SVA when a checkpoint is taken they must also be in the SVA when the job is restarted.

System Action: SM2 terminates.

Programmer Action: Check the SVA status of all modules.

7B00I --- CONTROL STATEMENT ---

Explanation: This is a printout of the control statement or part of control statement presently being scanned. If an error is detected a $-sign is printed under or near the parameter in error. (No $-sign is produced for the INCLUDE or OMIT statement).

System Action: None.

Programmer Action: None.

7B01I SUM FIELD n, xxxxxxxx INVALID

Explanation: The length (m) or position (p) of the n-th field defined in the SUM statement is invalid.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check FIELDS parameter in the SUM statement for invalid length or position value.

7B02I FIELD OR VALUE GT 8 CHAR - xxxxxxx

Explanation: A field or value has been detected in the statement represented by xxxxxx which is greater than 8 characters -- the longest valid length.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check specified statement for field or value greater than eight characters.

7B03I MULTIPLY DEFINED EXIT Enn

Explanation:
No exit number can be defined more than once in the MODS statement. Exit Enn has been defined more than once.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check MODS statement.

7B04I NO xxxxxx CARD

Explanation: An essential control statement has been omitted: either SORT or MERGE (not both), or RECORD. xxxxxx will be replaced by the statement definer.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Supply the missing statement.

7B05I STATEMENT DEFINER ERROR

Explanation: A valid statement definer has not been found between columns 2 and 70. The first field (or second field if a label is present) of a card that is not a continuation card must be a valid statement definer, that is, SORT, MERGE, RECORD, MODS, INPFIL, OUTFIL, INCLUDE, OMIT, ALTSEQ, SUM, OUTREC, ANALYZE, OPTION, or END.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the statement for incorrect, misplaced, or misspelled operation definers. This message can be triggered by an INCLUDE or OMIT continuation card if analysis of the previous card was left incomplete after an error was detected, because SM2 may not then be aware that the card is a continuation.

7B06I DUPLICATE xxxxxxx CONTROL CARD

Explanation: A statement definer, represented by xxxxxxx, has been specified more than once.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check for duplicate statement types. Note that SORT and MERGE count as the same type, as do INCLUDE and OMIT.

7B07I COL. 1 OR 1 - 15 NOT BLANK

Explanation:

1. Column 1 of a continuation card or line must be blank.
2. A continuation card which follows a control card with nonblank characters in columns 71 and 72 must be blank in columns 1-15.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check for nonblank characters in column 1, or 1-15 of continuation lines.

7B08I COL 2 - 16 BLANK IN CONTINUATION CARD

Explanation: A continuation card or line did not appear where expected.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check for keypunching error, or an overflow of parameters into column 72.

7B09I INSUFFICIENT STORAGE

Explanation:

1. Main storage available to SM2 is less than 32K bytes.
2. STORAGE parameter in the OPTION statement is less than 32K bytes.
3. Insufficient storage was available to store the INCLUDE/OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action:

1. Rearrange exit and return locations in calling program to allow at least 32K bytes for SM2.

2. Check STORAGE parameter in OPTION statements for value less than 32K.

3. Either increase the STORAGE parameter in the OPTION statement and/or SIZE parameter in the EXEC statement, or decrease the number of INCLUDE/OMIT conditions specified.

7B10I TOO MANY xxxxxx VALUES

Explanation: The number of values assigned to the parameter represented by xxxxxx exceeds the maximum allowed, as shown below.

Statement/ Parameter	Maximum number of values accepted
SORT/MERGE FIELDS	4 x 12 = 48 (unless the FORMAT keyword is used, when it is 3 x 12 = 36)
INPFIL VOLUME	Value assigned to FILES keyword in SORT or MERGE statement
OPTION	
LABEL	10
SORTOUT	1
SORTIN	9
SORTWK	9
FILNM	11

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check specified keyword operand.

7B11I INVALID xxxxxx KEYWORD

Explanation: A keyword not recognized by SM2 or a duplicate or contradictory keyword has been detected in the control statement represented by xxxxxx.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check appropriate control statement for invalid, duplicate, or contradictory keyword operands.

7B12I INVALID FORMAT

Explanation: The value assigned to f in the FIELDS parameter, or the value assigned to FORMAT, must be one of the following: CH, ZD, PD, BI, FI, FL, AC, ASL, AST, CSL, CST, CLO, CTO, or AQ.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check format values given in FIELDS parameter or FORMAT value of the SORT or MERGE control statement.

7B13I CONTROL FIELD xx DISPLACEMENT INVALID

Explanation: The value assigned to p in the FIELDS parameter of a SORT or MERGE statement must be a numeral greater than zero. The control field number is represented by xx.

System Action:
Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check displacement value specified in SORT or MERGE control statement.

7B14I CONTROL FIELD xx LENGTH INVALID

Explanation: The value assigned to m in the FIELDS parameter of a SORT or MERGE statement must be a number greater than zero. The length of the CST, CSL, AST, and ASL control fields must be at least two bytes. The control field number is represented by xx.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check length values specified in SORT or MERGE control statements.

7B15I UNIT ASSGN ERROR xxxxxxx SYS(y)

Explanation: Sort/merge file xxxxx with logical unit number y, as calculated by SM2, is assigned to a device type not supported in this role by SM2.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Reassign SYS(y) to a device type supported by the program. Check logical unit numbers on ASSGN EXTENT cards and the SORTWK, SORTIN, and SORTOUT parameters of the OPTION card. If defaults have been used, check those valid for your installation.

7B16I CONTROL FIELD xx SEQUENCE INVALID

Explanation: The value assigned to s in the FIELDS parameter of a SORT or MERGE statement must be either A or D. The control field number is represented by xx.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check sequence value specified in SORT or MERGE control statements for a keypunching error.

7B17I BOTH SORT AND MERGE DEFINED

Explanation: You must not specify both SORT and MERGE for the same execution of SM2.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check application and eliminate SORT or MERGE control statement.

7B18I xxxxxx yyyyyy KEYWORD MISSING OR INVALID

Explanation: A parameter which must be specified, and for which there is no default, has been omitted or is invalid. xxxxxx represents the statement, and yyyyyy the keyword.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check appropriate control statement for missing keyword.

7B19I BLANK CARD OR NO OPERAND ENCOUNTERED

Explanation: A completely blank card, a statement with no operands (other than END), or a card with only a label was found in the control statements. The card is ignored.

System Action: Sort/merge continues normal processing.

Programmer Action: Remove blank card from input stream for next application.

7B20I GIVEN FILE SIZE INVALID

Explanation: The value assigned to the SIZE parameter of a SORT statement must be a numeral.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check SORT control statement for invalid SIZE operand.

7B21I FILES VALUE INVALID

Explanation: The value assigned to the FILES parameter of a SORT or MERGE statement must be in the range 1-9.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check SORT or MERGE statement for invalid FILES operand.

7B22I xxxxxx OPTION HAS INVALID PARAMETER

Explanation: xxxxxx represents one of the following OPTION parameters: LABEL, WORKNM, FILNM, SORTIN, SORTOUT, or SORTWK.

If LABEL, the message is generated when any character other than U, N, or S is found between two successive commas in the LABEL parameter.

If WORKNM, the file name specified is not four characters long, or does not begin with an alphabetic character.

If FILNM, the file name specified has more than seven characters (four characters for work files) or does not begin with an alphabetic character.

If SORTIN, SORTOUT, or SORTWK, the message is generated when an integer with a value outside the range 1 through 221 is used in that specific parameter.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Ensure that the OPTION card contains correctly specified LABEL, FILMN, SORTIN, SORTOUT and SORTWK parameters.

7B23I SORT WORK VALUE INVALID

Explanation: The WORK parameter in a SORT statement has been assigned a value not recognized by SM2. Permissible values are 1-9 for disk work files when SD is specified or defaulted on the DLBL statement, and DA for a disk work file when DA is specified on the DLBL statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check SORT statement for invalid WORK operand.

7B24I INVALID DATA TYPE

Explanation: The parameter for the DATA operand on the INPFIL statement is neither E nor A.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Correct the DATA parameter to E(EBCDIC) or A(ASCII).

7B25I xxxxxx KEYWORD IGNORED

Explanation:

1. To be compatible with other sort/merge programs the keyword represented by xxxxxx is accepted as valid but ignored by this program. The keywords in question are: SIZE for the SORT statement; ALTWK, CALCAREA, FREEOUT, KEYLEN, PRESEQ, RESTART, SKIPBYTE, and TP for the OPTION statement; CKPT, CHKPT, EQUALS, SIZE and WORK for the MERGE statement; and BYPASS for the OUTFIL statement.

2. If EXIT is specified in an INPFIL statement, any keyword other than DATA which follows EXIT will be ignored and is represented by xxxxxx in the error message.

3. If EXIT is specified in an OUTFIL statement, any keyword which follows EXIT will be ignored and is represented by xxxxxx in the error message.

System Action: Processing continues.

Programmer Action: None.

7B26I INVALID PHx NAME

Explanation: The exit routine name specified in a MODS statement must be a valid DOS/VS name (1-8 alphameric characters: A-Z, 0-9, ., #, @, $). The x represents the sort/merge phase number as specified in the MODS statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check MODS statement for invalid phase name.

7B27I INVALID MODS ADDRESS/LENGTH FIELD

Explanation: The address or length specified in a MODS statement must be a valid number. If the length is given the number must be preceded by the character L.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check MODS statement for invalid address or length.

7B28I INVALID PHx EXIT

Explanation: An exit not recognized by SM2 has been specified in a MODS statement. The valid exits are listed in Chapter 5. The x represents the phase number.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check MODS statement for keypunching error or other error resulting in specification of invalid program exit number.

7B29I ERROR IN LENGTH VALUE

Explanation:

1. An error has been detected in either the BUFOFF or the BLKSIZE parameter. BUFOFF can be 0-99 for ASCII input but only 0 or 4 for output.

2. A length parameter has been specified as 0 in the RECORD statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action:

1. Check INPFIL and/or OUTFIL statement for invalid BLKSIZE parameter for device being supported, or invalid BUFOFF parameter.

2. Specify valid length parameter value.

7B30I BOTH INCLUDE AND OMIT DEFINED

Explanation: Both an INCLUDE and an OMIT statement have been found in the same sort/merge application.

System Action: SM2 is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Remove one of the conflicting statements.

7B31I RECORD TYPE INVALID

Explanation: The TYPE parameter in the RECORD statement must be F, V, or D.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check RECORD statement for invalid TYPE operand value.

7B32I ALTSEQ STATEMENT HAS INVALID DATA

Explanation: Valid data consists of exactly 4 valid hexadecimal digits per entry in the CODE parameter.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check ALTSEQ statement for invalid hexadecimal digits, unpaired digits, and missing commas.

7B33I SUM FORMAT INVALID

Explanation: An invalid format was specified in the SUM statement. Only the formats FI, BI, PD and ZD may be used.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the FIELDS and FORMAT parameters of the SUM statement.

7B34I VIRT PARAMETER IGNORED

Explanation: The VIRT parameter of the OPTION statement was specified but ignored, since sort/merge was running in real mode (REAL was specified on the EXEC job control statement).

System Action: Normal processing continues.

Programmer Action: None.

7B35I VOLUME VALUE(S) INVALID

Explanation: Invalid characters as VOLUME operand. A value assigned to the VOLUME parameter of the INPFIL statement must be a numeral.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check INPFIL statement for invalid VOLUME operand.

7B36I xxxxxx FIELDS BEYOND 4092

Explanation: A SUM, SORT, or MERGE field lies beyond byte field 4092 of the record.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check length and displacement values specified in the statement concerned.

7B37I SYNTAX ERROR - xxxxxx

Explanation: A syntax error has been detected in the control statement represented by xxxxxx. Common syntax errors are:

- Unbalanced parentheses
- Missing commas
- Embedded blanks
- Redundant operands
- Missing parameters

After issuing this message, SM2 skips the rest of the statement in error, including any continuation cards or lines, and continues to scan the next statement for errors. The remainder of the statement may therefore contain errors which will not be detected until the program is rerun.

System Action: SM2 is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check specified control statement for an error in syntax.

7B38I OUTREC FIELD xxx INVALID VALUE

Explanation: One of the following errors has been detected in OUTREC field xxx:

1. The position or the length of the field is less than 1 or the end of the field greater than 32,767.
2. The first field consists of only one parameter.
3. An invalid alignment was specified; only H, F or D is allowed.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the OUTREC statement for valid specifications.

7B39I PHASE 2 EXIT(S) IGNORED

Explanation: One or more phase 2 exits have been specified (E21, E25 or E27). These exits are invalid and are ignored by SM2.

System Action: Processing continues.

Programmer Action: None.

7B40I LABEL ERROR

Explanation: A label starting in column 1 of a control card has been detected which is more than 8 characters long, or does not start with an alphabetic character.

System Action: SM2 is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check control cards for a label that does not begin with an alphabetic character or for a label that is more than eight characters long, or for an operator beginning in column 1.

7B41I SUBTASKED CHECKPOINT IGNORED

Explanation: SM2 was requested to make a checkpoint while it was subtasked.

System Action: The checkpoint request was ignored. Sort continues normal processing.

Programmer Action: If the checkpoint is needed, reorganize the job so that SM2 is not subtasked.

7B42I CHECKPOINT IGNORED - % TRKS/BLKS NEEDED ON SORTCKP

Explanation: The disk extent allocated for SORTCKP was too small to take a requested checkpoint. % tracks is the minimum needed for a CKD device, % blocks for an FBA device.

System Action: The checkpoint request is ignored. Sort/merge continues normal processing.

Programmer Action: If the checkpoint is needed increase the disk extent for the SORTCKP data set to at least % tracks or blocks. The space needed can be calculated from the formula given in the DOS/VS Macro User's Guide, GC24-5139, in the section 'Checkpointing a Program'.

7B44I INVALID SPLIT CYLINDER EXTENT ON xxxxxxx

Explanation: xxxxxxx is the name of a work file. Split cylinder extent for work files cannot be used.

System Action: Sort terminates.

Programmer Action: Correct work file xxxx EXTENT card.

7B45I STORAGE PARAMETER IGNORED. xxxxx USED

Explanation: xxxxx gives the number of bytes used by SM2. The STORAGE parameter requested more space than available in the partition. The request may be explicit or it may be implied by use of the default value. Note that the standard default supplied with SM2 can have been changed for your installation.

System Action: Sort continues, standard default value is used.

Programmer Action: Increase size parameter value on // EXEC card and/or increase partition size. If sort is called you may have to move preloaded exits and the return part of the program to allow SM2 more room, if these are loaded above SM2.

7B46I ADDRESS ERROR IN PARAM. LIST FOR INVOKED SORT

Explanation: An address in the parameter list for invoked sort has a value outside the used partition. If the address of the return code is invalid the program will program check.

System Action: Sort/merge terminates when all parameter values have been checked.

Programmer Action: Check your parameter list for invalid given address.

7B47I ---PARMLIST PRINTOUT---

Explanation: This heading is followed by a printout of the control statements which have been passed to SM2 from another program in the form of statement images. Only those with valid statement definers are printed.
If an image is longer than 72 characters, it is printed 72 characters to a line. The maximum number of characters printed per image is 1296; any characters beyond that limit are omitted.
If an AQTT table (alternative collating sequence) is defined in image form, it is not printed.

System Action: None.

Programmer Action: None.

7B48I ROUTE=NUMBER IGNORED

Explanation: ROUTE=xxx is only allowed when SM2 is called from another program.

System Action: SM2 continues, using the default value for the ROUTE option.

Programmer Action: Correct the ROUTE parameter on the OPTION statement before any subsequent run.

7B49I ERROR(x) GETTING LABEL FOR yyyyyyy

Explanation: The symbolic label access routine returned a nonzero return code of x when accessed to read the label for yyyyyyy.

System Action: SM2 terminates.

Programmer Action: Look up the documentation for the symbolic label access routine, and if possible take the action recommended for return code x.

7B50I ERROR(x) READING VTOC FOR yyyyyyy LABEL

Explanation: The common VTOC handler returned a nonzero return code of x when accessed to read the label of file yyyyyyy.

System Action: SM2 terminates.

Programmer Action: Look up the documentation for the common VTOC handler, and if possible take the action required for return code x.

7B52I WORK FILE NAME SPECIFIED TWICE

Explanation: The name of the work file is specified in both the FILNM and the WORKNM parameters of the OPTION statement.

System Action: The name specified second is accepted.

Programmer Action: For any subsequent run, delete the FIINM work file name parameter.

7B54I ERROR (%) RETURN FROM GETVCE FOR xxxxxxx

Explanation: The error % was received as return code from a GETVCE issued for file xxxxxxx.

System Action: The program terminates.

Programmer Action: Look up the return code % in the documentation for GETVCE and take the appropriate action.

7B55I GETVIS FOR xxxxxxx CVH WORK AREA FAILED. % NEEDED.

Explanation: SM2 needed more room for the Common VTOC Handler. It has tried to GETVIS a larger area but failed, because the CI size of the VTOC for the FBA disk on which file xxxxxxx lies is too large for the work area available to SM2.

System Action: The program terminates.

Programmer Action: Use the SIZE parameter or command to allow the program more GETVIS space (between the end of the space defined by SIZE and the end of the partition). Alternatively, run the application in a larger partition.

| 7B58I xxxxxxx SYSNO IGNORED

| Explanation: The SYSNO specified on the OPTION statement (or in | the sort/merge default macro if used at installation time) is | ignored for disk files in jobs running under DOS/VSE. xxxxxxx | is the file name, for example, SORTIN1.

| System Action: Sort continues.

| Programmer Action: Remove parameter from OPTION statement and/or | recompile and relink the default macro without the specification | if you intend to continue running under DOS/VSE.

7C01I INSUFFICIENT STORAGE, xxxK AVAILABLE, ADD xxx K, MODULES ARE NOT IN SVA

Explanation: SM2 needs more main storage for execution. The value given in the text is the amount of extra storage needed.

System Action: SM2 terminates.

Programmer Action: Increase SM2 storage size by increasing either the EXEC SIZE parameter, the SIZE JCL statement or command, the STORAGE option, or the partition size; or by arranging for eligible SM2 modules to be put in the SVA.

7C02I MODULE STATUS: PARTITION (POSSIBLY PERFORMANCE DEGRADATION)

Explanation: ILUSOPT and (unless sort is terminated with message 'WRONG SVA STATUS') rest of SVA eligible modules are not in SVA. To achieve best performance they should be placed there.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C03I INTERNAL RECORD LENGTH (y BYTES) TOO LARGE FOR xxxx

Explanation: The internal record length calculated by SM2 is too large for a full track on the device xxx.

System Action: SM2 terminates after Phase 0.

Programmer Action: Change work device to a device with a larger track capacity.

7C04I OUTREC RECORD IS BUILT IN PHASE n

Explanation: DIAG message. Output records can be built during either Phase 1 or Phase 3 when OUTREC is specified. This message says which alternative has been selected.

System Action: Processing continues.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C05I ESTIMATED MERGE ORDER IN PH2 = xx PH3 = xx

Explanation: DIAG message. These are the maximum merge orders calculated by the optimization module. 'Merge order' means the number of strings to be merged in one pass.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C06I INDEX BLOCKSIZE = xx WORK BLOCKSIZE = xx

Explanation: Diag message. These are the internal block sizes chosen by the optimization module.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C08I BUFFERS $\left\{\begin{matrix}PH1\\PH2\\PH3\end{matrix}\right\}$ OUT=xx IN=xx

Explanation: DIAG message. xx is the number of buffers chosen for the specified phase by the optimization module.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C09I REAL I/O DECIDED FOR xxxx

Explanation: DIAG message; xxxx indicates the phase(s) where I/O with EXCPREAL is suitable.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C10I STORAGE USED = xxxx BYTES

Explanation: Gives the number of bytes used by SM2.

System Action: None.

Programmer Action: None.

7C11I FIXABLE STORAGE = xxxx BYTES

Explanation: Gives the amount of fixable storage available to SM2.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C12I INTERNAL RECORD LENGTH = xxxx BYTES

Explanation: DIAG message. This is the length of a record handled internally by SM2. For variable-length records it is the maximum length.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C13I PH0: TIME A: yy SEC, TIME B: zz SEC

Explanation: DIAG message. Time A (yy) is the difference in the values returned by the GETIME macro at the start and end of phase 0. Time B (zz) is one of the following:

- If Job Accounting is supported in your system, zz is the difference in the values obtained at the start and end of phase 0 from the CPU time counter (ACCTCPUT field) in the Job Accounting Interface Partition Table.
- If you do not have Job Accounting, zz is the difference in the values returned at the start and end of phase 0 by the TTIMER macro.

System Action: None.

Programmer Action: None.

7C14I RSA BINSIZE = xxx BYTES

Explanation: DIAG message issued when input records are variable-length. This is the bin size used in ILUSCRE or ILUSVRE.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7C15I GENERATED EXTRACT/RESTORE ROUTINE > 4K

Explanation: The generated extract/restore routine is larger than 4K.

System Action: Sort/merge terminates.

Programmer Action: Decrease the number of OUTREC fields.

7C16I MODULE STATUS: SVA

Explanation: DIAG message. ILUSOPT and (unless sort is terminated with message 'WRONG SVA STATUS'), the rest of SVA eligible modules are in the SVA.

System Action: None.

Programmer Action: None.

7C17I CHAINING FACTOR OUT=n_1, IN=n_2, INDEX=n_3

Explanation: DIAG message. Nx physical blocks are read or written in one I/O operation on input, output, and index handling.

System Action: None.

Programmer Action: None.

7C18I ECPS:VSE MODE OF OPERATION

Explanation: DIAG message. SM2 is running in ECPS:VSE mode.

System Action: None.

Programmer Action: None.

7C19I ANALYZE END

Explanation: An ANALYZE statement has been supplied and the program has terminated after phase 0.

System Action: The program terminates.

Programmer Action: None.

7C20I INSUFFICIENT STORAGE, xxxK AVAILABLE, ADD xxx K, MODULES ARE IN SVA

Explanation: SM2 needs more main storage for execution. Add the value given in the text to your partition size--or, if SM2 is subtasked, add the value to the STORAGE parameter value. The SM2 modules are assumed to be in the SVA.

System Action: SM2 terminates.

Programmer Action: Increase the main storage available to SM2 by increasing either the EXEC SIZE parameter value, the STORAGE option value, or the partition size.

7C21I OVERLAPPING WORK EXTENTS ON SYS(y) AND SYS(y)

Explanation: Two work extents have been specified which overlap each other.

System Action: The program terminates.

Programmer Action: Redefine the named work extents. Check all other work files for overlapping extents: you will only get a message for the first detected, if there are several.

7C22I FBA SORT WORK AREA IS LARGER THAN 2000 MB

Explanation: Work space has been defined on FBA devices and is larger than 2000 megabytes, the maximum that SM2 can handle.

System Action: SM2 terminates.

Programmer Action: Reduce the work space assignment, if possible. Alternatively, if so much space is really needed, allocate some or all of it on CKD devices.

7C23I FBA SORT WORK EXTENT IGNORED, SMALLER THAN 32KB

Explanation: A work space extent has been defined on FBA which is smaller than 32K bytes, the minimum that can be used by SM2.

System Action: SM2 continues trying to sort in the space available to it, ignoring the extent which is too small.

Programmer Action: If the program subsequently terminates for lack of work space, allocate more and rerun.

7C24I MODAL RECORD LENGTH ASSUMED = xxx BYTES

Explanation: xxx is the l_s value from the RECORD statement LENGTH parameter, if supplied; or else the value for modal length calculated by SM2 (the average of l_2 and l_4).

System Action: None.

Programmer Action: None.

7D01I TOO MANY RECORD LENGTH PARAMETERS

Explanation: The number of length parameters specified in the RECORD statement exceeds the maximum allowed. For fixed-length records the number of parameters allowed is three (l_4 - l_3) and for variable-length records five (l_4 - l_5).

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the LENGTH keyword in the RECORD statement for too many parameters.

7D02I KEYWORD(S) IN INPFIL|OUTFIL STATEMENT IS(ARE) IGNORED

Explanation:

1. When VSAM/KSDS, ESDS, or RRDS is specified, all other parameters in the INPFIL/OUTFIL statement are ignored exept for EXIT and TOL. EXIT overrides VSAM.

2. When EXIT is specified all other parameters in the INPFIL/OUTFIL statement are ignored exept DATA in the INPFIL statement.

System Action: Processing continues.

Programmer Action: None

7D03I INVALID INCLUDE/OMIT DELIMITER

Explanation: A punctuation error has been detected in the INCLUDE or OMIT control statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check for operands that are incorrectly split between first and continuation cards.

7D04I NO VALID ALTSEQ STATEMENT FOUND

Explanation: An ALTSEQ statement must be specified when format AQ is specified in the SORT, MERGE, INCLUDE, or OMIT statement(s).

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Supply the missing statement.

7D05I xxxxxx KEYWORD IGNORED

Explanation:

1. When MERGE is specified the keyword represented by xxxxxx is ignored.

2. If both SUM and EQUALS are specified, EQUALS is ignored. Note that EQUALS can be the default, if this has been reset for your installation. To suppress this message, specify NOEQUALS.

3. If ADDROUT is specified when SUM and/or OUTREC is specified ADDROUT is ignored.

4. If SPAN is specified with fixed-length records, ASCII records, or ADDROUT specified, SPAN is ignored.

System Action: Processing continues.

Programmer Action: Remove unwanted parameter before next run.

7D06I BOTH INCLUDE AND DELBLANK DEFINED

Explanation: An INCLUDE statement and the DELBLANK parameter on the RECORD statement have both been found.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: The DELBLANK parameter can be used on its own or with an OMIT statement, but not with an INCLUDE statement. Remove the conflicting specification.

7D07I RECORD DESCRIPTOR WORD NOT INCLUDED

Explanation: When the OUTREC statement is being used to reformat variable-length records, the Record Descriptor Word (RDW; bytes 1-4 of a variable-length record) must be included in the reformatted output record. In other words, for variable-length records, the first entries in the FIELDS parameter of the OUTREC statement must be 1,n where n is greater than or equal to 4.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Ensure that the first entries in the FIELDS parameter of the OUTREC statement are correct.

7D08I xxxxxx FIELD BEYOND RECORD

Explanation: A field specified in an OUTREC or SUM statement extends beyond the end of the shortest record. The minimum record length is defined by the LENGTH parameter of the RECORD statement: l_1 for fixed-length records or l_4 as specified for variable-length records.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check OUTREC and SUM statements for incorrectly specified field position or length. Check RECORD statement for incorrect record length specification.

7D09I ADDROUT OPTION INVALID

Explanation: ADDROUT has been specified when the EXIT keyword is present in the INPFIL statement. You cannot have the ADDROUT option when you read all input to the program via an exit.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: The combination is invalid so choose either the ADDROUT option or the INPFIL EXIT specification.

7D10I INVALID LENGTH IN REL COND n - INCLUDE/OMIT

Explanation: The length of the n-th relational condition of the INCLUDE or OMIT statement is invalid:
- The length (parameter m) is not a decimal number, or
- It is a negative number, or
- It is greater than 256.

System Action: SM2 is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check INCLUDE/OMIT statement for invalid specifications.

7D11I xxxxxx EXIT REQUIRES Eyy

Explanation: xxxxxx is replaced in the message by INPFIL or OUTFIL; yy is the number of the exit required.

1. When EXIT is specified in the INPFIL statement, E15 (Sort) or E32 (Merge) must be specified in the MODS statement.

2. When EXIT is specified in the OUTFIL statement, E35 must be specified in the MODS statement.

System Action: SM2 terminates after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check appropriate control statement for invalid keyword operand.

7D12I INVALID FORMAT FOR ASCII DATA

Explanation: The value assigned to f (control field format) must be AC, ASL, or AST for ASCII data.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check value assigned to f in SORT/MERGE and INCLUDE/OMIT statements.

7D13I DELBLANK POSITION BEYOND 4092

Explanation: The DELBLANK parameter of the RECORD statement lies beyond byte 4092 of the record.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check DELBLANK parameter value.

7D14I EFFECTIVE L3 VALUE = xxxxxx

Explanation: This message is issued when an OUTREC statement has been supplied, and output record length (l_3 in the LENGTH parameter of the RECORD statement) was not specified, or specified incorrectly. xxxxxx is a decimal number specifying the total length of the reformatted record, including all fields and alignment padding.

If variable-length records are being reformatted with OUTREC, and if the variable portion of the record is included in the reformatted record, xxxxxx gives the maximum length of the output records.

System Action: If l_3 was incorrectly specified in the RECORD statement (in which case message 7D38I will have been issued), sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments. Otherwise, normal processing continues.

Programmer Action: Check RECORD statement for l_3 or l_4 which are invalid for an OUTREC statement, or remove unwanted OUTREC statement. Note that unless you change the record length at E35 (after OUTREC), it is not necessary to specify l_2 or l_3, as SM2 will calculate the correct values for you.

7D15I SUM FIELD n OVERLAPS CONTROL FIELD m

Explanation: The n-th field defined in a SUM statement overlaps the m-th control field defined in a SORT or MERGE statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Correct the SUM statement; also check that the SORT or MERGE statement specifies the control fields correctly.

7D16I SUM FIELD n OVERLAPS RECORD DESCRIPTOR WORD

Explanation: The n-th field defined in a SUM statement overlaps the Record Descriptor Word (RDW) of the variable-length records being processed.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Correct the SUM statement; also check that the RECORD statement is correct.

7D17I SUM FIELD n OVERLAPS SUM FIELD m

Explanation: The n-th field defined in a SUM statement overlaps the m-th field.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the SUM statement for errors.

7D18I TOO MANY VOLUME POSITIONAL PARAMETERS

Explanation: The number of positional parameters in the VOLUME keyword of the INPFIL statement exceeds the FILES parameter value specified in the SORT/MERGE statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Correct the VOLUME parameter or the number of files specified on the SORT/MERGE statement.

7D19I INVALID SELF DEF TERM IN REL COND n - INCLUDE/OMIT

Explanation: The self-defining term is invalid in the n-th relational condition of the INCLUDE or OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the INCLUDE/OMIT statement for invalid specifications.

7D20I OUTREC FIELD n INVALID VALUE

Explanation: A positional parameter was specified as the last numerical entry in the OUTREC FIELDS keyword, i.e. with no corresponding length indication; and the record type specified in the RECORD statement is fixed. This usage is only allowed with variable-length records.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the OUTREC statement for invalid specifications; check for commas or entries missing from the FIELDS parameter.

7D21I SPECIFIED Exx VALID ONLY FOR VSAM FILE

Explanation: The specified program exit is only allowed when a VSAM file is used. The exit is ignored.

System Action: Processing continues. Sort/merge ignores the exit.

Programmer Action: None

7D22I INVALID FORMAT IN REL COND n - INCLUDE/OMIT

Explanation: The format is invalid in the n-th relational condition of the INCLUDE or OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the INCLUDE/OMIT statement for invalid specifications.

7D23I INSUFFICIENT STORAGE FOR INCLUDE/OMIT FUNCTION

Explanation: Insufficient main storage is available to contain SM2 plus the code generated for the INCLUDE or OMIT function.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Increase main storage available to SM2.

7D24I PHASE 1 EXITS IGNORED BY MERGE

Explanation: Phase 1 of SM2 is not used for a merge-only operation, therefore any Phase 1 exits specified in the MODS statement of a merge-only operation are ignored.

System Action: Sort/merge continues, ignoring exits specified.

Programmer Action: Make sure the application was set up properly before next run.

7D25I EXIT E32 OR E38 IGNORED BY SORT

Explanation: Exits E32 and E38 are available only for a merge-only operation. They are ignored when specified in the MODS statement of a sort operation.

System Action: SM2 continues, ignoring exits specified.

Programmer Action: Make sure the application was set up properly before next run.

7D26I PRIORITY PARENTHESIS MISPLACED - INCLUDE/OMIT

Explanation: A parenthesis has been found in a syntactically invalid position in the COND parameter of an INCLUDE or OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the INCLUDE/OMIT statement for invalid syntax, paying special attention to the parentheses.

7D27I INCLUDE/OMIT FORMAT INVALID

Explanation: An invalid format was specified in the FORMAT keyword of an INCLUDE/OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check INCLUDE/OMIT statement for invalid FORMAT parameter. Except for FL, you can use any format valid for a SORT statement.

7D28I INVALID CONDITION IN REL COND n - INCLUDE/OMIT

Explanation: The condition is invalid in the n-th relational condition of the INCLUDE or OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the INCLUDE/OMIT statement for invalid specifications.

7D29I ERROR IN LENGTH VALUE

Explanation: An error has been detected in either the BUFOFF, the BLKSIZE, or one of the RECORD length parameters. The message is issued when:

1. l_5 is greater than l_4 (RECORD statement).
2. BLKSIZE is greater than 9999 with DATA=A (INPFIL and OUTFIL statements).
3. BUFOFF not equal to 0 when DATA=E (INPFIL statement).
4. BUFOFF not equal to 0 when DATA=A and RECORD TYPE=F (OUTFIL statement).
5. BUFOFF not equal to 4 when RECORD TYPE=D (OUTFIL statement).

Programmer Action: Check the RECORD statement for invalid length values. Check INPFIL/OUTFIL statements for invalid block size or invalid BUFOFF values.

7D30I L1 VALUE INVALID

Explanation: l_4 value must be greater than four for variable-length records.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check RECORD statement for missing or invalid l_4 value.

7D31I DATA=A INVALID

Explanation:

1. ASCII data (DATA=A or RECORD TYPE=D) is not allowed when ADDROUT is specified.

2. DATA=A is not allowed when RECORD TYPE=V.

3. ASCII data is not allowed with disk input/output.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Change to correct combination of RECORD TYPE, ADDROUT, DATA, and device.

7D32I ALTERED RECORDS REQUIRE OUTREC OR EXIT E15/E35

Explanation: If the RECORD statement indicates that record length will be modified (the l_1, l_2, and l_3 values are not the same), then either an OUTREC statement must be provided, or program exits E15 and/or E35 must be specified in the MODS statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check RECORD and MODS statements for inconsistency. Check for missing OUTREC statement.

7D33I xxxxxx BLOCK SIZE = yyyy BYTES

Explanation: No block size has been specified for xxxxxx (either INPFIL or OUTFIL), so it is assumed to be yyyy, calculated as follows:

- For EBCDIC fixed-length records, block size equals record length.
- For EBCDIC variable-length records, block size equals record length * 4.
- For ASCII fixed or variable-length records, block size equals record length * buffer offset.

Warning: If the assumed value is not reasonably valid, performance reduction or job termination may result.

System Action: Sort/merge continues with the assumed value.

Programmer Action: Check block size parameter to see if appropriate for next run of application. If not, make appropriate change on INPFIL/OUTFIL statement.

7D34I RECORD CONFLICTS WITH xxxxxx BLKSIZE

Explanation: xxxxxx is replaced by INPFIL or OUTFIL. The block size specified in the INPFIL or OUTFIL statement must be consistent with the record length specified in l_1 or l_3.

- If the OUTREC statement is in use, the OUTFIL block size must be consistent with the effective length of the reformatted record (including padding, if any; the effective length is given by message 7D14I).
- For EBCDIC fixed-length records, block size must be an exact multiple of record length.
- For EBCDIC variable-length records, block size must be at least record length + 4.
- If ASCII input data was specified, block size must be the sum of the block prefix and an exact multiple of the record length.
- If ADDROUT is specified with variable-length records, the rules for fixed-length records apply for l_3.

System Action: If the conflict is between EBCDIC records and block size, or between ASCII output records and block size, SM2 is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

If the conflict is between ASCII input records and block size, normal processing continues.

Programmer Action: Check RECORD statement and INPFIL or OUTFIL statements for inconsistency in specifying lengths.

7D35I MISSING FORMAT IN REL COND n - INCLUDE/OMIT

Explanation: The format specification is missing from the n-th relational condition of the INCLUDE or OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check INCLUDE/OMIT statement for missing 'f' subparameter of COND.

Note: If DATA=A or RECORD TYPE=D is specified only formats AC, AST, or ASL are allowed.

7D36I DELBLANK POSITION BEYOND yy

Explanation: The DELBLANK parameter of the RECORD statement extends beyond the end of the minimum record length yy. The minimum record length is defined by the RECORD statement LENGTH parameter l_1 for fixed-length records when no E15 is specified, or l_2 if E15 is specified. For variable-length records l_4 specifies the minimum record length.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check RECORD statement for incorrect record length specification or incorrect DELBLANK specification.

7D37I SYNTAX ERROR - INCLUDE/OMIT

Explanation: A syntax error has been detected in the INCLUDE/OMIT control statement. Common syntax errors are:

- Unbalanced parentheses
- Missing commas
- Embedded blanks
- Redundant operands
- Missing parameters

After issuing this message, SM2 continues to scan the statement for errors. Since the statement is in error, any messages issued later for this statement may be spurious.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check specified control statement for syntax error.

7D38I Lx INVALID FOR yyyyyy

Explanation: x in the message text is replaced by the length parameter number (3 or 4); yyyyyy is replaced by ADDROUT or OUTREC.

1. If ADDROUT is specified in the OPTION statement, the value assigned to l_3 in the RECORD statement must be equal to the length of the disk address. Disk addresses are 10 bytes long for SAM files and 5 bytes long for VSAM files.

2. If the OUTREC statement is in use, l_3 and l_4 have been specified in the RECORD statement, and are not consistent with the specification in the OUTREC statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action:

1. Check RECORD statement for invalid l_3 value, or check OPTION statement for unwanted ADDROUT option.

2. Check RECORD statement for invalid l_3 or l_4 values, or remove unwanted OUTREC statement.

Note: Unless you change the record length after ADDROUT or OUTREC (at E35), it is not necessary to specify l_2 or l_3. SM2 will calculate the correct value for you.

7D39I SUM FIELD n LENGTH INVALID

Explanation: In the n-th field defined in a SUM statment, the length is invalid for the format specified.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the SUM statement for invalid specification. The permitted lengths are:

Format	Length
BI	2, 4, or 8 bytes
FI	2, 4, or 8 bytes
PD	1-16 bytes
ZD	1-18 bytes

7D40I FIELD OR VALUE GT 8 CHAR - INCLUDE/OMIT

Explanation: A field or value has been detected in the INCLUDE or OMIT statement which is greater than 8 characters. (This restriction does not apply to character strings enclosed in quotation marks.)

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check specified statement for field or value greater than 8 characters.

7D41I L4 GREATER THAN xx

Explanation: The minimum length specified or defaulted for input records must not be greater than the specified or defaulted maximum or modal lengths. xx is replaced by L1 or L5.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check RECORD statement for invalid l_1, l_4, and/or l_5 values of the LENGTH parameter.

7D42I INVALID FORMAT COMBINATION IN REL COND n - INCLUDE/OMIT

Explanation: The n-th relational condition in the INCLUDE or OMIT statement specifies a comparison which is invalid. Figure 6 (for field-to-field comparisons) and Figure 7 (for field-to-constant comparisons) show the valid comparisons.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the COND parameter of the INCLUDE/OMIT statement for invalid comparisons.

7D43I INVALID LOGICAL OPER IN REL COND n - INCLUDE/OMIT

Explanation: The logical operator is invalid in the n-th relational condition of the INCLUDE or OMIT statement. It can be EQ, NE, GT, GE, LT, or LE.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the INCLUDE/OMIT statement for invalid specifications.

7D44I TOO MANY xxxxxx KEYWORDS

Explanation: The maximum number of keywords that can be specified in the INCLUDE or OMIT statement has been exceeded.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check INCLUDE/OMIT statement for too many keyword operands.

7D45I INCLUDE/OMIT FIELD IN REL COND n BEYOND xxxxxx

Explanation: The field in the n-th relational condition of the statement is beyond byte 4092 of the record, or beyond the length specified in the l_2 value of the RECORD LENGTH parameter.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the INCLUDE/OMIT statement for invalid length or displacement value. Check the RECORD statement for incorrect l_2 value.

7D46I CONTROL FIELD n TOO LONG FOR TYPE

Explanation: A control field with packed decimal format (PD) exceeds 32 bytes. Control field number is represented by n.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check length and format of specified control field on SORT or MERGE statement.

7D47I EXIT Enn NOT GIVEN FOR NONSTANDARD LABELS

Explanation: If nonstandard labels are specified in the OPTION statement, exits E11, E17, E31, and/or E37 must be specified in the MODS statement. Enn is replaced by the exit number.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check MODS statement for omitted exit and OPTION statement for incorrect label specification.

7D48I INVALID FIELD POSITION IN REL COND n - INCLUDE/OMIT

Explanation: The field position is invalid in the n-th relational condition of the INCLUDE or OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the INCLUDE/OMIT statement for valid specifications.

7D49I CONTROL FIELD xx BEYOND RECORD LENGTH

Explanation: A control field specified in the FIELDS parameter of the SORT or MERGE statement extends beyond the end of the record. The control field must not extend beyond l_2 (or l_4 when ADDROUT is specified). The control field number is represented by xx.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check SORT or MERGE statements for incorrectly specified control field displacement or length. Check RECORD statements for incorrectly specified or defaulted record length.

7D50I ALTSEQ CANNOT BE USED WITH DATA=A

Explanation: An alternative sequence cannot be used with ASCII data.

System Action: Sort/merge is terminated after phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Either change data type from DATA=A to DATA=E, or remove ALTSEQ statement.

7D51I INVALID WORK DEVICE

Explanation: A sort work file has been allocated on an invalid device, or more than two device types have been specified. The rules for mixed devices and the valid device types described in Chapter 1, 'Introduction', must be adhered to.

System Action: SM2 is terminated after phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the device types allocated and correct them.

7D52I xxxxxxx OPTION NOT VALID FOR yyyyyy DEVICE

Explanation: An option has been specified which is not applicable to the I/O device being used. xxxxxxx in the message text is replaced by one of the option names listed below; yyyy is replaced by the word 'INPFIL' or 'OUTFIL'. The message will be issued if any of the following parameters are specified:

1. OPEN/CLOSE for disk devices (INPFIL/OUTFIL statement).
2. NOTPMK for disk output device (OUTFIL statement).
3. VERIFY for tape output device (OPTION statement). Note that the VERIFY option can have been made the default for your installation. In that case you will need to specify NOVERIFY in order to suppress this message.
4. ADDROUT for FBA or tape input device (OPTION statement).
5. LABEL=U for disk devices (OPTION statement).
6. NOTPMK if LABEL=U is not specified.

System Action: Sort/merge continues, ignoring the option specified.

Programmer Action: Check application to see if it was set up correctly before next run.

7D53I BLKSIZE TOO LARGE/SMALL FOR xxxxxx DEVICE

Explanation: Permitted maximum/minimum input and output block sizes are:

Device	Max bytes	Min bytes
Tape (input)	32767	1
Tape (output)	32767	18
2311	3625	1
2314 or 2319	7294	1
3330 or 3333	13030	1
3340 or 3344	8368	1
3350	19069	1
FBA	32761	1

When mixed input is used the device with the largest capacity determines the maximum allowable blocksize.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check INPFIL/OUTFIL for invalid block size specification.

7D54I ALTSEQ STATEMENT IGNORED

Explanation: An ALTSEQ statement has been found, but no field was specified as format AQ.

System Action: The ALTSEQ statement is ignored, and normal processing continues.

Programmer Action: Check SORT, MERGE, INCLUDE, and OMIT statements for incorrect field format specification, or remove unwanted ALTSEQ statement.

7D55I EXIT ADDRESS OUTSIDE PARTITION

Explanation: The loading information in the MODS statement is invalid. The absolute loading address is outside the partition.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the loading information in the MODS statement for invalid value and correct the absolute loading address for the exit routine, or increase the SIZE parameter on the EXEC card or STORAGE parameter in the OPTION statement.

7D56I TOTAL LENGTH OF CONTROL FIELDS > 256 BYTES

Explanation: The total length of the SORT/MERGE control fields fields must not exceed 256 bytes.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check the SORT or MERGE statement for invalid length specification in the fields parameter.

7D57I INVALID INCLUDE/OMIT KEYWORD

Explanation: An invalid or duplicate keyword has been detected in an INCLUDE or OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check appropriate control statement for invalid or duplicate keyword.

7D58I INCLUDE/OMIT COND KEYWORD MISSING

Explanation: The COND keyword is missing from an INCLUDE or OMIT statement.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check appropriate control statement for missing COND keyword.

7D59I GEN xxxxxxx ROUTINE GREATER THAN 4K BYTES

Explanation: The routines generated by SM2 to perform the record selection function specified in an INCLUDE/OMIT statement or the summary function specified in a SUM statement have a total length greater than 4096 bytes.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Reorganize the statement concerned to require less generated code. Appendix B gives information on lengths of generated code.

7D60I OUTREC FIELD CONTAINS ONLY RDW

Explanation: The OUTPUT record specified in the OUTREC statement contains only the first four bytes. This is not allowed for variable-length records, where the first four bytes are the RDW (record descriptor word). At least one data byte must be included from the fixed data part of the record.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Correct your OUTREC statement.

7D61I INPUT RECORD LENGTH OR BLKSIZE < 12 BYTES

Explanation: The input record length specified in the RECORD statement, or BLKSIZE in the INPFIL statement, is smaller than 12 bytes for tape input. This may result in physical input blocks which are less than 12 bytes, and if error recovery is attempted such a block may be lost. This will not be discovered by SM2.

System Action: Warning message; the program continues.

Programmer Action: None.

7D62I SPECIFIED Exx VALID ONLY FOR SAM FILE

Explanation: The specified exit is not allowed together with VSAM file processing.

System Action: Informational message. SM2 ignores the exit.

Programmer Action: Check your input/output assignment for correctness.

7D63I SPECIFIED E31 VALID ONLY WHEN CHKPT SPECIFIED

Explanation: If output is VSAM and E31 is specified, checkpoint must also be specified.

System Action: Informational message. Sort/merge ignores exit E31.

Programmer Action: Check whether E31 checkpointing is wanted.

7D64I OUTPUT RECORD LENGTH < 18 BYTES

Explanation: Output is on tape, and the output record length specified in the RECORD statement LENGTH parameter is less than 18 bytes for fixed-length records (l_3) or 14 bytes for variable-length record (l_4). Only when ADDROUT is specified can l_3 be 5 or 10 bytes. When OUTREC is used for variable record files the fixed portion of the output record must be at least 14 bytes.

System Action: Sort/merge is terminated after Phase 0 has completed its error checking of control statements and unit assignments.

Programmer Action: Check record length parameter or unit assignment for correctness.

7D65I xx VALUE IS IGNORED FOR MERGE

Explanation: An invalid value has been specified in the RECORD LENGTH parameter:

1. When EXIT and E32 are not specified: if l_4 is not equal to l_2, the l_2 value is overridden.
2. When EXIT and E32 are specified: if l_4 is not equal to l_2, the l_4 value is overridden.

System Action: Merge continues ignoring the l_4/l_2 value.

Programmer Action: None.

7D66I Lx VALUE TOO LARGE FOR SORTINy

Explanation:

Lx is either L1 or L4, representing a RECORD LENGTH value. y is the number of the SORTIN file (1-9).

1. When record TYPE=F, l_4 must not be larger than the maximum block size for the device specified by SORTINy.
2. When record TYPE=V, l_4 + 4 must not be larger than the maximum block size for the device specified by SORTINy.
3. When record TYPE=D, l_4 + BUFOFF must not be larger than the maximum block size for the device specified by SORTINy.

System Action: Terminate after Phase 0.

Programmer Action: Correct the L1 or L4 value.

7D67I INVALID CI SIZE - xxxxxx

Explanation: xxxxxx in the message text is replaced by INPUT or OUTPUT. The CI size for the file is not a multiple of 512; or, if >8K bytes, is not a multiple of 2K bytes.

System Action: SM2 terminates.

Programmer Action: Correct the CI size, probably incorrectly specified in the output DLBL statement. If the file in error is an input file you should check its labels for errors.

7D68I CI SIZE NOT SPECIFIED FOR FBA OUTPUT. (x) USED

Explanation: No CISIZE parameter was specified on the DLBL statement for an output FBA device. x is the size used.

System Action: SM2 continues, using the default value for the parameter. This is the next largest multiple of 512 that will hold the block size plus seven bytes. If the result is larger than 8K bytes, it is rounded up to the next multiple of 2K bytes.

Programmer Action: If the default is not satisfactory, add a CISIZE parameter to the output DLBL statement.

7D69I BLOCK SIZE DOES NOT MATCH xxxxxxx CI SIZE

Explanation: The control interval size specified for file xxxxxxx is incompatible with the block size you have specified for the file. It should be at least the block size plus seven bytes. However for mixed FBA and CKD input you can have a block size on CKD files which is larger than the CI size for the FBA files. The FBA and CKD block sizes must then be compatible in the usual way, that is:

- Variable length records -- not greater than specified block size (unless records are defined as spanned)
- Fixed-length records -- not greater than specified block size, and a multiple of input record size

System Action: SM2 terminates.

Programmer Action: Change your file definitions so that they are compatible.

| 7D70I ADDROUT INVALID WITH CI FORMAT INPUT

| Explanation: ADDROUT is not supported for FBA input files; nor
| for VSAM managed SAM input files, unless accessed as VSAM.

System Action: The program terminates.

Programmer Action: Plan the job differently.

7D71I BYPASS IGNORED FOR FBA I/O ERRORS

Explanation: Self-explanatory.

System Action: The program continues. If there is an I/O error on an FBA file the program will terminate.

| Programmer Action: None.

| 7D72I LABELS SET STANDARD FOR MANAGED xxxxxxx

| Explanation: LABEL=U or LABEL=N was specified for files xxxxxxx
| which is a VSAM managed SAM file. Since these files must be
| standard labelled files the program ignores the LABEL parameter
| and assumes that the labels are standard.

| System Action: The program continues.

| Programmer Action: Remove the LABEL specification from the
| OPTION card for this file.

7E01I INVALID SIGN

Explanation: A control field format with separate sign (CST, CSL, AST or ASL) has an invalid value in the sign byte. The valid hexadecimal values for EBCDIC input are '4E' for + and '60' for -. For ASCII input the valid hexadecimal values are '2B' for + and 'RD' for -

System Action: Sort/merge terminates after message 7J02I has been printed.

Programmer Action: Check field format descriptions in the SORT or MERGE statement.

7E02I xxxxxxxx HAS WRONG SVA STATUS

Explanation: The module xxxxxxxx must reside in the same place as module ILUSOPT.

System Action: Sort/merge terminates.

Programmer Action: Store the module in the same place as ILUSOPT, either both in or both out of the SVA.

7E03I INPUT FILE TOO LARGE FOR IN-CORE SORT

Explanation: The input file is not exhausted but the record storage area is full and no work file is specified.

System Action: SM2 terminates.

Programmer Action: Either allocate more main storage or specify some work file space. See Appendix B.

7E04I IN-CORE SORT

Explanation: DIAG message. The complete input file was sorted in main storage, without using work files.

System Action: None.

Programmer Action: None.

7E05I PH1: WORK BUFFERS PFIXED

Explanation: DIAG message. The work buffer area is fixed in phase 1.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7E06I PH1: WORK BUFFERS NOT PFIXED

Explanation: DIAG message. The work buffer area is not fixed in phase 1.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7E21I I/O ERROR ON SORTWORK CCB=xx...x

Explanation: This message is generated when a permanent I/O error occurs on a work file. xx...x is replaced by 16 bytes of the Channel Control Block (CCB) or IORB.

System Action: SM2 terminates.

Programmer Action: None.

Operator Action: Rerun job with the DUMP option. If the error persists call IBM.

7E22I INSUFFICIENT WORK SPACE

Explanation: All space on the work file is used and the input file has not been exhausted.

System Action: Sort/merge terminates after message 7J02I or message 7J13I has been printed.

Programmer Action: Check the input file size against the record count given in message 7J02I or message 7J13I. Rerun the sort task with more work storage space allocated. See Appendix B for details of work storage requirements.

7E80I - 7E84I

Explanation: These are program error messages. Details of the messages, special procedures, and recommended actions are listed at the end of this appendix under the heading 'Program Error Messages'.

7F02I xxxxxxxx HAS WRONG SVA STATUS

Explanation: The module xxxxxxxx (ILUSPARI or ILUSPAR) must reside in the same place as module ILUSOPT.

System Action: SM2 terminates.

Programmer Action: Store ILUSPARI and ILUSPAR in the same place as ILUSOPT, either all in or all out of the SVA.

7F03I { PH1 / PH2: TIME A: yy SEC, TIME B: zz SEC / PAR }

Explanation: DIAG message. Time A (yy) is the difference in the values returned by the GETIME macro at the start and end of the Phase. Time B (zz) is one of the following:

- If Job Accounting is supported in your system, zz is the difference in the values obtained at the start and end of the Phase from the CPU time counter (ACCTCPUT field) in the Job Accounting Interface Partition Table.
- If you do not have Job Accounting, zz is the difference in the values returned at the start and end of the Phase by the TTIMER macro.

System Action: None.

Programmer Action: None.

7F04I PH1: xxx RECORD BLOCKS

Explanation: DIAG message. xxx record blocks were written in Phase 1.

System Action: None.

Programmer Action. None.

7F05I PH2: xxx INPUT, yyy INDEX, zzz WRITE BACK BLOCKS

Explanation: DIAG message. Input to Phase 2 was xxx record blocks read via an index of yyy index blocks. zzz write back blocks were produced.

System Action: None.

Programmer Action. None.

7F07I PAR: WORK BUFFERS PFIXED

Explanation: DIAG message. The work buffer area is fixed in the partitioning part of phase 2.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7F08I PAR: WORK BUFFERS NOT PFIXED

Explanation: DIAG message. The work buffer area is not fixed in the partitioning part of phase 2.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7F10I PH2: MERGE ORDER xx

Explanation: DIAG message. Actual Phase 2 merge order.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how to use the DIAG messages.

7F11I PH3: MERGE ORDER xx

Explanation: DIAG message. Actual Phase 3 merge order.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how to use the DIAG messages.

7F21I I/O ERROR ON SORTWORK CCB=xx...x

Explanation: This message is generated when a permanent I/O error occurs on a work file. xx...x is replaced by 16 bytes of the Channel Control Block (CCB) or IORB.

System Action: SM2 terminates.

Programmer Action: None.

Operator Action: Rerun the job with the DUMP option. If the error persists call IBM.

7F22I INSUFFICIENT WORK SPACE

Explanation: All space on the work file is used. More space is needed for index handling.

System Action: SM2 terminates.

Programmer Action: Rerun the sort task with more work storage space allocated. See Appendix B for information on work storage requirements.

7F23I PAR: xxx PHYSICAL, yyy LOGICAL STRINGS

Explanation: DIAG message. Input to the partitioning part of Phase 2 was xxx physical strings and the output was yyy logical strings. yyy=0 means that logical strings were not built.

System Action: None.

Programmer Action: None.

7F81I-7F83I

Explanation: These are program error messages. Details of the messages, special procedures, and recommended actions are listed at the end of this appendix under the heading 'Program Error Messages'.

7G02I xxxxxxxx HAS WRONG SVA STATUS

Explanation: Module xxxxxxxx must reside in the same place as module ILUSOPT.

System Action: SM2 terminates.

Programmer Action: Store the module in the same place as ILUSOPT, either both in or both out of the SVA.

7G05I PH2: WORK BUFFERS PFIXED

Explanation: DIAG message. The work buffer area is fixed in phase 2.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7G06I PH2: WORK BUFFERS NOT PFIXED

Explanation: DIAG message. The work buffer area is not fixed in phase 2.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7G21I I/O ERROR ON SORTWORK CCB=xx...x

Explanation: This message is generated when a permanent I/O error occurs on a work file. xx...x is replaced by 16 bytes of the Channel Control Block (CCB) or IORB.

System Action: Sort/merge terminates.

Programmer Action: None.

Operator Action: Rerun job with the DUMP option. If the error persists call IBM.

7G22I INSUFFICIENT WORK SPACE

Explanation: All space on the work file is used and more space is required.

System Action: SM2 terminates after message 7J02I has been printed.

Programmer Action: Rerun the sort task with more work storage space allocated. See Appendix B for information on work storage requirements.

7G23I PH2: ETTR SAVE LIST HAS xxx ENTRIES

Explanation: DIAG message. The rest of available storage forms a list for DASD block addresses. xxx is the number of entries in that list.

System Action: None

Programmer Action: None.

7G81I-7G83I

Explanation: These are program error messages. Details of the messages, special procedures, and recommended actions are listed at the end of this appendix under the heading 'Program Error Messages'.

7H02I xxxxxxxx HAS WRONG SVA STATUS

Explanation: Module xxxxxxxx must reside in the same place as module ILUSOPT.

System Action: SM2 terminates.

Programmer Action: Store the module in the same place as ILUSOPT, either both in or both out of the SVA.

7H05I PH3: WORK BUFFERS PFIXED

Explanation: DIAG message. The work buffer area is fixed in phase 3.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7H06I PH3: WORK BUFFERS NOT PFIXED

Explanation: DIAG message. The work buffer area is not fixed in phase 3.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7H21I I/O ERROR OR SORTWORK CCB=xxx...x

Explanation: This message is generated when a permanent I/O error occurs on a work file. xx...x is replaced by 16 bytes of the Channel Control Block (CCB) or IORB.

System Action: SM2 terminates.

Programmer Action: None.

Operator Action: Rerun the job with the DUMP option. If the error persists call IBM.

7H24I PH3: xxx BYTES UNUSED

Explanation: DIAG message. xxx is the number of bytes of available storage that remained unused in Phase 3.

System Action: None.

Programmer Action: See Chapter 6 for suggestions as to how the DIAG messages can be used.

7H81I-7H82I

Explanation: These are program error messages. Details of the messages, special procedures, and recommended actions are listed at end of this appendix under the heading 'Program Error Messages'.

7J01I SORT|MERGE COMPLETE, INSERT v, DELETE x, IN y, OUT z

Explanation: SM2 completed successfully. v is the number of records inserted at a program exit, x is the number deleted at an exit or by use of an INCLUDE, OMIT, or SUM statement, y the number received for processing by SM2, and z the number sent to output by SM2. This message will appear on SYSLOG, as well as on the printer to which messages are routed.

System Action: None.

Programmer Action: None.

7J02I SORT|MERGE ERROR, INSERT v, DELETE x, IN y, OUT z

Explanation: SM2 has terminated unsuccessfully. v is the number of records inserted at a program exit, x is the number deleted at an exit or by use of an INCLUDE, OMIT, or SUM statement, y the number received for processing by SM2, and z the number sent to output by SM2. This message will appear on SYSLOG, as well as on the printer to which messages are routed.

System Action: None.

Programmer Action: There will be another message giving the type of condition which has caused failure. The record counts in this message can be used in conjunction with the other message to pinpoint the cause of the problem.

7J03I SORT|MERGE COMPLETE, IN y, OUT z

Explanation: SM2 has completed successfully. y is the number of records received for processing by SM2, and z the number sent to output by SM2. This message will appear on SYSLOG, as well as on the printer to which messages are routed.

System Action: None.

Programmer Action: None.

7J04I VSAM CB ERROR (xx) AT aaaaaa

Explanation: A VSAM control block error has been detected. Error code xx (decimal) was received from a SHOWCB, TESTCB, GENCB or MODCB macro. The error was detected by SM2 at address aaaaaa.

If aaaaaa is zero, sort/merge was unable to load the necessary VSAM processor (possibly due to lack of virtual storage).

System Action: SM2 terminates.

Programmer Action: If the program is called, check that it is not overlaid by your code.

Look up the error code in the appropriate VSAM publication and take the action implied by the nature of the error. Check that the EXEC statement of the application specifies the SIZE parameter, giving a value small enough to leave room in the partition for VSAM. Check that REAL is not specified on the EXEC statement. If necessary call IBM for help.

Note: This message is mainly diagnostic and should not normally occur.

7J05I VSAM CLOSE ERROR yyyyyy (xx)

Explanation: A VSAM CLOSE module has returned an error code xx (decimal) while trying to close file yyyyyy.

System Action: Sort/merge continues if possible.

Programmer Action: Look up the error code in the publication DOS/VS Supervisor and I/O Macros or VSE/VSAM Messages and Codes, and take the action implied by the nature of the error.

7J06I ERASE IN PROGRESS

Explanation: Work files are being erased as requested.

System Action: None.

Programmer Action: None.

7J07I I/O ERROR DURING ERASE

Explanation: An I/O error occurred during erase of a work file.

System Action: SM2 terminates without completely erasing the work areas used.

Programmer Action: If work files must be erased take suitable action.

7J08I PH3: xxx INPUT, yyy INDEX, zzz WRITE BACK BLOCKS

Explanation: DIAG message. Input to phase 3 was xxx record blocks read via an index of yyy blocks. zzz write back blocks were produced.

System Action: None.

Programmer Action: None.

7J09I { PH1 / PAR / PH3 } TIME A: yy SEC, TIME B: zz SEC

Explanation: DIAG message. Time A (yy) is the difference in the values returned by the GETIME macro at the start and end of the phase. Time B (zz) is one of the following:

- If Job Accounting is supported in your system, zz is the difference in the values obtained at the start and end of the phase from the CPU time counter (ACCTCPUT field) in the Job Accounting Interface Partition Table.
- If you do not have Job Accounting, zz is the difference in the values returned at the start and end of the phase by the TTIMER macro.

System Action: None.

Programmer Action: None.

7J10I RCD COUNT OFF. INSERT v, DELETE x, IN y, OUT z

Explanation: This message is generated if the number of records leaving SM2 does not equal the number of records which entered, discounting any inserted or deleted. v is the number of records inserted at a program exit, x is the number deleted at an exit or by use of an INCLUDE, OMIT, or SUM statement, y the number received for processing by SM2, and z the number sent to output by SM2.

System Action: SM2 terminates.

Programmer Action: Check any routines at exits E15, E32, or E35. Rerun the job. If the error persists, call IBM.

7J11I SORTWK SPACE USED: xxx FBA BLOCKS

Explanation: Issued when work files are on FBA devices. xxx is the number of blocks actually used.

System Action: None.

Programmer Action: None.

7J12I SORTWK SPACE USED: xxx TRACKS ON yyyy

Explanation: Issued when work files are on CKD devices. xxx is the number of tracks actually used, and yyyy is the device type.

System Action: None.

Programmer Action: None.

7J13I SORT|MERGE ERROR, IN y, OUT z

Explanation: SM2 has terminated unsuccessfully because the number of records sent to output is not the same as the number of records which entered. This message will appear on SYSLOG, as well as on the printer to which messages are routed.

System Action: SM2 terminates.

Programmer Action: Rerun the job.

Operator Action: Rerun the job. If the error persists call IBM.

7J14I SORT CAPACITY APPROX xxx RECORDS

Explanation: xxx is the approximate number of records that can be sorted.

System Action: None.

Programmer Action: None.

7J15I SORT CAPACITY APPROX xxx RECORDS OF MODAL LENGTH

Explanation: xxx is the approximate number of records that can be sorted, on the assumption that SM2's figure for modal record length is correct. Modal record length is printed in message 7C24I.

System Action: None.

Programmer Action: None.

7K01I NOT POSSIBLE TO REACH SRCLIB

Explanation: A source statement library cannot be found from SM2, so no formatted dump can be produced.

System Action: None.

Programmer Action: Check your JCL for a missing ASSGN statement.

7K02I ILUCOMMA NOT FOUND IN SRCLIB

Explanation: The source statement library presently assigned does not contain ILUCOMMA. No formatted dump can therefore be produced.

System Action: None.

Programmer Action: See to it that ILUCOMMA is cataloged in this source statement library before next running SM2 with the DUMP option.

7K03I TRACE TABLE NOT USED

Explanation: No entries have been made in the trace table.

System Action: None.

Programmer Action: None.

7L01I INVALID SIGN

Explanation: A control field with a separate sign (format CST, CSL, AST or ASL) has an invalid value in the sign byte. The valid hexadecimal values for EBCDIC input are '4E' for + and '60' for -. For ASCII input the valid hexadecimal values are '2B' for + and 'RD' for -.

System Action: SM2 terminates after message 7J02I has been printed.

Programmer Action: Check field format descriptions in the SORT or MERGE statement.

7L02I OUT OF SEQUENCE ON SORTIN n

Explanation: Records on the SORTINn input file to a merge are out of sequence.

System Action: SM2 terminates after this message.

Programmer Action: Check the input file. Sort it if necessary.

7L03I xxxxxxxx HAS WRONG SVA STATUS

Explanation: Module xxxxxxxx must reside in the same place as module ILUSOPT.

System Action: Sort/merge terminates.

Programmer Action: Store the module in the same place as ILUSOPT, either both in or both out of the SVA.

7M01I I/O ERROR CCB/IORB = xx...x

Explanation: I/O error on a SAM file. In System/370 mode the 16-byte CCB is printed out; it includes the logical unit to identify the file. In ECPS:VSE the first 16 bytes of the IORB are printed. Details of the CCB and IORB can be found in the Physical IOCS section of the DOS/VS Supervisor and I/O Macros manual or the VSE/Advanced Functions Macro Reference.

System Action: Sort/merge terminates.

Programmer Action. None.

Operator Action: Rerun job with the DUMP option. If the error persists call IBM.

7M02I I/O ERROR - BYPASS

Explanation: An I/O error was encountered on a SAM input data set, but the BYPASS option was specified. This message is printed only once.

System Action: The block in error is bypassed.

Programmer Action. None.

7M03I WRONG LENGTH RECORD - xxxxxx, yyyyyy

Explanation: Either a record or a block has been detected in the input which is too long or too short for the current application:

1. If xxxxxx is LOGICAL then this means that a wrong length variable-length logical record was detected in the input. In that case, yyyyyy is the size of the record in question.

2. If xxxxxx is the name of the file, then a wrong-length block has been detected. Information concerning the length of the incorrect block is displayed in yyyyyy:

   - For a disk file, the length of the block in bytes, taken from the count field of the block.
   - For the tape with variable-length records, the length of the block in bytes, taken from the RDW.
   - For tape with fixed-length records:
     - for a short block, the number of bytes read in (calculated from the CCB count)
     - for a long block, the number of bytes read in (that is, the defined block size), plus one.
   - For VSAM managed SAM files
     - variable format, the length of the block in bytes taken from the RDW
     - fixed format, the length of the block in bytes returned to sort by SAM, for example, block length for short blocks. Too long blocks are truncated and not checked.

System Action: SM2 terminates.

Programmer Action.

1. Check the BLKSIZE parameter on the INPFIL statement.

2. Check the lengths specified on the RECORD statement. If these parameters are correct check that the input file corresponds to them.

3. In the case of a wrong-length record, also check any user routines you have at program exit E15 for incorrect record lengths.

7M04I WRONG LENGTH RECORD xxxxxx, yyyyyy

Explanation: As for 7M03I, but BYPASS was specified, and yyyyyy is always the length of the block in error. This message is printed only once for each file in which an error is detected.

System Action: The block is bypassed and SM2 continues.

Programmer Action. As for 7M03I.

7M05I OPEN ERROR - xxxxxx

Explanation: The file xxxxxx could not be opened.

System Action: Sort/merge terminates.

Programmer Action: Check that the JCL is correct and that necessary disks and tapes were mounted.

7M06I WRONG LENGTH RECORD EXIT Exx - %

Explanation: A wrong-length variable-length record was inserted at exit Exx. % is the length of the record in error.

System Action: SM2 terminates.

Programmer Action: Recode the routine at exit Exx.

7M07I SOME LOGICAL RECORDS ARE LESS THAN MIN LENGTH

Explanation: A variable-length record shorter than the specified or default minimum length has been found in the input.

System Action: SM2 continues.

Programmer Action: None.

7M08I xxxxxxxx HAS WRONG SVA STATUS

Explanation: The module xxxxxxxx must reside in the same place as ILUSOPT.

System Action: SM2 terminates.

Programmer Action: Store the module in the same place as ILUSOPT, either both in or both out of the SVA.

7M09I RETURN CODE ERROR Exx

Explanation: The user routine at program exit Exx returned an invalid code in the action word. The valid codes are 0, 4, 8, 12, 16 decimal (0, 4, 8, C, 10 hexadecimal). Note that when INPFIL or OUTFIL EXIT is specified or after input file EOF, the acceptable return codes are restricted. See the coding instructions given in Chapter 5.

System Action: SM2 terminates.

Programmer Action: Recode the user routine correctly. See the relevant exit coding instructions in Chapter 5 for details of valid return codes.

7M10I TOO SHORT RECORD FOUND - %

Explanation: A variable-length input record was found which was too short to contain all the sort control fields, all the SUM fields specified, or all the major INCLUDE/OMIT fields. % is the length of the record in error. (Note, that a negative value will be shown as the corresponding positive number).

System Action: SM2 terminates.

Programmer Action: Either reorganize the input file to leave out the short records, or remove the definition of the fields in question, to allow these functions with the short records. Remove any records with erroneous record descriptor words.

7M11I INVALID SIGN IN INCLUDE/OMIT INPUT

Explanation: This message is generated by the sort/merge when the sign of a separately signed numeric field is not a valid plus or minus.

System Action: SM2 terminates.

Programmer Action: Make sure format is correctly specified. Make sure that all input records contain signed data in the field specified in the SORT, MERGE, or INCLUDE/OMIT statement, and that the data is correctly recorded in the input records.

Operator Response: Make certain that the correct input file is mounted.

7M12I INPUT: REAL I/O NOT USED

Explanation: DIAG message. The optimization routine selected real I/O but sufficient main storage could not be obtained at the time buffers were to be allocated and page fixed. Virtual I/O was used. This can happen if a user exit is fixing buffer pages.

System Action: None.

Programmer Action: Increase real partition size or inspect user routines that fix pages.

7M13I xxxxxx BUFFERS NOT PFIXED

Explanation: DIAG message in ECPS:VSE mode. SM2 was unable permanently to fix the I/O areas for xxxxxx (input or output), so fix lists had to be used

System Action: None.

Programmer Action: To improve performance allow more real or page-fixable storage to the partition.

7M14I INPUT SEGMENTS IN WRONG ORDER xxxxxxx

Explanation: A first segment of a variable-length spanned record was detected in the input where a continuation segment was expected; or a continuation segment was detected where a first segment was expected. xxxxxxx is replaced by the file name.

System Action: SM2 terminates.

Programmer Action: Check the input file, and the program creating it.

7M15I SORTOUT FILE ON SYS(y) OVERLAPS WORK EXTENT ON SYS(y)

System Action: SM2 terminates.

Programmer Action: Respecify work or output extent.

7M16I xxxxxx BUFFERS PFIXED

Explanation: DIAG message in ESPC:VSE mode. The xxxxxx buffers (INPUT or OUTPUT) were page fixed.

System Action: SM2 continues normal processing.

Programmer Action: None.

7M17I SEGMENT LENGTH FIELD ERROR SEG=a, TOT=b, xxxxxxx

Explanation: An input segment was found with a length which makes record length greater than the l_4 value. a is the length of the segment found; b is the total length of the deblocked record including this segment; xxxxxxx is the file name.

System Action: SM2 terminates.

Programmer Action: Check the program which produces the input file.

7V01I VSAM OPEN ERROR yyyyyy (xx)

Explanation: File yyyyyy could not be opened. A VSAM OPEN module has returned an error code xx (decimal) from the ACB.

System Action: SM2 terminates.

Programmer Action: Look up the error code in the publication DOS/VS Supervisor and I/O Macros or VSE/VSAM Messages and Codes, and take the action implied by the nature of the error. If the error is a 'warning' message the program can be made to ignore it by using the TOL parameter on the INPFIL or OUTFIL statement as appropriate.

7V02I VSAM CB ERROR (xx) AT aaaaaa

Explanation: A VSAM control block error has been detected. Error code xx (decimal) was received from a SHOWCB, TESTCB, GENCB, or MODCB macro. The error was detected by sort/merge at address aaaaaa.

If aaaaaa is zero, sort/merge was unable to load the necessary VSAM processor (possibly due to lack of virtual storage).

System Action: SM2 terminates.

Programmer Action: If SM2 is called from another program, check that it is not overlaid by your code. Look up the error code in the publication DOS/VS Supervisor and I/O Macros or VSE/VSAM Messages and Codes, and take the action implied by the nature of the error. Check that the EXEC statement of the application specifies the SIZE parameter, giving a value small enough to leave room in the partition for VSAM. Check that REAL is not specified on the EXEC statement. If necessary call IBM for help.

Note: This message is mainly diagnostic and should not normally occur.

7V03I VSAM INPUT ERROR yyyyyy t (xx)

Explanation: A VSAM CLOSE module has returned an error code xx (decimal) for input file yyyyyy, the error was of type t (P for physical, L for logical). xx is the RPL FDBK code.

System Action: SM2 terminates.

Programmer Action: Look up the error code in the publication DOS/VS Supervisor and I/O Macros or VSE/VSAM Messages and Codes, and take the action implied by the nature of the error.

7V04I VSAM OUTPUT ERROR t (xx)

Explanation: A VSAM PUT module has returned an error code xx (decimal) for sort output file; the error was of type t (P for physical, L for logical).

System Action: SM2 terminates.

Programmer Action: Look up the error code in the publication DOS/VS Supervisor and I/O Macros or VSE/VSAM Messages and Codes, and take the action implied by the nature of the error.

7V05I VSAM CLOSE ERROR yyyyyy (xx)

Explanation: A VSAM CLOSE module has returned an error code xx (decimal) while trying to close file yyyyyy.

System Action: SM2 continues if possible.

Programmer Action: Look up the error code in the publication DOS/VS Supervisor and I/O Macros or VSE/VSAM Messages and Codes, and take the action implied by the nature of the error.

7V07I VSAM LOAD ERROR

Explanation: VSAM could not load its modules.

System Action: SM2 terminates.

Programmer Action: Make sure you have a GETVIS area and enough virtual storage for VSAM. Ensure that REAL is not specified on the EXEC statement.

Program Error Messages

SM2 has some 'built-in' self diagnostic code which checks certain parameters while the program is executing. If one of these checks fails, a program error message is produced. On the appearance of one of these messages you should inform your system programmer and rerun your job with the DUMP option specified. If the program error recurs call your IBM representative and provide him or her with the dump obtained. Possible temporary fixes to bypass the problem are suggested with some of the error messages listed below.

7E80I CODE OVERLAID BY BUFFERS

Explanation: A code overlay has occurred caused by an error in the optimization calculation.

System Action: SM2 terminates.

Programmer Action: Inform system programmer, rerun job with DUMP option, and call IBM representative.

7E81I ABNORMAL RETURN FROM REALAD

Explanation: An unacceptable return code was received from the REALAD macro when translating channel program address to absolute form.

System Action: SM2 terminates.

Programmer Action: Inform system programmer, rerun job with DUMP option, and call IBM representative.

Possible Bypass: The problem can probably be overcome temporarily by rerunning the job with the VIRT option specified.

7E82I PFREE ERROR

Explanation: An unacceptable return code was received from the PFREE macro.

System Action: SM2 terminates.

Programmer Action: Inform system programmer, rerun job with DUMP option, and call IBM representative.

Possible Bypass: The problem can probably be overcome temporarily by rerunning the job with the VIRT option specified.

7E83I ABNORMAL CODE OVERLAY 'RSA'

Explanation: There is not enough space in the RSA; (record storage area) ; fewer than three records can be stored.

System Action: SM2 terminates.

Programmer Action: Inform system programmer, rerun job with DUMP option, and call IBM representative.

System Action: SM2 terminates.

Programmer Action: Inform system programmer, rerun job with DUMP option, and call IBM representative.

Possible Bypass: Try increasing the amount of storage allocated to SM2 by altering the SIZE parameter on the EXEC statement.

7G83I MODULE OVERLAID BY TABLES

Explanation: A code overlay has occurred caused by an error in the optimization calculation.

System Action: SM2 terminates.

Programmer Action: Inform system programmer, rerun job with DUMP option, and call IBM representative.

7H81I ABNORMAL RETURN FROM REALAD

Explanation: An unacceptable return code was received from the REALAD macro when translating channel program address to absolute form.

System Action: SM2 terminates.

Programmer Action: Inform system programmer, rerun job with DUMP option, and call IBM representative.

7H82I WRITE BACK LIST FULL

Explanation: An abnormally large number of write back blocks has been created in Phase 3. There are too many for the program to handle.

System Action: SM2 terminates.

Programmer Action: Inform system programmer, rerun job with DUMP option, and call IBM representative.

Index

E

F

G

H

I

J

K

L

M

S

T

U

SC33-4044-2

DOS/VS Sort/Merge Version 2 Programmer's Guide (File S370-33 (DOS/VS)) Printed in U.S.A SC33-4044-2

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

DOS/VS Sort/Merge Version 2
Programmer's Guide

SC33-4044-2

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

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

List TNLs here:

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

Last TNL _____

Previous TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Reader's Comment Form

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

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

Fold and tape Please Do Not Staple Fold and tape

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

DOS/VS Sort/Merge Version 2 Programmer's Guide (File No. S370-33 (DOS/VS)) Printed in U.S.A SC33-4044-2