SERIES 60 (LEVEL 64)

SORT/MERGE

SUBJECT

Description of JCL Statements and Utility Command Parameters Associated with the Sort/Merge Facilities of GCOS Level 64

SPECIAL INSTRUCTIONS

Change bars indicate changes and asterisks indicate deletions.

SOFTWARE SUPPORTED

Level 64 GCOS Software Release 0400

ORDER NUMBER

AQ85, Rev. 1

September 1978

Honeywell

PREFACE

This manual presents the JCL statements and Data Services Language (DSL) associated with the sort/merge facilities of GCOS Level 64. Unless otherwise stated, the term "GCOS" as used in this manual refers to the General Comprehensive Operating Supervisor (GCOS) Level 64 software system.

Section I gives an overview of the sort/merge capabilities and an indication of the ways in which the command language can be used. The manual's subsequent organization represents the two levels of user interaction with these facilities. Section II presents the interaction at the JCL level. Here, resources are described and basic processing options selected. Sections III and IV present the DSL statements submitted as a COMFILE within the job description. These commands specify record-level functions based on the format and content of the user's files.

Appendices provide reference information. Appendix A provides a description of the sort work files. Appendix B describes the contents of the sort report provided for each sort run, Appendix C is a list of error messages, Appendix D describes the use of $SORTWORK and Appendix E lists the various collating sequences which are available. Appendix F gives guidelines for improving performance, Appendix G deals with the sorting of card files and subfiles, Appendix H describes the special characteristics of sorting large records, and Appendix I deals with the Checkpoint/Restart facility in sort.

Each section of this document is structured according to the heading hierarchy shown below. Each heading indicates the relative level of the text which follows it.

Level	Heading Format
1 (highest)	ALL CAPITAL LETTERS, UNDERLINED
2	Initial Capital Letters, Underlined
3	ALL CAPITAL LETTERS, NOT UNDERLINED
4	Initial Capital Letters, Not Underlined
5 (lowest)	ALL CAPITAL LETTERS FOLLOWED BY COLON: Text begins on same line.

The following notation conventions are used in this manual:

UPPERCASE	The keyword item is coded exactly as shown.
lowercase	Indicates a user-supplied parameter value.
[item]	An item within square bracket is optional.

{item1 item2 item3}	A column of items within braces means that one value must be selected if the associated parameter is specified. If the parameter is not specified the underlined item is taken as the default value.
( )	Parentheses must be coded if they enclose more than one item.
...	An ellipsis indicates that the preceding item may be repeated one or more times.

Note: All references to divice class MSU0350 in this manual include MSU0350 and MSU0351. Similarly, all references to MSU0400 include MSU0400, MSU0401, MSU0402 and MSU0452.

File No.: IN53

LEVEL 64 DOCUMENT LIST

Order Number	Title
AQ02	*Series 100 Program Mode Operator Guide*
AQ03	*Series 100 Conversion Guide*
AQ04	*Series 200/2000 Conversion Guide*
AQ05	*System 360/370 Conversion Guide*
AQ09	*System Management Guide*
AQ10	*Job Control Language (JCL) Reference Manual*
AQ11	*Job Control Language (JCL) User Guide*
AQ13	*System Operation Operator Guide*
AQ14	*System Operation Console Messages*
AQ18	*Operator Reference Manual*
AQ20	*Data Management Utilities Manual*
AQ21	*Series 200/2000 Program Mode User Guide*
AQ22	*Series 200/2000 Program Mode Operator Guide*
AQ26	*Series 100 File Translator*
AQ27	*Series 200/2000 File Translator*
AQ28	*Library Maintenance Manual*
AQ40	*System 3 Conversion Guide*
AQ49	*Network Control Terminal Operation Manual*
AQ50	*Terminal Operations Manual*
AQ52	*Program Checkout Facility Manual*
AQ53	*Communications Processing Facility Manual*
AQ55	*TDS/64 Standard Processor Site Manual*
AQ56	*TDS/64 User Guide*
AQ57	*Standard Processor Programmer Reference Manual*
AQ59	*Unit Record Devices User Guide*
AQ63	*COBOL User Guide*
AQ60	*Interactive Operation Facility*
AQ64	*COBOL Language Reference Manual*
AQ65	*FORTRAN Language Reference Manual*
AQ66	*FORTRAN User Guide*
AQ67	*FORTRAN Mathematical Library*
AQ68	*RPG Language Reference Manual*
AQ69	*RPG User Guide*
AQ72	*Series 200/2000 COBOL to Level 64 COBOL Translator*
AQ73	*IBM COBOL Translator*
AQ82	*BFAS User Guide*
AQ83	*HFAS User Guide*
AQ84	*UFAS User Guide*
AQ85	*Sort/Merge Manual*
AQ86	*Catalog Management Manual*
AQ87	*Library Maintenance User Guide*
AQ88	*I-D-S/II User Guide, Volume 1*
AQ89	*I-D-S/II User Guide, Volume 2*
AQ90	*COBOL Reference Card*
AQ92	*Operator's Reference Card*
AQ93	*RPG Reference Card*
AQ94	*FORTRAN Reference Card*

CONTENTS

ILLUSTRATIONS

TABLES

SECTION I

SORT AND MERGE FACILITIES

This manual describes the sort and merge utilities available to a user operating under GCOS. The design of these utilities incorporates facilities to minimize the task of job description and optimize the efficiency and flexibility of the sort or merge processes. The variety of optional parameters permit a user to tailor the sort or merge process to the requirements of the application.

The control language used to describe a sort or merge consists of Job Control Language (JCL) statements which describe the user's files and operating environment and a set of commands written in the Data Services Language (DSL) that describe functions based on fields within the user's input records.

SORT/MERGE CAPABILITIES

The sort capabilities documented here include the following features:

- Four output record formats
- Specification of up to 64 key fields per sort
- Summation of values in up to 16 fields of identically keyed records
- Presort record selection based on data values in the input record (up to 16 conditions can be specified).
- Field level record restructuring (up to 128 fields can be specified)
- Use of disk or tape for work files
- Use of EBCDIC, Series 100, ASCII or Series 200/2000 Collating sequence, or a privately defined collating sequence
- Sorting only part of a file
- Checkpoint/Restart facilities

- File concatenation on input (up to 9 files)

The merge capabilities documented here include the following features:

- Merging of up to eight input files
- Specification of up to 64 key fields per merge
- Summation of values in up to 16 fields of identically keyed records
- Premerge record selection based on input data values (up to 16 conditions can be specified)
- Field level record restructuring (up to 128 fields can be specified)
- Use of EBCDIC, Series 100, ASCII or Series 200/2000 collating sequence or a privately defined collating sequence

SORT/MERGE JCL STATEMENTS

Both the sort and merge processes are activated as job steps through extended JCL statements; the respective statements begin with $SORT and $MERGE. A sequence of parameters is associated with each of these statements. These are described in detail in Section II. Generally, these parameters specify files involved (input, output, work), describe the location of the Data Services Language statements, and select general job control processing options.

THE DATA SERVICES LANGUAGE (DSL)

The record level command file, written in DSL, must be provided as an input enclosure or as a previously established member of a source library. Statements in this file are described in Section III for the sort utility and Section IV for the merge utility. All but one of these statements are optional. The mandatory statement, KEYS, establishes the key fields for the sort or merge process.

The keys consist of one or more fields in the input record. For example, if the records are to be sorted according to an employee code contained in the first four bytes of the record and, within that sequence, according to the employee's age (contained in bytes 8 and 9), the user would establish a key composed of bytes 1 through 4 as the major key, and bytes 8 and 9 as the minor key. The sort utility would sort the input into sequence of the employee code and within a given value of the employee code in the sequence according to age. Likewise, the merge utility would use the major key (employee code) to merge records. When equality is reached on the major key then the records are merged in sequence according to the minor key (age).

SORT WORK FILE SPACE

The sort utility requires work file space in order to produce sorted output. This may be on either tape or disk . For a given sort, all work space must be on the same device class.

A disk work file may be permanent or temporary. In either case, if the work file is supported by more that one volume, the appropriate devices must all have the same attributes.

When magnetic tape work files are used, the tape volumes must have been prepared using the utility $VOLWORK. It is not necessary for the device attribute "density" to be the same for all tapes. The work tapes' capacity must be greater than the total input file size.

If no work space is explicitly requested then the sort utility will use a temporary file on the system disk (10 cylinders).

USING SORT WITH COBOL

In the COBOL language the programmer may invoke the sort utility in a dynamic manner, through the SORT verb. In this case, the key fields, input and output files, and other control information is supplied inside the invoking COBOL program. However, the user must supply information on the sort work space in the job step description of the executing program. The extended JCL statement $SORTWORK is used. For further details see Appendix D.

USING SORT WITH HFAS (Series 200/0)

Both Sort and Merge can be applied to Series 200/0 files. This manual deals mainly with the use of the utilities on BFAS files. For further details on processing Series 200/0 files see the publication "HFAS User Guide". The description in this manual on work space definition and DSL use apply also in the HFAS case.

SECTION II

SORT/MERGE JOB DESCRIPTION STATEMENTS

The JCL statements for sorting and merging files are constructed as in Figure 2-1. As this Figure snows, each utility requires the input, output, and command files to be specified. All other elements of the JCL statements are optional. Default values exist for the parameters, which, if they are appropriate to the application, allow the specification of the parameter to be omitted. These other elements relate to work files (sort only), reporting, performance optimization, error record processing, part-file processing, checkpointing, and abnormal termination. The various parameters are discussed on the following pages.

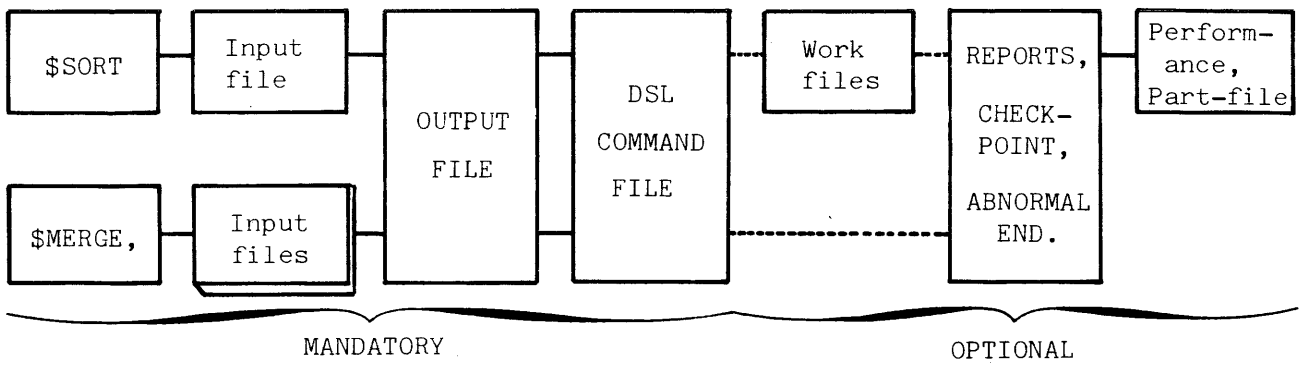


Figure 2-1. Construction of Sort and Merge JCL Statements.

Notes relating to SORT

1. Input and output files may be BFAS, HFAS, or UFAS independently of each other.

2. Input and output files may be on disk or tape independently (i.e., input disk file may produce output disk or tape file, and input tape file may produce output disk or tape file.)

3. Similarly, input and output files may be temporary or permanent independently.

4. The output may be written onto the input file ★ ((OUTFILE=(INFILE)).

5. Options are available to reduce the device requirements of a sort.

   . a single device can be reserved for the input file and the output file, provided, of course, the volumes supporting them are of the same device class (REPLIN).

   . for a tape sort, the same device can be used successively to process the input file, a work file and the output file, provided again that the volumes supported all belong to the same device class.

   These options, however, save drive reservation at the cost of increasing elapsed time, due to volume mounting and dismounting during the sort.

6. Output records are padded or truncated depending on the size of the output record compared with the input (or arranged) ★ record.

7. Tape Sort (work media is tape) : Unless USEOUT is specified (work tape subsequently used as output tape volume) it is assumed that all the tape volumes to be used in the sorting process have a work status, that is, have been obtained by using the $VOLWORK utility. The number of devices required to support work files for the sort must be specified through NBDV.
   When REPLIN, REPLOUT, or USEOUT is specified (see below), all the volumes involved should have the same device characteristics, including density.

8. Disk Sort (work media is disk) : If WKDISK is specified with the SIZE parameter option, a temporary file, named SORTWORK, ★ is allocated.

9. If neither WKTAPE nor WKDISK is specified, the sort uses a temporary work file of ten cylinders on the resident disk. The temporary file is allocated under the name SORTWORK.

Notes Relating to MERGE.

1. The file specified as INFILE1 is known as the primary file. Other input files, INFILE2 through INFILE8 must have the same record size and format as the primary file.

2. Input and output files may be BFAS, HFAS or UFAS independently of each other.

3. Any combination of BFAS, HFAS and UFAS files may be merged. ★

Format of $SORT Statement

Below is the generalized format of the $SORT JCL statement. Lower-case elements are expanded later in this section.

```
$SORT    INFILE = (file)

       [,INFILE1 = (file)
       [,INFILE2 = (file)
       [,INFILE3 = (file)
       [,INFILE4 = (file)
       [,INFILE5 = (file)
       [,INFILE6 = (file)
       [,INFILE7 = (file)
       [,INFILE8 = (file)]]]]]]]]

        ,OUTFILE =  {(file)    }
                    {(INFILE)  }

                    {*input enclosure name}
        ,COMFILE =  {(file)               }

       [, {WKTAPE[S] = (tape work file (s))}]
          {WKDISK[S] = (disk work file (s))}

       [,  REPORT = ( AUDIT )]
                    ( PARAM )
                    ( ALL   )
                    ( NONE  )

       [, PRTFILE = file]

       [, SIZE = nnnn]

       [, INVREC = ([ {ABORT   }]                          ]
                    {CONTINUE  }                           ]
                      [,ERROPT = ( PRINTID )]),            ]
                                 ( PRINT   )               ]
                                 ( LOG     )               ]
                                 ( IGNORE  )               ]

       [, LOGFILE = (file)]

       [, START = nnnnnnnnnn]

       [, HALT = nnnnnnnnnn]

       [, REPEAT]

       [, DUMP = {NO  }]  ;
                 {DATA}
```

Format of $MERGE Statement

Below is the generalized format of the $MERGE JCL Statement. Lower-case elements are expanded later in this section.

```
$MERGE   INFILE1 = (file), INFILE2 = (file)
        [,INFILE3 = (file) [,INFILE4 = (file)
        [,INFILE5 = (file) [,INFILE6 = (file)
        [,INFILE7 = (file) [,INFILE8 = (file)]]]]]]
        ,OUTFILE = (file)
                   {*input enclosure name}
        ,COMFILE = {(file)                }
        [,REPORT = {AUDIT}]
                   {PARAM}
                   {ALL  }
                   {NONE }
        [,PRTFILE = (file)]
        [,BUFFER = nnn]
        [,DUMP =  {NO  }]  ;
                  {DATA}
```

Specifying files in $SORT and $MERGE

The files specified in the $SORT and $MERGE JCL statements have many common elements, as illustrated by Figure 2-2.

Use Figure 2-2 to specify files as follows:

INFILE Boxes 1,2 and 4 apply.

INFILEn If $MERGE boxes 1, 2 and 4 apply.
         If $SORT box 1 applies.

OUTFILE Boxes 1,2,3,4,5 apply.

PRTFILE Boxes 1 and 3 apply.

LOGFILE Boxes 1,2 and 3 apply.

COMFILE Box 1 applies. (Note that SUBFILE parameter is mandatory)

```
external-file-name

[ ,FILESTAT = { CAT    } ]
              { UNCAT  }
              { TEMPRY }
              { IN     }
[,CATALOG = n]

[,SHARE = { NORMAL    }]
          { ONEWRITE  }
          { DIR       }

[ ,END = { DEASSIGN } ]
         { PASS     }
         { UNLOAD   }

[,FSN = nnn]

[, ABEND = { DEASSIGN } ]
           { PASS     }
           { UNLOAD   }

[,SUBFILE = member-name]
[ ,LABEL = { NATIVE   } ]
           { H200     }
           { H200E80  }
           { H200C80  }
           { H200C120 }
           { NONE     }
           { H200NONE }
[,MOUNT = nn]

[,FIRSTVOL = nn]
[,LASTVOL = nn]
[ , { RESIDENT                                                                    } ]
    { DEVCLASS = device-class-name [ ,MEDIA = ( { WORK                    } )] }
                                              { media-name [,...]         }
```

BOX 1

Figure 2-2. File Specification in $SORT and $MERGE

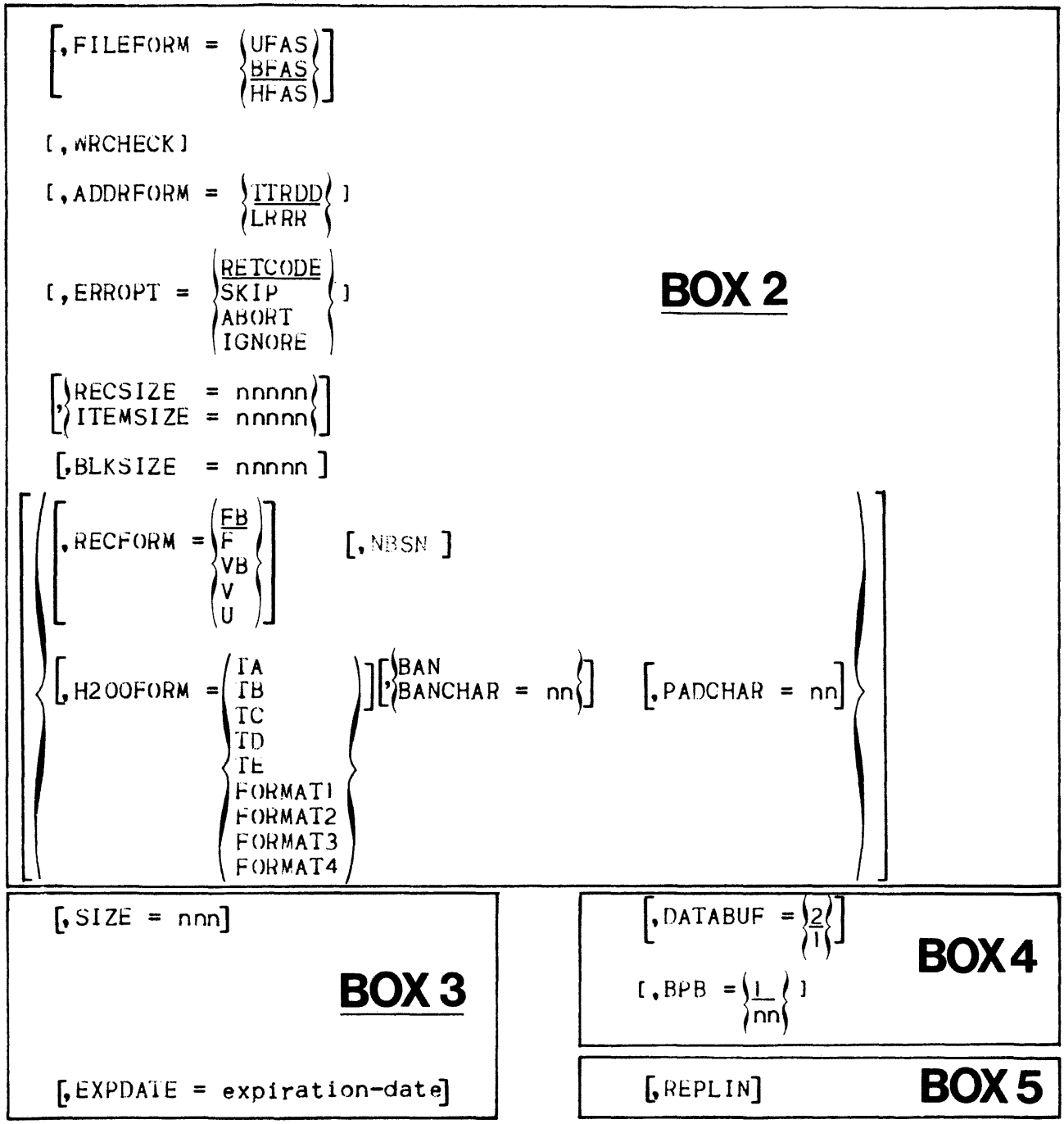


Figure 2-2. File Specification in $SORT and $MERGE (Cont.)

Specifying Sort Work Files

Workfiles are specified in the $SORT Statement as shown in Figure 2-3.

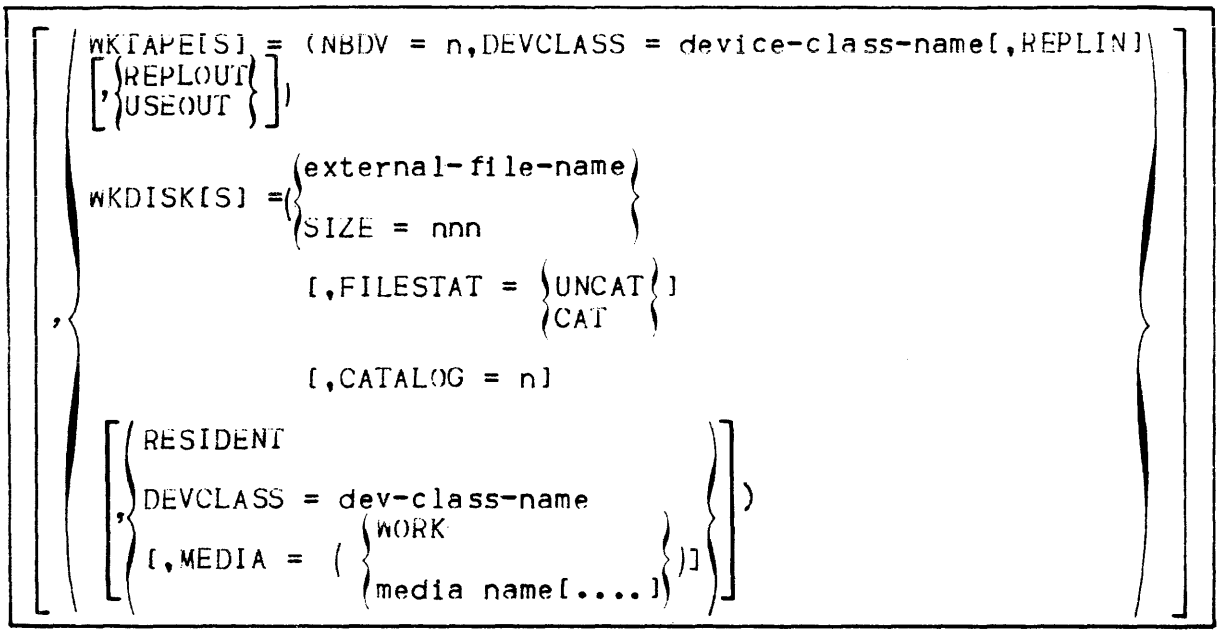


Figure 2-3. Work File Specification in $SORT

File Descriptions Parameters

external-file-name — as this is a positional parameter, it must appear first in the parameter string. If the file is temporary, the maximum length is 16 characters. If permanent, its maximum length is 44 characters. There is no default value.

FILESTAT — = TEMPRY signifies that the file is temporary. If specified for an input file, the file must have been passed by a previous step. If specified for an output file, the file must be passed to a subsequent step.
= UNCAT signifies that the file is permanent and not cataloged.
= CAT signifies that the file is cataloged.
= IN signifies that the file is an input enclosure (SUBFILE must then be specified).

CATALOG — This is used to override the normal catalog search rules when several private catalogs have been specified in an $ATTACH JCL statement. Normally a file is searched for in all the attached catalogs, starting from the first one specified in the $ATTACH statement. However,

when the CATALOG = n option is used only the nth catalog specified in the $ATTACH statement is searched.

SHARE

This specifies the sharing level requested for the file. The types of access made by SORT or MERGE are as follows:
Read for INFILE, INFILEn and COMFILE.
Write for OUTFILE, PRTFILE and LOGFILE.
= NORMAL allows any number of readers, or a single writer and no concurrent readers. This is the default value.
=ONEWRITE allows any number of readers and a single writer to operate on the file concurrently.
= DIR only applies to libraries. The library may be assigned to any number of users; however, subfile sharing is only allowed for input.

END

specifies the disposition of the file on completion of the step when it is normally terminated.

= DEASSIGN signifies that the file is not automatically made known to next step (default value).
= PASS signifies that the file should be passed to the next step.
= UNLOAD signifies that the file is deassigned and the tape unloaded (value reserved for tape files).

FSN

This gives the file sequence number when the file is on a multifile tape. The file sequence number may be 1 through 255.
The number 255 indicates that the file sequence number is not known. In this case, if the file is to be input, the whole tape will be searched for the specified external file name. If the file is to be output, it will be written immediately after the end of the last file held on the tape.
If a file sequence number less than 255 is specified, the following constraints apply:
For input: the file sequence number must refer to a file possessing the specified external file name.
For output: the file sequence number of the last file held on the multifile tape must be one less than the file sequence number specified in the FSN option.

ABEND

This has the same meaning as END but applies to abnormal termination.

SUBFILE

member-name is the name of the member of a

library when the file description applies to a library member.

LABEL — specifies the volume label. It applies to all the volumes given in the MEDIA parameter (see below).

= NATIVE. Label is native Level 64 format (default value).
= NONE. Unlabelled tape file.
= H200. HFAS disk file.
= H200E80. File label is extended, size is 80 characters.
= H200C80. File label is 80 characters.
= H200C120. File label is 120 characters.
= H200NONE. Unlabelled tape file.

(H200, H200E80, H200C80, H200C120, H200NONE: HFAS tape files.)

MOUNT — gives the number of volumes to be mounted. The first nn volumes specified in MEDIA are to be mounted. The others are mounted as required. MOUNT is only meaningful for multivolume files. If not specified, then all volumes specified in MEDIA are mounted at the beginning of the step execution. (See also REPLIN below.)

FIRSTVOL — This specifies the volume sequence number of the first volume of the file to be read. The default value is one. The volume sequence number refers to the order of volumes in the volume list associated with the keyword MEDIA.

LASTVOL — This specifies the volume sequence number of the file to be read. The default value is the sequence number of the last volume existing in the file.

RESIDENT — Specifies that the file resides on one (or more) disks volumes that are resident.

DEVCLASS — Device class name and attributes. For example, MS/M350, MT/T9/D1600, or DH/H279.

MEDIA — Lists media identifiers. If MEDIA is not specified then the media are all assigned the WORK attribute.
= (WORK) signifies a Work volume. Should be used for output files only.
= (media-name) gives the volume name of the volume-external-identifier.

FILEFORM — Specifies the access system, UFAS, BFAS or HFAS.

	The following file organizations are suported by these access systems: BFAS Sequential (disk or tape) Indexed sequential Direct HFAS Sequential (disk or tape) Indexed sequential Random UFAS Sequential Indexed Relative DOS 360 files (input only) are processed as BFAS files. The only difference is that some parameters must be specified for a DOS 360 file (see File Specification Rules for BFAS, HFAS and UFAS below). The DOS 360 organizations supported are Indexed Sequential (ISAM), Sequential (SAM - disk or tape) and Direct (DAM). Level 62 files (input only) are also processed as BFAS files. The Level 62 files (input only) are also processed as BFAS files. The Level 62 organizations supported are Sequential and Direct.
WRCHECK	This is only used for BFAS disk output files. When this option is selected, each block written to the output file is re-read to check that it has been correctly written. If the WRCHECK option is not selected, no read after write will be carried out.
ADDRFORM	This keyword is used to specify the format of the input record disk addresses to be included in the output record. ADDRFORM is significant only when one of the ADDATA, ADDROUT or KEYADDR output record formats is selected. = TTRDD signifies that the address is to be stored in the form: Bytes 1 and 2: relative track number in the file (TT). Byte 3: block number on the track (R). Bytes 4 and 5: displacement within the block (DD). TTRDD is the default value. = LRRR signifies that the address is to be stored in the form: Bytes 1, 2 and 3: Logical record number, that is, the serial number of the record within the file. Bytes 4 and 5: undefined value. LRRR is applicable to any file organization with fixed length records.
ERROPT (parameter of the file description)	This specifies the action to be taken in case of an I/O error on the file. = RETCODE signifies that the SORT or MERGE should display the return code via SYSOUT and then terminate. This is the default value. = SKIP signifies that the block at which the I/O

error occurred should be skipped and processing should continue with the next block.
= ABORT signifies that the SORT or MERGE should terminate immediately.
= IGNORE signifies that the remainder of the block which has not been read should be padded with blanks and processing should continue with the incorrectly read block. This option should be used with care since invalid decimal data may be introduced, resulting in abortion of the SORT or MERGE.

RECSIZE — gives the size of a logical record in a UFAS or BFAS file.
For fixed length records, nnnnn is the size, in bytes.
For variable length records, nnnnn is the maximum size, in bytes, not including the record descriptor word (RDW).

ITEMSIZE — Same as RECSIZE, but for HFAS files.

BLKSIZE — gives the size of a block in UFAS or BFAS files. ★
Fixed - nnnnn is the size, in bytes, of the block not including block serial number.
Variable - nnnnn is the maximum block size, in bytes, not including the block descriptor word.

RECFORM — Specifies the logical record format for UFAS and BFAS files only.
= FB - fixed length, blocked (default value).

= F - fixed length, unblocked.

= VB - Variable length, blocked.

= V - Variable length, unblocked.

= U - Undefined format.

NBSN — For UFAS and BFAS tape file only, indicates that the file has no block serial numbering. Default is the contrary.

H200FORM — Indicates the format for HFAS magnetic tape files.

= TA - Type A
= TB - Type B
= TC - Type C
= TD - Type D
= TE - Type E
= FORMAT1 - Form 1
= FORMAT2 - Form 2
= FORMAT3 - Form 3
= FORMAT4 - Form 4

BAN	Indicates that the HFAS file uses banner characters 41 in octal.
BANCHAR	Indicates that the octal value given as nn is used as banner characters in the HFAS file. If neither BAN nor BANCHAR is specified, no bannering is assumed.
PADCHAR	allows the default value for the padding character to be overridden by the octal value given in nn. Restricted to HFAS files.
SIZE (in output file description)	gives the number of cylinders to be allocated to the output file described. Note that space allocation takes place when the files are opened, i.e., at the beginning of the last pass. If the amount of requested space is not found on the specified disks, the sort process will be aborted at that time. (Should not be specified for HFAS files).
EXPDATE	Specifies the expiration date of the file. expiration-date may take 3 forms: = nnn where nnn is the number of days the file must be retained, = yy/ddd where yy is the year within the century, ddd is the day within the year, = yy/mm/dd where yy is the year within the century, mm the month of the year and dd the day of the month.
DATABUF	Specifies the number of buffers to be used by the sort or merge when processing the file. The default value is 2, that is, double buffering, but single buffering may be required in order to minimize memory usage.
BPB	This specifies the number of blocks per buffer. That is, the number of blocks to be read at each physical access of the input file, or the number of blocks to be written at each physical access of the output file. The default value is 1 (the best value for normal SORT or MERGE applications). However, higher values may be used to improve performance when the working set is greater than 256K or when the blocks are very small.
REPLIN	Used to indicate to the sort or merge that the volume(s) supporting the output file need not be mounted before the sort starts, and may use the

	device supporting the input file volume(s). This parameter may be used to minimize the device requirements of a sort. In case of multivolume input and output files, the MOUNT option must be set equal to 1 for both input and output file. If the input and output files are on tape, and a tape sort is executed, the specification of REPLIN in OUTFILE has no effect if either REPLIN, REPLOUT or USEOUT is used as a work tape option (see below).
INFILE	Specifying OUTFILE = (INFILE) allows the output to overwrite the input file. It must not be specified when there is file concatenation on input.
INFILEn ($SORT parameter, not $MERGE).	The keywords INFILE 1 through INFILE 8 may be used to specify files that must be concatenated with INFILE. File concatenation requires that all input files have the same file format, file organization, record format, record size and block size. The INFILEn parameter must appear immediately after the INFILE parameter and the values used for n must start at 1 and must be in sequence. For example: INFILE = ( ), INFILE 1 = ( ), INFILE 2 = ( ) is correct. INFILE = ( ), INFILE 2 = ( ), INFILE 3 = ( ) is incorrect.
COMFILE	Specifies the location of the sort/merge DSL statements. Data format may be either SSF or SARF. = *input-enclosure-name signifies that the DSL statements are in the named input enclosure. = (file) Defines the library member containing the DSL statements.
WKTAPE[S]	Declares that the sorting media are magnetic tape volumes.
NBDV	Allows the number of devices required to support the sort work tapes to be specified in n. Minimum value for n is 3, maximum is 6.
DEVCLASS	Device class name and attributes.
REPLIN	A tape device used to support the reading of the input file is to be later used for one of the sort work tapes.
REPLOUT	A tape device used for one of the sort work tapes is to be later used to support the writing

of the output file.

USEOUT — The tape volume to be subsequently used for the output file is to be initially used as a sort work tape.

WKDISK[S] — Declares that the sorting media are disks. The work file can be either permanent, in which case it must have been preallocated, or temporary, in which case the space specified will be allocated when the file is opened. The file will be formatted by the sort as needed.

SIZE (in work file description) — Indicates that the file is temporary, and specifies that nnn cylinders should be allocated on the volume specified in the MEDIA list.

WORK — Indicates, for temporary files, that work media should be used.

REPORT — Specifies the type of report the user wishes to have supplied by the sort or merge. Unless PRTFILE (below) is specified, the report is written on SYSOUT (See Appendix B).

- = AUDIT — Causes diagnostic and process error information to be printed, and a statistical summary of record processing. (Default value).
- = PARAM — Causes a list of parameters that are specific to the sort/merge to be printed, in addition to diagnostic and process error information.
- = ALL — Combines AUDIT and PARAM
- = NONE — Limits the report to process information.

PRTFILE — Specifies a permanent SYSOUT file. The file must be a native one with LABEL = NATIVE or LABEL = NONE. For a permanent output file on tape, the following default values are supplied RECFORM = VB, RECSIZE = 264, BLKSIZE = 1072.

SIZE ($SORT parameter, not file description) — Specifies the total memory requirements in units of K bytes allocated to the sort for the execution. This includes code, data and buffers. The default value is 53, but any value may be specified up to a maximum of 512. (For processing large records, refer to Appendix H.)

INVREC — Specifies the action to be taken by the sort if an invalid record is encountered in the input file, that is, a record whose length is smaller

than the minimum needed to apply the INCLUDE/OMIT clauses, or the KEY and/or SUM and/or ARRANGE clauses of the DSL commands.

= ABORT (Default value) Sort process should terminate when an invalid record is found.

= CONTINUE Sort process should continue.

ERROPT (INVREC parameter, not file description) — Indicates what is to be done with the invalid record.

= PRINTID The invalid record's number relative to the start of the sort is printed in the report. This is the default value, and is the only value valid for HFAS files.

= PRINT As for PRINTID, and a display on the report of the entire invalid record in hexadecimal and alphanumeric formats.

= LOG The invalid record is written onto the file specified in LOGFILE.

= IGNORE No message is printed.

LOGFILE — Contains a description of a native output file onto which the invalid records are to be logged. As only ERROPT = PRINTID is allowed for HFAS files, this parameter must not be specified for HFAS file sort.

START — Gives the record number with which sorting is to begin. Values from 1 to 2147483647 are acceptable. The default value is 1.

HALT — Gives the record number with which sorting is to end. The same values are acceptable as for START. The default is end-of-file.

REPEAT — The sort checkpoint capability is requested. The default is no checkpoint.

DUMP	Defines the printed output in case of abnormal termination. = NO (default) No dump is output. = DATA Data segments are output.
BUFFER	Normally, 30K bytes of memory are reserved for the input and output buffers of a merge. Depending on the block sizes of the files, and the number of buffers per file, this may be too low or too high. The user can override the default value by specifying this parameter, with nnn equal to the number of units of K bytes required. The total merge memory requirement will be (12 + nnn) K bytes. The buffer requirement is calculated by adding together the products of the blocksize and the buffering factor of all the input files and the output file.

File Specification Rules for BFAS, HFAS and UFAS

These rules are easy to understand if the reader bears in mind the normal overriding rules mechanism. A file to be processed by the sort or the merge will be assigned and then opened. If the file already exists and has native labels, then there is no need to specify any additional parameters since the labels contain the information required for consistent file processing.

On the otherhand, if the file is to be created or if the file to be processed in input has not self sufficient labels (e.g., unlabelled tape file, HFAS tape file, DOS 360-SAM files, DOS 360-DAM files) then some additional parameters must be specified in the file specification.

The circumstances under which these parameters must be specified are described below. Parameters are either optional (o) or mandatory (m). Optional means that if the parameter value is not explicitly given in the JCL, the default value for this parameter is used. Mandatory means that a value must be given in JCL for the corresponding parameter since no default value is provided.

Native and Compatible Organizations: BFAS, UFAS, DOS 360, Level 62

DOS 360 and Level 62 supported organizations are processed as BFAS. However, DOS 360 and Level 62 labels are not always self sufficient and some parameters must sometimes be specified. DOS 360 and Level 62 organizations are supported in input only. The parameters required for disk files are summarised in figure 2-4.

FILE TYPE	FILE ORGANIZATION	EXTRA PARAMETERS REQUIRED	OPTIONAL OR MANDATORY
PERMANENT or TEMPORARY DISK INPUT	BFAS UFAS DOS 360-ISAM LEVEL 62	none	
	DOS 360-SAM DOS 360-DAM	FILEFORM RECFORM RECSIZE BLKSIZE	O O M M
PERMANENT DISK OUTPUT	BFAS UFAS	none	
TEMPORARY DISK OUTPUT	BFAS	FILEFORM RECFORM RECSIZE BLKSIZE	O O M M
	UFAS (*)	FILEFORM RECFORM RECSIZE	M O M

(*) A default value of 2048 bytes will be taken for CISIZE

Figure 2-4. Parameters Required for Disk Files

The parameters required for tape files are summarized in figure 2-5.

FILE TYPE	FILE ORGANIZATION	EXTRA PARAMETERS REQUIRED	OPTIONAL OR MANDATORY
PERMANENT OR TEMPORARY LABELLED TAPE INPUT FILE	BFAS UFAS	none	
	DOS 360-SAM (Permanent file only)	FILEFORM RECFORM RECSIZE BLKSIZE	O O M M
	Level 62 (Permanent file only)	FILEFORM NBSN RECFORM RECSIZE BLKSIZE	O O O M M
UNLABELLED TAPE INPUT FILE	BFAS(*) UFAS	FILEFORM RECFORM RECSIZE BLKSIZE	O O M M
(PERMANENT OR TEMPORARY) (LABELLED OR UNLABELLED) TAPE OUTPUT FILE	BFAS UFAS	FILEFORM RECFORM NBSN RECSIZE BLKSIZE	O O O(*) M M

(*) Unlabelled tape files are always processed by GCOS Level 64 assuming they do not have a block serial number.

Figure 2-5. Parameters Required for Tape Files

Coexistence Organization: HFAS

At JCL translation time, the detection of HFAS files is based on the LABEL value. That means that the FILEFORM = HFAS parameter becomes optional. In other words if an HFAS file is to be processed, then the absence of FILEFORM or the specification of FILEFORM = HFAS produce the same result. The parameters required for disk and tape files are summarized in figure 2-6.

FILE TYPE	EXTRA PARAMETERS REQUIRED	MANDATORY OR OPTIONAL
DISK INPUT or DISK OUTPUT FILE	none	
TAPE INPUT OR TAPE OUTPUT FILE	FILEFORM BAN or BANCHAR PADCHAR ITEMSIZE BLKSIZE H200FORM	O O O M M M

Figure 2-6. Parameters Required for Disk and Tape HFAS Files

$SORT and $MERGE Examples

An input file named PAYLIST on tape volume TV24 with a record length of 200 bytes and a blocksize of 4000 bytes is to be sorted. The output file will also be on tape file name SPAYLIST, volume TV81. A smaller blocksize, 2000 bytes, will be employed on output. The installation has three tape drives, all with the same attributes, and the DSL commands for the sort are held as a source library member, SCPAY, in a RESIDENT source library names SQLLPY. The memory allocated to the sort is 50K bytes.

```
$SORT   INFILE=(PAYLIST,DEVCLASS=MT,MEDIA=TV24),
        OUTFILE=(SPAYLIST, EXPDATE=300, DEVCLASS=MT,
                 MEDIA=TV81, RECSIZE=200,BLKSIZE=2000),
                 SIZE=50,
        WKTAPE=(NBDV=3, DEVCLASS=MT, REPLIN,REPLOUT),
        COMFILE=(SQLLPY, SUBFILE=SCPAY) ;
```

Since there are only three drives available the REPLIN and REPLOUT parameters are required. When the sort job step starts there will be two WORK tapes and the input tape, TV24, mounted. When the input phase of the sort process is complete the volume TV24 will be replaced by a WORK volume and sorting will continue. For the output phase of the sort, one of the three WORK volumes will be replaced by the volume TV81 and the output file SPAYLIST will be written.

If the above sort was performed using the provided default disk work space then the statement would be:

```
$SORT     INFILE= (PAYLIST,DEVCLASS=MT,MEDIA=TV24),
          OUTFILE= (SPAYLIST,EXPDATE=300, DEVCLASS=MT,
                    MEDIA=TV81, RECSIZE=200,BLKSIZE=2000),
          SIZE=50,COMFILE = (SQLLPY,SUBFILE=SCPAY);
```

If the temporary disk file space is used then the statement would have the following form:

```
$SORT     INFILE= (PAYLIST,DEVCLASS=MT,MEDIA=TV24),
          OUTFILE= (SPAYLIST,EXPDATE=300, DEVCLASS=MT,
                    MEDIA=TV81, RECSIZE=200,BLKSIZE=2000),
          WKDISK= (SIZE=15,DEVCLASS=MS/M400, MEDIA=BD14),
          COMFILE= (SQLLPY,SUBFILE=SCPAY);
```

In this case the sort work space is a temporary field of 15 cylinders on an MSU0400 disk volume named BD14. The temporary file is named H_SRTWKD.

If the file SPAYLIST is too large to be accomodated in the disk space available the user may wish to split the sort process into two parts and then subsequently merge the results to form the complete sorted file. If the file contains 10000 records then a two-part sort and merge would be of the form:

```
$JOB     LSPAY,USER= LX518,PROJECT=INTER;
$SORT    INFILE=(PAYLIST,DEVCLASS=MT,MEDIA=TV24),
         OUTFILE=(SPAYLIXA,DEVCLASS=MT,
                  MEDIA=TVVV,RECSIZE=200,BLKSIZE=2000),
                  SIZE=50,COMFILE=(SQLLPY,SUBFILE=SCPAY),
                  HALT=5000;
$SORT    INFILE=(PAYLIST,DEVCLASS=MT,MEDIA=TV24),
         OUTFILE=(SPAYLIXB,DEVCLASS=MT,
                  MEDIA=TVVB,RECSIZE=200,BLKSIZE=2000),
                  SIZE=50,COMFILE=(SQLLPY,SUBFILE=SCPAY),
                  START=5001;
$MERGE   INFILE1=(SPAYLIXA,DEVCLASS=MT,MEDIA=TVVV),
         INFILE2=(SPAYLIXB,DEVCLASS=MT,MEDIA=TVVB),
         OUTFILE=(SPAYLIST,DEVCLASS=MT,MEDIA=TV81,
                  RECSIZE=200,BLKSIZE=2000),
         COMFILE=*MERCAR;
$INPUT MERCAR ;
       .
   DSL statements
       .
$ENDINPUT;
$ENDJOB;
```

Note how the DSL statements for the merge are entered as an input enclosure. The two-part operation is performed using as intermediates the files SPAYLIXA and SPAYLIXB, on tape volumes TVVV and TVVB.

File Specification Examples

INFILE

Sorting a UFAS or BFAS file

. on resident disk : INFILE = (FILE1)

. on a user disk pack : INFILE = (FILE2,DEVCLASS = MS/M400,
MEDIA=DISK1)

. on tape : INFILE = (FILE3,DEVCLASS = MT/T9/D1600,MEDIA = TAPE1)

Sorting a HFAS file

. on Disk : INFILE = (FILE4,DEVCLASS = MS/M350,
MEDIA = DISK2,LABEL = H200)

. on TAPE : INFILE = (FILE5,DEVCLASS = MT/T9/D800,MEDIA = TAPE2,
LABEL = H200C80,ITEMSIZE = 90,
BLKSIZE = 900,BAN,H200FORM = TB)

. Sorting an Unlabelled tape file
INFILE = (FILE6,DEVCLASS = MT/T7/D800,
MEDIA = TAPE1,LABEL = NONE,
RECSIZE = 80,BLKSIZE = 800)

OUTFILE

Sort output is UFAS or BFAS file

. Preallocated file on disk :
OUTFILE = (FILE7,DEVCLASS = MS/M400,MEDIA=DISK3)

. Temporary BFAS file on disk

OUTFILE = (FILE8,DEVCLASS = MS/M300,MEDIA = DISK4,
FILESTAT = TEMPRY,
SIZE = 10,RECSIZE = 80, BLKSIZE = 400)

. Temporary UFAS file on disk

OUTFILE = (FILE9,DEVCLASS = MS/M400,MEDIA = DISK5,
FILESTAT = TEMPRY,
SIZE = 4,FILEFORM = UFAS,RECSIZE = 127,RECFORM = V)
(Default CISIZE is 2048 bytes (2K)).

. Permanent BFAS file on disk

as for temporary BFAS file with the following differences:

FILESTAT = UNCAT, CAT (or omitted)
EXPDATE can be specified.

- Permanent UFAS file on disk

  as for temporary UFAS file with the following differences:

  FILESTAT = UNCAT, CAT (or omitted)
  EXPDATE can be specified.

- BFAS tape file

```
OUTFILE = (FILE0,DEVCLASS = MT/T9/D800,MEDIA = TAPE4,
           RECSIZE = 90,BLKSIZE = 270,EXPDATE = 78/12/31,
           NOSN)
```

- Unlabelled tape file

  Similar to unlabelled tape file in input

- HFAS output file

  Similar to HFAS input files.

SECTION III

DATA SERVICES LANGUAGE FOR $SORT

The $SORT Data Services Language (DSL) statements, presented via the JCL COMFILE parameter group, describe the record-level functions to be performed during process. These functions are based on the nature of the input data content and format. Since the KEYS statement is always mandatory, a COMFILE must be provided with every sort.

Sort DSL statements are interpreted according to punctuation or keyword analysis, not position. The command file should be thought of as a continuous stream of characters. If the statements are punched on cards, punching may be continued from one card to the next card, provided that punching is continued starting from the first column on the continuation card. If a card is to begin with a new parameter, the first column must contain a space to set it off from the preceding card.

COMMAND FILE ORGANIZATION

The DSL statements are organized into three paragraphs called FUNCTION, RECORD, and END. Of these, the FUNCTION paragraph is optional but, if supplied, must precede the other two. FUNCTION paragraph parameters are used to select special processing options. The default conditions for these options are underlined in the text; if a single parameters is given, the default option should be understood to be the reverse or absence of the associated function.

The RECORD paragraph follows FUNCTION (if supplied) and contains a set of statements that provide the sort utility with information regarding the content and format of individual fields within the user's records. A RECORD paragraph is mandatory since it includes the KEYS specification.

The final paragraph in a sort parameter file consists simply of the parameter END which terminates the file.

Each paragraph heading can be followed by a colon. Within a paragraph, statements or parameters may appear in any order, and must be separated from one another by either blanks or commas.

Elements of a statement specification may be parenthesized at the user's option to enhance readability or aid organization. Comments may be inserted to improve readability. A comment has the form /* comment */. Neither of the character combinations /* or */ is allowed in the comment itself. Comments may appear anywhere a blank or a comma is allowed, and must be preceded and followed by one.

FUNCTION PARAGRAPH

The complete FUNCTION paragraph is shown below.

```
FUNCTION: [DESCEND]
          [COLLATE = { EBCDIC          }]
                     { G100            }
                     { H200            }
                     { ASCII           }
                     { 'hexadecimal-string' }
          [OUTPUT = { DATA     }]
                    { ADDATA   }
                    { KEYADDR  }
                    { ADDROUT  }
          [DUPREC = { NONE }]
                    { FIFO }
          [DELETE]
          [SORTSIZE = number-of-records]
```

The parameters in the FUNCTION paragraph are described in the following paragraphs.

DESCEND

DESCENDING SEQUENCE

Unless otherwise specified, the sort process will output records in ascending order of the key values; that is, a record whose key value is 1 will appear before a record whose key value is 2, and alphabetic characters will appear in the order appropriate to their collating sequence. When DESCEND is specified, this order is reversed and records are sorted into a descending key value order.

COLLATING SEQUENCE

The COLLATE keyword allows the user to explicitly state which collating sequence is to be employed by the sort utility. There are four sequences available:

EBCDIC : Extended Binary Coded Decimal Interchange Code (the standard Series 60 Level 64 code set)

G100 : Series 100 Code Set

H200 : Series 200/0 Binary Collating Sequence

ASCII : American National Standard Code for Information Interchange

'hexadecimal-string' : privately defined collating sequence

Figure 3-1 shows the circumstances under which each sequence may apply.

FILE TYPE / SEQUENCE	HFAS input or output UFAS BFAS	HFAS input and output Series 200/2000 (IBCD or HBCD)
EBCDIC	X	X*
G100	X *	
H200	X *	X
ASCII	X *	
'hexadecimal-string'	X *	X *

X : valid combination
* : key fields must not overlap

Figure 3-1. Collating Sequence Selection.

If COLLATE is omitted then the default value chosen depends on the type of file being sorted. If the input file and the output file are both HFAS (i.e., Series 200/2000) then the default is H200. Otherwise the default is EBCDIC.

Option	Effect
EBCDIC	This is the standard internal code of Series 60 Level 64. The sequence can also be applied to a Series 200/0 file, in which case the resulting order is that in Table E-4 of Appendix E.
G100	This value should only be specified when the input file has been translated from a Series 100 environment to the GCOS Level 64 environment. This translation can be performed using the Series 100 File Translator. G100 causes the records to be sorted into the same order as would have occurred if the sort was performed on a Series 100 system (Table E-2, Appendix E). The use of G100 for translated files is not mandatory. If the following conditions apply: . keys do not contain special characters and . keys do not mix numeric and non-numeric characters in the same positions then G100 is not necessary. The collating sequence EBCDIC is valid. The performance of sort is better using EBCDIC than any other collating sequence.
H200	This value is the default when the sort is applied to Series 200/0 file. It will cause the records to be sorted into the same order as would have occurred if the sort was performed on a Series 200/0 system (Table E-3, Appendix E). Note also that H200 can be applied to native (EBCDIC) files. This allows the user to maintain the original Series 200/0 collating sequence for a file written in the standard native format and code set. The resulting order is that shown in Table E-4, Appendix E.
ASCII	This value specifies that the ASCII (American National Standard Code for Information Interchange) collating sequence be applied. This sequence is defined in Table E-5 of appendix E. This table shows the low-order to high-order sequence as it is applied to the complete range of EBCDIC hexadecimal values available for a character in Series 60 Level 64. The ASCII collating sequence may only be applied to a GCOS (EBCDIC) standard file.

'hexadecimal-string'

To define a collating sequence other than the above, the hexadecimal representation of all the characters in the character set are entered in order, starting with the highest (that which should always win a comparison) and ending with the lowest (that which always loses). The number of elements in the character set, and hence the length of the hexadecimal string, is a function of the type of file. HFAS file : the data is assumed to be in Series 200/0 code. The collating sequence must include 64 characters, that is 128 hexadecimal digits. For example, the DSL command

```
COLLATE = '1112131415161718192122232425262728292
           32333435363738390001020304050607080 9
           0A0B0C0D0E0F101A1B1C1D1E1F202A2B2C2D2E2F
           30313A3B3C3D3E33F'
```

has the effect of collating alphabetic characters, ahead of numeric digits, ahead of other characters (A,B,C...Z,0,1,2...9,', = ,:,.....,C). See Appendix E.

Other files : the data is assumed to be in EBCDIC code. The collating sequence must include 256 characters, that is 512 hexadecimal digits. For example, if it is desired to sort with numeric higher than alphabetics, (0123456789ABC....), the DSL command would begin

```
COLLATE = 'F0F1F2F3....F9C1C2C3....'
```

OUTPUT

OUTPUT FILE CONTENT

The OUTPUT keyword is used to specify the contents of the records output by the Sort program. Depending on the value given to the parameter, the output record may contain the entire input record content, or combinations of disk address, data and key fields. The four values of the parameter are given below.

Option	Effect
DATA	The output record will consist only of data that appears in the corresponding input record. Either the entire record may appear or only those fields extrac- ted from the record during the sort process (via the ARRANGE parameter). This is the only valid OUTPUT choice for HFAS.
ADDATA	The output record will consist of original input data as above but will also contain a prefix field giving the original record's disk address.
ADDROUT	The output record will contain no data at all but will contain the disk address of the original input record.
KEYADDR	The output record will begin with a field giving the disk address of the original record. This will be followed by the contents of the major and minor key fields of the corresponding input record.

The format of the disk address is : TTRDD (TT - relative track number; R - block number relative to the beginning of the track; DD - record displacement within block), or LRRR (logical record number).

An example of the output record formats for these options is given in Figure 3-2.

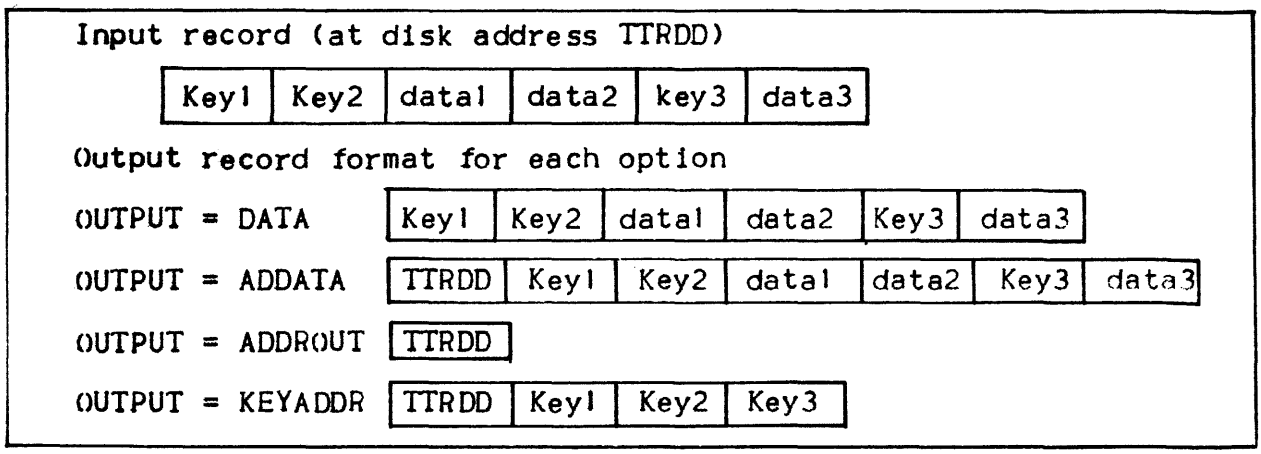


Figure 3-2. Sample Output Formats

The DATA and ADDATA options are used when the user wants the output file to contain all or most of the original data. ADDROUT and KEYADDR are suitable for applications requiring an index of sorted records, or when there is a limitation on space available for the output file. ADDATA, ADDROUT and KEYADDR should not be used when concatenated files are specified for input.

When the address format is LRRR, bytes 1 to 3 of the output record contain the logical record number. The contents of bytes 4 and 5 are undefined. The options ADDATA, ADDROUT and KEYADDR are only valid when the input file is disk resident. Note that for ADDROUT and KEYADDR output to tape, the record length specified must be at least 18 bytes.

DUPREC

DUPLICATE KEY OCCURRENCES

The DUPREC keyword specifies the action to be taken when two or more records have identical values for each of the key fields specified. The meaning of the two options is given below:

Option	Effect
NONE	All of the records with identical key fields are output in a random sequence.
FIFO	First-In-First-Out. The duplicate records are output in the order in which they occur in the input file.

DELETE

DUPLICATE RECORD DELETION

The DELETE option specifies that all but one record of a set with identical key field values should be deleted. The choice of which of these records to retain for the output file is dependent on the setting of the DUPREC option. If the output file is in key sequence and DUPREC is NONE, one of the duplicate records will be selected randomly. If FIFO has been selected, the first record in will be the one retained.

If in the RECORD paragraph, summation is specified then DELETE must not be specified in the FUNCTION paragraph since deletion is automatic with summation.

SORTSIZE

TEMPORARY STORAGE REQUIREMENTS - PERFORMANCE IMPROVEMENT

When insufficient secondary working storage has been assigned, the sort process cannot be completed. In order to ensure that there is sufficient storage space available, the user can give in the SORTSIZE keyword an estimate of the number of input records.

SORTSIZE should be set equal to a decimal number which is the user's estimate of the number of input records. The Sort program uses this estimate to determine the amount of working storage that will be required, and commences the sort process only if the required amount is available. Performance of the sort is improved if this estimate is close to the exact number. Overestimation is better than underestimation for performance optimization.

EXAMPLE OF FUNCTION PARAGRAPH

Example 1:

```
FUNCTION:  DUPREC = FIFO    SORTSIZE=10250
```

The sort output is to be ascending order. Records with identical key field values will appear on the output file in the same order as on the input file. The output file record will contain the same fields as the input record.

Example 2:

```
FUNCTION:  OUTPUT=KEYADDR,DELETE,SORTSIZE=5320
```

The FUNCTION paragraph above specifies that the output file, to be ordered in ascending key sequence, should contain only the key and the original disk address of each record. DUPREC=NONE has been chosen by default. Thus the DELETE parameter ensures that if two or more records are found to have the same key, only one of them, selected at random, will be selected and its address written in the output file.

RECORD PARAGRAPH

The statements in the RECORD paragraph specify fields within the user's input records which must be examined by the sort process during execution. They are shown in the format model below.

```
RECORD:  KEY[S]= key-element  [key-element] ...

    [ { INCLUDE = condition-elemt [AND condition-elemt]...
      {   [INCLUDE = condition-elemt [AND condition-elemt]...]...
      { OMIT = condition-elemt [AND condition-elemt]...
      {   [OMIT = condition-elemt [AND condition-elemt]...]...  ]

    [SUM = sum-element [sum-element] ...]

    [ARRANGE = arrange-element [arrange-element]...]
```

KEY specifies one or more key fields whose values determine the sort order.
INCLUDE and OMIT are mutually exclusive record selection statements specifying whether or not an input record is to be included in or omitted from the sort process on the basis of field contents. Both statements must not appear in the same paragraph.
SUM specifies a field whose value is to be totaled for a series of records with identical key values. The sum of these values will be sorted in the record that will be retained for output. However, overflow during summation is not detected.
ARRANGE permits the user to select certain fields from the input record for omission or placement in other positions.

KEY [S]

KEY FIELD SPECIFICATION

Every sort process requires the specification of at least one key field. Up to 64 key fields may be specified by means of the KEYS statement. The maximum length of a key field is 255 bytes. The format of the statement is as follows:

KEY[S] = pos length [dtype][RV]...

In the format, pos is the byte position of the first byte of the key field relative to the first data byte of the record, and length is the length in bytes of the key field. The numbering of data bytes in a record starts with byte 1. For example, the numbers 1,3 refer to the field consisting of the first three data bytes of a record. They can be separated by a comma or a space. The dtype entry is one of the data type keywords given in Table 3-1. When not specified, the default data type is CHAR. CHAR is the only type which may be applied to HFAS files. The RV parameter specifies that the contents of this particular key field should be ordered in the opposite sequence from that prevailing for the sort process as a whole.

Table 3-1. Data-type Keywords

Keyword	Meaning
UBIN	Unsigned binary data (2- or 4-byte)
SBIN	Signed fixed binary data (2- or 4-byte)
CHAR	Character string (maximum 256 characters)
UDEC	Unpacked decimal data (maximum 31 decimal digits and sign)
PDEC	Packed decimal data (maximum 31 decimal digits and sign packed into 16 bytes). The length is given in bytes (maximum 16).

EXAMPLE OF THE KEYS STATEMENT

The example below shows the use of the KEYS statement to declare two fields in a user record as key fields for sorting. The states of the file before and after sorting are also shown.

Example:

RECORD: KEYS = 4,1,5,1

	Input File Record Fields A	B	C	D	E		Output File Record Fields A	B	C	D	E	Key Field Values
Rec1	3	1	X	X	9	Rec6	8	2	X	X	4	X 4
Rec2	7	2	X	Y	6	Rec3	6	1	X	X	5	X 5
Rec3	6	1	X	X	5	Rec7	8	3	X	X	6	X 6
Rec4	4	1	X	Z	1	Rec1	3	1	X	X	9	X 9
Rec5	7	1	X	Y	3	Rec5	7	1	X	Y	3	Y 3
Rec6	8	2	X	X	4	Rec2	7	2	X	Y	6	Y 6
Rec7	8	3	X	X	6	Rec4	4	1	X	Z	1	Z 1

The file consists of seven records, labeled Rec1 to Rec7 for the purpose of the example. Each record is five bytes long and contains five 1-byte fields. The KEYS statement declares fields 4 and 5, in that order as the sorting key fields. The output file is ordered as if the Sort program has sorted the records according to the content of field 4, and then for a given value of field 4 further sorted the records on the basis of the content of field 5, wherever necessary.

INCLUDE/OMIT

RECORD SELECTION CRITERIA

The INCLUDE and OMIT statements establish record selection criteria for the sort process by testing fields in the input record. To test a value the user writes up to 16 condition elements, a combination of operators and operands, which yield a value of TRUE or FALSE. The details of constructing condition elements are described below. If omitted, variable length records need not contain all key fields, but if included, must.

STATEMENT WITH ONE CONDITION ELEMENT-SAMPLE 1

A sample INCLUDE parameter shown below contains one condition element.

```
INCLUDE=1, 1 EQ 2,1 /* FIRST BYTE = SECOND BYTE */
```

Here two fields are being compared to see if they are equal. Tne form for describing a field consists of a position digit indicating the byte of the record that is the beginning byte of the field followed by a length digit indicating the field length. In the example, the fields are one byte long and are bytes 1 and 2 of the record. The operator EQ indicates that the test is for equality. If bytes 1 and 2 are equal, the condition element yields the value TRUE and the record will be included in the sort. If the same condition element were used after an OMIT, yielding the same value, the record would be omitted from the sort.

STATEMENT WITH TWO CONDITION ELEMENTS-SAMPLE 2

A single INCLUDE or OMIT statement may contain more than one condition element, specifying several tests to be carried out. The parameter AND is used as a condition element separator, as shown below:

```
OMIT = 1,1 EQ 2,1 AND 3,3 EQ 10,3
```

When multiple condition elements separated by AND are present in a statement, records will be selected (for omission or inclusion as appropriate) only if all the condition elements yield a TRUE value. Thus in an OMIT statement, all condition elements being true would cause the record to be omitted.

MULTIPLE INCLUDE OR OMIT STATEMENTS-SAMPLE 3

Alternatives for record selection can be established by using a sequence of INCLUDE statements or a sequence of OMIT statements. When a sequence of INCLUDE or OMIT statements occurs in a RECORD paragraph, the statements are processed in the order in which they appear. Processing of a sequence is terminated as soon as a complete statement is found to be TRUE; the remaining statements in tne sequence are then ignored.

The example below illustrates the analysis of multiple INCLUDE statements in the same paragraph:

```
INCLUDE = 1,5 EQ 6,5 AND 1,1 EQ 10,1

INCLUDE = 1,5 EQ 6,5 AND 1,1 NE 10,1
```

The first INCLUDE statement performs two tests. First the 5-byte fields beginning at bytes 1 and 6, respectively, are compared. Let us assume they are equal. Byte 1 is then compared to byte 10. If

they are equal, the conditions of the statement have all been met and the record is included; the second INCLUDE statement will be ignored, and will have no effect on the selection of the record. If bytes 1 and 10 are not equal, however, the second INCLUDE statement will be processed and will cause the record to be included in the sort process input record stream.
The other tests that can be performed and expressed as condition elements are described in detail in the following subsections.

CONDITION ELEMENTS

A condition element consists of an expression that specifies a test to be made. Two types of expressions may appear in a condition element:

1. A comparison operation which uses a comparison operator to compare the field value against another specified value;
2. A numeric condition which tests the field for a positive, negative or zero value.

Comparison Operations

A condition-element consisting of a comparison operator has one of the following forms:

- Field Comparison:

  pos length operator pos length [dtype]

- Literal Comparison:

  pos length operator [dtype] 'literal'

The field comparison compares one field with another. The literal comparison compares a field of a specific data type with a literal value.
The operator is one of the comparison operators shown in Table 3-2.

Table 3-2. Condition Element Comparison Operations

Operation	True If
a EQ b	a is equal to b
a NE b	a is not equal to b
a LT b	a is less than b
a LE b	a is less than or equal to b
a GT b	a is greater than b
a GE b	a is greater than or equal to b

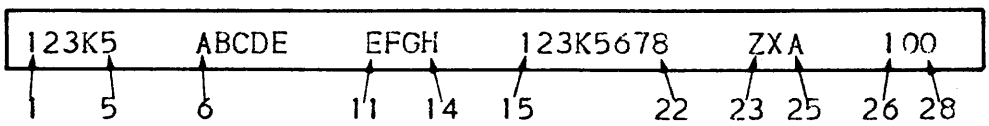
Table 3-1 lists the allowable data types and their respective keywords.

A literal operand is a character string delimited by apostrophes as shown in the preceding format model. If the apostrophe (') is to be included as a character within a string literal, it must be duplicated. For example, the literal 'can''t' specifies the string can't.

If a literal string of digits is to be compared against a field with a numeric data type, then the literal is converted to the appropriate numeric data type using, if necessary, right justification and leading zeros to equalize the length. If the literal is to be compared with a character field, the user must ensure that the length of the literal is equal to the length of the comparison field, by adding blanks as padding characters in the correct positions, if necessary.

Sample condition elements using comparison operators are shown below. The numbers indicate the values in the field and not necessarily their internal representation.

Input Record



Condition Element	Value
4 2 EQ 18 2	TRUE
1,1 GT 28 1 UDEC	TRUE
26 3 GT 1 3 UDEC	FALSE
1 1 NE 2 1 UDEC	TRUE
6 5 G T CHAR 'ABCDD'	TRUE

Numeric Conditions

A condition element consisting of a numeric condition expression has the following general form:

pos length operator dtype

Pos length are the same as for the comparison operations. The operator here is one of the three numeric condition operators shown below:

Operator	Meaning
POS	Positive nonzero value
NEG	negative nonzero value
ZERO	Zero

The allowable data types are:

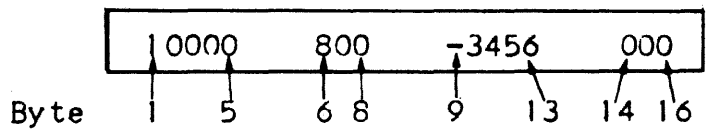
SBIN - Signed fixed binary data

PDEC - Packed decimal data

UDEC - Unpacked decimal data

Sample condition elements using numeric condition operators are shown below. Note that the field values are not the true internal numeric representations. Algebraic representations are used for illustrative purposes.

Input Record

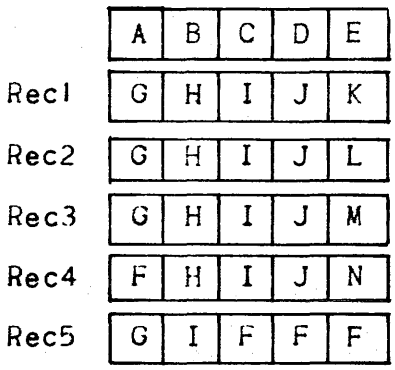


Condition Element	Value
1,5 POS UDEC	TRUE
2 5 POS UDEC	TRUE
2 5 ZERO UDEC	FALSE
9,8 NEG UDEC	TRUE

RECORD PARAGRAPH INCLUDE/OMIT EXAMPLE

Input File

Record Fields



	A	B	C	D	E
Rec1	G	H	I	J	K
Rec2	G	H	I	J	L
Rec3	G	H	I	J	M
Rec4	F	H	I	J	N
Rec5	G	I	F	F	F

1. Include all records whose first byte is a G

   RECORD: INCLUDE=1 1 EQ 'G'

   Records selected: All but rec 4.

2. Omit all records whose first byte is not G.

   RECORD: OMIT=1 1 NE 'G'

   Records selected: All but rec 4.

3. Include all records whose first byte is G and whose second byte is I.

   RECORD: INCLUDE=1 1 EQ 'G' AND 2 1 EQ 'I'.

   Records selected: Rec 5

SUM

Summation Field Specification

The SUM statement is used to cause the values in one or more specified fields to summed for all the records of the same key. SUM is not valid for Series 200/2000 files. The fields to be summed are specified using pos lengths as shown in the model below.

$$\text{SUM} = \text{pos length} \begin{Bmatrix}\text{UDEC}\\\text{PDEC}\\\text{SBIN}\end{Bmatrix}\left[\text{pos length} \begin{Bmatrix}\text{UDEC}\\\text{PDEC}\\\text{SBIN}\end{Bmatrix}\right]\ldots$$

The three data type keywords permit the user to specify the data type of the field. Up to 16 sum fields may be specified for a record: fields must not overlap, and a key field or a part of a key field must not be specified for summation.

The SUM statement applies only to a group of records with identical key values. Its effect is to accumulate a total of data values for the specified sum fields throughout the group of duplicate keyed records and to store this total in the record which will be retained for output. Overflow is not detected. The selection of this record is governed by the setting of the DUPREC option. The DELETE parameter must not be specified in the same run as SUM since deletion of duplicate keyed records is automatic.

Other restrictions on the selection of sum fields are that the field must be one that will appear on the output record (that is, SUM is incompatible with the ADDROUT and KEYADDR options); and that every sum field must appear as an arrange element in an ARRANGE statement if the ARRANGE option is selected (see next subsection). Within this context, the sum field must be specified exactly as it was initially; it may appear within the boundaries of a larger field but sum fields must not overlap. A portion of a sum field must not be specified as an ARRANGE element. If a sum field is duplicated in the output record by the ARRANGE facility, then summation occurs only for the first occurrence of that field in the output record. The remaining occurrences of that field will have an indeterminate value.

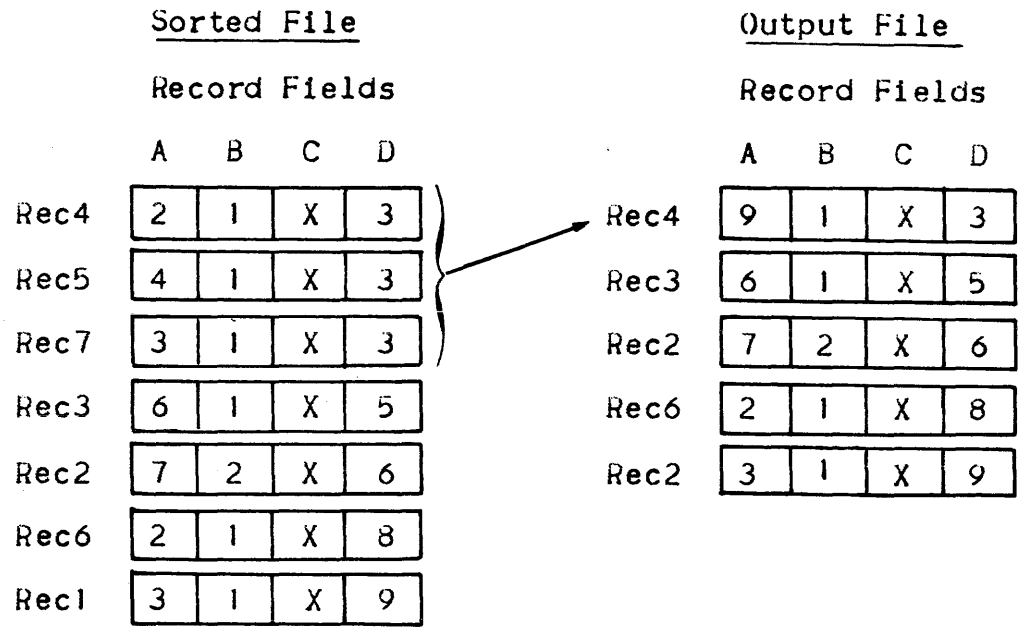
EXAMPLE OF SUM STATEMENT

Three record values are summed into one record for output. The original file and key fields are shown first and then the sorted file, first without summation and deletion and then as it would actually be output.

Input file

	Record Fields				
	A	B	C	D	Key Field Values
Rec1	3	1	X	9	X 9
Rec2	7	2	X	6	X 6
Rec3	6	1	X	5	X 5
Rec4	2	1	X	3	X 3
Rec5	4	1	X	3	X 3
Rec6	2	1	X	8	X 8
Rec7	3	1	X	3	X 3

RECORD: KEYS = 3 1 4 1
SUM = 1 1 UDEC



In the output file, field A of rec4 contains the summed values for the three occurrences of records with the identical key field values X and 3.

NOTE: Field A in other records is unaffected.

ARRANGE

OUTPUT RECORD STRUCTURE SPECIFICATION

The ARRANGE statement is used to define the structure of the output record if it is to be different from that of the input record. The structure is defined by listing all the fields of the record in an arrange list, in the sequence in which they are to appear in the output record. Each field is defined in the list by means of its position and length in the input record, as shown below:

ARRANGE = pos length [pos length]...

The Sort program structures each output record to conform to the structure defined in the ARRANGE parameter, and then writes the record in the output file. Fields not specified in the parameter are dropped. Note that the pos length specification of each field refers to the position of the field in the input record; the position of the field in the output record is determined by the position of the (pos, length) entry in the arrange list.The maximum "length" of an arrange element is 256.

The arrange list may contain a maximum of 128 pos length entries. A field may be repeated in the output record by repeating the corresponding entry at the appropriate positions in the arrange list. The limit of 128 on the number of entries includes all the repetitions, and also key fields, irrespective of whether or not the

key fields are included in the arrange list. Fields that have been defined as sum fields in the SUM parameter must appear in the arrange list, either in their entirety or as subfields of other fields. Clearly, ARRANGE can be used only when OUTPUT is DATA or ADDATA.

EXAMPLE OF AN ARRANGE STATEMENT

A RECORD paragraph containing an ARRANGE statement is shown below. Also shown are the input file and the output file.

```
RECORD:  KEY = 1 1 RV
         INCLUDE =  2 1 EQ 'Y' AND 1 1 LT '7'
         ARRANGE = 3,1 2,1 1,1
```

Input File

Record Fields

	A	B	C	D
Rec1	4	Y	10	Z
Rec2	3	X	15	W
Rec3	7	Y	24	Z
Rec4	5	Y	14	W
Rec5	1	Y	11	Z

Output File

Record Fields

	C	B	A
Rec4	14	Y	5
Rec1	10	Y	4
Rec5	11	Y	1

Field D has been dropped entirely. Field B has retained its original place, and fields A and C have been rearranged. The records are ordered in descending sequence of values in field A, the reverse of the default sequence.

Examples of Record Paragraphs

Example 1:

```
RECORD:  INCLUDE = (40 4 EQ CHAR 'MAIN') AND (30 2 LT 32 2 SBIN)
         KEY     = (10 5 CHAR), (15 3 PDEC RV)
         SUM     = (25 5 PDEC), (50 4 SBIN)
```

INCLUDE specifies that records to be included in the sort process must meet the following criteria:

- The character string MAIN must appear in bytes 40 through 43.
- The fixed binary value in bytes 30 and 31 must be less than the fixed binary value in bytes 32 and 33.

KEY nominates the following fields:

- Major key is an alphanumeric character string in bytes 10 through 14.
- Minor Key is a packed decimal value in bytes 15 through 17 which is to be sorted in reverse order to the sequence specified for the output file.

SUM establishes the following two summation fields:

- Five-byte packed decimal field beginning in byte 25.
- Four-byte fixed binary field beginning at byte 50.

Example 2:

```
RECORD: KEY = (8 2 SBIN RV) (15 5 CHAR)
        OMIT = (1 7 NEG PDEC)
        OMIT = (1 7 ZERO PDEC)
```

The effects of the RECORD paragraph parameters shown above are as follows.

OMIT is specified twice, establishing two alternatives for record exclusion: first, if field 1,7 is negative, the record is excluded. Otherwise, the second OMIT is processed and if field 1 7 is zero, the field is excluded. If neither condition is true (that is, field 1,7 is positive, the record is not excluded from the sort process.

KEY establishes the following two key fields:

- Major key is a signed fixed binary value in bytes 8 through 9 to be sorted in reverse order from the order prevailing for the sort as a whole.
- Minor key is the 5-character field beginning at byte 15.

END PARAGRAPH

The last item in a set of DSL statements must be the paragraph:

END:

There are no parameters associated with it.

SECTION IV

DATA SERVICES LANGUAGE FOR $MERGE

As with the sort process, merge DSL statements are provided via the COMFILE JCL parameter group. They are organized into FUNCTION and RECORD paragraphs, and are terminated by the paragraph END. The keywords and statement within these paragraphs are similar to those associated with Sort, although a different meaning will be attached to some keywords.

The input to $MERGE consists of up to eight sequential files, all of which have identical file attributes such as record length, block length and record format. The files must all have been ordered on the basis of an identical key field or set of key fields. One of these input files, specified by the INFILE1 parameter group in the $MERGE JCL statement, is treated during the merge process as the primary input file; all other input files will be referred to as secondary files in the following description. The difference between primary and secondary files will be explained later in this section.

The output file produced by $MERGE is ordered in the same sequence (i.e., ascending or descending) based on the same set of key fields as the primary file. ★

FUNCTION PARAGRAPH

The FUNCTION paragraph describes overall characteristics of the merge process. All of its parameters are optional and the user may leave out the paragraph altogether if the associated default options serve his purpose. If present, the FUNCTION paragraph must precede the RECORD paragraph.

The general form of a merge FUNCTION paragraph is shown below.

```
FUNCTION:  [DESCEND]   [DELETE]  [COLLATE = { EBCDIC      }  ]
                                            { G100       }
                                            { H200       }
                                            { ASCII      }
                                            { 'hexadecimal-string' }
```

The parameters DESCEND, and COLLATE have the same meaning as in the SORT process; DELETE is interpreted differently, as will be seen in Table 4-1.

Table 4-1. FUNCTION Paragraph Parameters for $MERGE

Parameter	Purpose
DESCEND	When present, declares that the input files are in descending order and that the output file will also be in descending order.
DELETE	When present, DELETE specifies that multiple occurrences of records having the same key field values will cause all but one of the records to be excluded from the output file.
COLLATE	Specfies the collating sequence to be used. The $MERGE collating sequence must be the same as that used in the preceding $SORT executions. For further details see the description of COLLATE in Section III.

RECORD PARAGRAPH

The RECORD paragraph for a merge process is the same as that for a sort:

```
RECORD:   KEY[S] = key-element [key-element] ...

          [ { INCLUDE = condition-elemt [AND condition-elemt]...
              [INCLUDE = condition-elemt [AND condition-elemt] ... ... }
          [ { OMIT = condition-elemt [AND condition-elemt]...
              [OMIT = condition-elemt [AND condition elemt]... ... }

            [SUM = sum-element [sum-element]...]

          [ARRANGE = arrange-element [arrange-element] ...]
```

The syntactic models and descriptions of the elements listed above have already been described in detail in Section III.

The statements are interpreted largely as for the sort with some differences.
KEYS is used to specify key fields. Since it is assumed that the primary merge file is in the order desired for output, the key fields are used for reference when merging. That is, a secondary merge file record with the key value A will be placed on the output file contiguous to the primary record, or set of records whose key field value is A. Minor key fields are likewise used to merge secondary records in the appropriate position within the file.

INCLUDE and OMIT work exactly as for the sort but are applied to all the input files.

The SUM statement as applied to the merge is based the primary-file-first order fof precedence for retaining one of a group of records with the same key field values. Thus, the sum of the specified field will automatically be stored on the record which is to be retained. If a SUM statement is present then DELETE need not, and must not, be specified in an associated FUNCTION Paragraph.

The ARRANGE statement works as for the sort with the same restrictions on sum fields.

MERGE EXAMPLE

```
RECORD:  KEYS = (1 1)  (2 1)

END:
```

INFILE1 (primary File)		INFILE2		Output File	
Record Fields		Record Fields		Record Fields	Key Fields
A 1 2		A 2 1		A 1 2	A 1
A 1 3		A 3 5		A 1 3	A 1
A 2 5		A 7 4		A 2 1	A 2
B 0 0		A 8 0		A 2 5	A 2
B 1 0		B 0 1		A 3 5	A 3
				A 7 4	A 7
				A 8 0	A 8
				B 0 0	B 0
				B 0 1	B 0
				B 1 0	B 1

APPENDIX A

SORT WORK FILE SPACE

Sort work file space may reside on disk, MSU0310 or MSU0400/MSU0350, or may use magnetic tape volumes that meet the NATIVE volume format standard and have been prepared using $VOLWORK.

DISK WORK FILE

A disk work file may be permanent or temporary and must always be allocated by cylinder.

If no work space information is coded in $SORT then a temporary file named SORTWORK, resident on the system disk will be used. The size of this file is 10 cylinders.

CALCULATION OF DISK WORK FILE SIZE

Sort Record Size

The size of the record handled internally by the sort is significant in determining the necessary size for a sort work file. Its value is dependent upon the sort processing options being used.

Calculation of the sort record size for the various cases is as follows:

Let the input record size be i bytes and the sort record size be s bytes.

1. OUTPUT = DATA, DUPREC = NONE
   No ARRANGE elements.
   s = i
2. OUTPUT = DATA, DUPREC = FIFOP
   No ARRANGE elements.
   s = i + 4 bytes
3. OUTPUT = DATA, DUPREC = NONE
   ARRANGE present.

   a. All key fields are contained within the ARRANGE specification

      s = sum of the sizes of the ARRANGE fields.

   b. Some key fields are not present in the ARRANGE fields.

   s = sum of ARRANGE + sum of KEY fields not present in ARRANGE.

   For example, RECORD: KEYS = (10, 4, 30, 6)
   ARRANGE = (4, 5, 20, 3, 10, 4)

   s = (5 + 3 + 4) + (6)
   ARRANGE excluded KEY
   s = 18

   If DUPREC = FIFO, add four bytes to the value of s.

4. OUTPUT = ADDATA, DUPREC = NONE add 5 to the value of s obtained in step 1,2 or 3.

5. OUTPUT = ADDROUT or KEYADDR
   s = sum of KEY fields + 5
   IF DUPREC = FIFO, S = sum of KEY fields + 5 + 4

For example, FUNCTION: OUTPUT = ADDROUT, DUPREC = NONE
RECORD: KEYS = (10, 7, 19, 15, 80, 20)

s= (7 + 15 + 20) + 5 = 47

CALCULATION OF NUMBER OF CYLINDERS

MSU0310

Use this table for R to find the smallest entry (e) such that its value is greater than s (sort record size):

R = (475, 971, 1457, 1963, 2955, 5931)

Calculate b, where be is the integer part of e/s

For example, s = 150 then e = 475
b = int (475 / 150) = 3

If the remainder of e /s > 1/2s, use the next higher entry to establish a new entry.

For example, s = 250
e = 475
e / s = 1 with a remainder of 225 which is greater than (1/2) 250

Use e = 971. Then b = int(971/250) = 3

Let t = number of blocks per track

where t = int (5931/e)

If size of input file is n records, then size of work file in cylinders is as follows:

number of cylinders = n/(20*t*b) rounded up

MSU0400/MSU0350

Use this table for R to find the smallest entry (e) such that its value is greater than s (sort record size):

R = (475, 971, 1963, 2459, 4939)

Calculate b where b is the integer part of e /s

For example, s = 150 then e = 475
b = int (475 / 150) = 3

If the remainder of e/s > 1/2s, use the next higher entry to establish a new value.

Let t = number of blocks per track, where t = (9899/e) rounded down

If the size of the input file is n records, then the size of the work file in cylinders is as follows:

number of cylinders = n/(19*t*b) rounded up

If REPEAT is specified then increase the computed size by 20%.

Example 1: MSU0310

```
FUNCTION:  OUTPUT = DATA, DUPREC = FIFO
RECORDS:   KEYS = (20,  7)
ARRANGE = (1, 60 80, 50)

END:
```

Calculate s:

s = 60 + 50 + 4 = 114

Using the table R for MSU0310 select:

e = 475

Therefore b = e/s = 4

The remainder of e/s is 19 which is less than (1/2)s, (=57). Therefore it is not necessary to choose a higher value from R.

Calculate t, the number of blocks per track:

t = int(5931/e) = int(5931/475) = 12

If the file is to contain 20000 records then

number of MSU0310 cylinders = 20000/(209*t*b) rounded up

= 20000/20*12*4)

= 21 cylinders.

Example 2: MSU0400

```
FUNCTION:  OUTPUT = KEYADDR
RECORD:    KEYS = 1,100 150,90  200,100
END:
```

Calculate s:

s = 100 + 90 + 100 +5 = 295

Using the MSU0400 table for R select:

e = 475

Therefore b = e/s = 1

However the remainder 180 > (1/2) 295 so choose

e = 971 , b = e/s = int(971/295)=3

In this case, the remainder 86 < (1/2)295.

Calculate t, the number of blocks per track:

t = int(9899/e) = int(9899/971) = 10

If the file is to contain 15000 records then

number of MSU0400 cylinders = 15000/(19*t*b) rounded up

= 15000/(19*10*3)

= 27 cylinders

★

MAGNETIC TAPE WORK SPACE

When the work space for the sort is on magnetic tape the I/O tape volumes must have the attribute WORK. The volumes to be used should be prepared using $VOLWORK. This utility is described in the manual "GCOS Level 64 Data Management Utilities".

★

APPENDIX B

SORT/MERGE REPORT

The contents of the Sort Report which is delivered to SYSOUT is determined by the value assigned by the user to the REPORT keyword. Specifiable values are AUDIT, PARAM, ALL, or NONE. In the examples that follow it should be understood that the appearance of Sort Execution Diagnostics or Sort Parameter Diagnostics would terminate the report.

REPORT = ALL

```
                          *****   SORT REPORT   *****

***** SORT PARAMETER LISTING *****

RECORD: KEY = 1 10 CHR
        SUM = A 2 SBIN
        INCLUDE = 12 2 EA 'MA'

***** SORT PARAMETER DIAGNOSTICS *****

* * * F 750028 SYNTAX ERROR ,INVALID DATA TYPE PARAMETER
        "KEY = 1 10 CHR"

* * * F 750023 SYNTAX ERROR ,INVALID POSITION PARAMETER
        "SUM = A 2  SBIN"

* * * F 750008 SYNTAX ERROR, 'END' MISSING

***** SORT EXECUTION DIAGNOSTICS *****

* * * F 750524 OUTPUT FILE NOT ASSIGNED

***** SORT AUDIT INFORMATION *****

INPUT FILE IS XXXXX
OUTPUT FILE IS YYYYY

TOTAL NUMBER OF RECORD READ IS 4000
TOTAL NUMBER OF RECORDS OMITTED IS 1000
TOTAL NUMBER OF RECORDS DELETED BY SUM IS 100
TOTAL NUMBER OF DUPLICATE RECORDS DELETED IS 0
TOTAL NUMBER OF INVALID RECORD IS 0
TOTAL NUMBER OF RECORDS WRITTEN IS 2900

***** END OF SORT REPORT *****
```

REPORT = AUDIT (default)

```
                          ***** SORT REPORT *****

***** SORT PARAMETER DIAGNOSTICS *****

* * * F 750028 SYNTAX ERROR, INVALID DATA TYPE PARAMETER
        "KEY = 1 10 CHR"

* * * F 750023 SYNTAX ERROR, INVALID POSITION PARAMETER
        "SUM = A 2 SBIN"

* * * F 750008 SYNTAX ERROR, 'END' MISSING

***** SORT EXECUTION DIGNOSTICS *****

* * * F 750524 OUTPUT FILE NOT ASSIGNED


***** SORT AUDIT INFORMATION *****

INPUT FILE IS XXXX
OUTPUT FILE IS YYYY

TOTAL NUMBER OF RECORDS READ IS 4000
TOTAL NUMBER OF RECORDS OMITTED IS 10000
TOTAL NUMBER OF RECORDS DELETED BY SUM IS 100
TOTAL NUMBER OF DUPLICATE RECORDS DELETED IS 0
TOTAL NUMBER OF INVALID RECORDS IS 0
TOTAL NUMBER OF RECORDS WRITTEN IS 2900

                          ***** END OF SORT REPORT *****
```

REPORT = PARAM

```
                              ***** SORT REPORT *****

     ***** SORT PARAMETER LISTING *****

     RECORD: KEY = 1 10 CHR
             SUM = A 2 SBIN
             INCLUDE = 12 2 EQ 'MA'


     ***** SORT PARAMETER DIAGNOSTICS *****

     * * * F 750028 SYNTAX ERROR, INVALID DATA TYPE PARAMETER

             "KEY = 1 10 CHR"

     * * * F 750023 SYNTAX ERROR, INVALID POSITION PARAMETER
             "SUM = A 2 SBIN"

     * * * F 750008 SYNTAX ERROR, 'END' MISSING


     ***** SORT EXECUTION DIAGNOSTICS ******

     * * * F 750524 OUTPUT FILE NOT ASSIGNED

                              ***** END OF SORT REPORT *****
```

REPORT = NONE

```
                              ***** SORT REPORT *****

     ***** SORT EXECUTION  DIAGNOSTICS *****

     * * * F 750524 OUTPUT FILE NOT ASSIGNED

                              ***** END OF SORT REPORT *****
```

The MERGE report has the same format as the SORT report, but the word SORT is replaced by the word MERGE in report headings.

APPENDIX C

ERROR MESSAGES FOR SORT/MERGE

The diagnostics which can appear in the JOR (Job Occurrence Report) are caused by syntax or inconsistency errors in statements $SORT, $MERGE, $SORTFMT and $SORTWORK. Most of the messages are self explanatory. With each message there is an associated error severity code - Fatal(F) or warning (W). If a Fatal error is encountered then execution is abandoned.

Note also that after the expansion of an extended JCL statement, errors may later be found in the resulting basic JCL statements (such as $ASSIGN). These errors in basic JCL are described in the JCL Reference Manual.

Sort and Merge set the status variable that can be tested with appropriate JCL statements. The possible values are:

- SEV0 : the function has been correctly executed.
- SEV1 : the function has been correctly executed. However a warning is given to the user who should carefully examine the report. (e.g. : an empty output file has been created. The situation is normal for sort if, for example, the input file was empty. But the specification of an empty input file may be the result of a user error).
- SEV3 : (Sort only). The function has not been completed, due to an I/O error. If the REPEAT option was used, the function can be restarted (see Appendix I.)
- SEV4 : the function has not been completed, due to a user error or a system error whose origin cannot be precisely diagnosed.

Table C-1 shows the messages that can result from errors in the $SORT JCL statement.

Table C-1. ERROR MESSAGES FROM $SORT JCL STATEMENT

Error number	Text and explanation
106	CONTRADICTORY VALUES IN LABEL AND FILEFORM PARAMETERS Verify both parameters. A HFAS fileform implies a H200 type label, a standard label implies a native fileform. CONTRADICTORY VALUES IN INFILE AND WKTAPE PARAMETERS Appears when REPLIN is requested in WKTAPE and the device class of the output file and work tapes are different. CONTRADICTORY VALUES IN OUTFILE AND WKTAPE PARAMETERS Appears when REPLOUT or USEOUT is requested in WKTAPE and the device class of the output file and work tapes are different. CONTRADICTORY VALUES IN HALT AND START PARAMETERS Verify both values. The value given for HALT must be greater than or equal to the value given for START. CONTRADICTORY VALUES IN WKTAPE AND WKDISK PARAMETERS WKTAPE and WKDISK are mutually exclusive. CONTRADICTORY VALUES IN ABORT AND CONTINUE PARAMETERS In INVREC, ABORT and CONTINUE are mutually exclusive. ERROR IN OUTFILE PARAMETER; ILLEGAL VALUE OF MEDIA IN CASE OF REPLIN When a disk sort is executed, then if the input file and the work file are on the same media the REPLIN option is not allowed in OUTFILE because it would cause the work file to be dismounted before the end of sort. ERROR IN FILESTAT PARAMETER;ILLEGAL VALUE TEMPRY FOR H200 FILES The concept of temporary file does not exist for HFAS files.

Table C-1 (cont.) ERROR MESSAGES FROM $SORT JCL STATEMENT

Error number	Text and explanation
106 (cont.)	ERROR IN COMFILE PARAMETER;ILLEGAL VALUE INPUT-ENCLOSURE-NAME Verify the syntax of the input enclosure name. ERROR IN WKTAPE PARAMETER;ILLEGAL VALUE DEVCLASS TYPE MUST BE MT ERROR IN WKTAPE PARAMETER;ILLEGAL VALUE NBDV MUST BE > = 3 OR < = 6 ERROR IN SIZE PARAMETER;ILLEGAL VALUE MUST BE > = 22K ERROR IN MEMORY PARAMETER;ILLEGAL VALUE MUST LIE BETWEEN 8 AND 120K ERROR IN START PARAMETER;ILLEGAL VALUE MUST LIE BETWEENE 1 AND 2 147 483 647 ERROR IN HALT PARAMETER;ILLEGAL VALUE MUST LIE BETWEEN 1 AND 2 147 483 647 ERROR IN BUFFER PARAMETER;ILLEGAL VALUE MUST BE < OR = TO 32767 ERROR IN INFILE PARAMETER;ILLEGAL VALUE REPLIN OR SIZE Neither REPLIN nor SIZE should appear in INFILE description. ERROR IN WKDISK PARAMETER;ILLEGAL VALUE FILESTAT,END,MOUNT or EXPDATE None of these parameters is allowed in WKDISK ERROR IN WKDISK PARAMETER;ILLEGAL VALUE LABEL OR SORT DEFINITION PAR. Neither LABEL nor sort-definition parameters should appear under WKDISK. The file used as work file has always native labels. ERROR IN WKDISK PARAMETER;ILLEGAL VALUE SUBFILE SUBFILE is not a valid keyword in WKDISK description.

Table C-1. (cont.) ERROR MESSAGES FROM $SORT JCL STATEMENT

Error number	Text and explanation
106 (cont.)	ERROR IN COMFILE PARAMETER;ILLEGAL VALUE MOUNT OR EXPDATE Neither MOUNT nor EXPDATE are valid keywords for the COMFILE description. ERROR IN COMFILE PARAMETER;ILLEGAL VALUE LABEL OR MEDIANAME TABLE LABEL is not a valid keywork in COMFILE description. The library is assumed to be native. The library cannot be a multivolume file. ERROR IN COMFILE PARAMETER;ILLEGAL VALUE SORT DEFINITION FORBIDDEN Parameters of sort definition are not allowed for COMFILE which must be an already created library. ERROR IN PRTFILE PARAMETER;ILLEGAL VALUE SORT DEFINITION PARAMETERS The PRTFILE is either created and there is no need to specify sort-definition parameters, or the PRTFILE is a tape file to be created by the Sort and default values will be used. ERROR IN ERROPT PARAMETER;ILLEGAL VALUE ONLY PRINTID OR IGNORE FOR H200 FILES ERROR IN WKDISK PARAMETER;ILLEGAL VALUE DATABUF,BPB OR REPLIN None of these keywords is allowed in WKDISK. ERROR IN COMFILE PARAMETER;ILLEGAL VALUE DATABUF, BPB OR REPLIN None of these keywords is allowed in COMFILE. ERROR IN PRTFILE PARAMETER;ILLEGAL VALUE DATABUF,BPB,REPLIN OR SIZE None of these keywords is allowed in PRTFILE. ERROR IN DATABUF PARAMETER;ILLEGAL VALUE MUST BE EQUAL TO 1 OR 2

Table C-1. (cont.) ERROR MESSAGES FROM $SORT JCL STATEMENT

Error number	Text and explanation
106 (cont.)	ERROR IN BPB PARAMETER:ILLEGAL VALUE MUST BE GREATER OR EQUAL TO 1 ERROR IN LOGFILE PARAMETER:ILLEGAL VALUE H200 FILE FORBIDDEN ERROR IN LOGFILE PARAMETER:ILLEGAL VALUE DATABUF,BPB OR REPLIN
108	ERROR IN LABEL PARAMETER CONTRADICTORY VALUES IN INPUT AND OUTPUT FILES HFAS input must produce HFAS output, and HFAS output can only be obtained from HFAS input. ERROR IN DEVCLASS PARAMETER CONTRADICTORY VALUES IN INPUT AND OUTPUT FILES When REPLIN is used in OUTFILE, the input and output file must be of the same device class.
110	CONTRADICTORY PARAMETERS RESIDENT,DEVCLASS When a resident volume is described, the device class should not be specified.
111	MEDIA PARAMETER MAY APPEAR ONLY IF DEVCLASS EXPLICITLY GIVEN No default value is supplied for DEVCLASS. So when the volume is not resident, DEVCLASS and MEDIA must be specified together.
112	ERROR IN NOBSN PARAMETER : NOBSN MAY NOT APPEAR IN THE PRESENT CONTEXT OF DISK FILE NOBSN use is restricted to tape files. ERROR IN BANCHAR PARAMETER : BANCHAR MAY NOT APPEAR IN THE PRESENT CONTEXT OF DISK OR NON H200 TAPE FILE BANCHAR is meaningful only for HFAS tape files. ERROR IN H200FORM PARAMETER : H200FORM MAY NOT APPEAR IN THE PRESENT CONTEXT OF DISK OR NON H200 TAPE FILE H200FORM is meaningful only for HFAS tapes files.

Table C-1.(cont.) ERROR MESSAGES FROM $SORT JCL STATEMENT

Error number	Text and explanation
112 (cont.)	ERROR IN WKDISK PARAMETER : SIZE MAY NOT APPEAR IN THE PRESENT CONTEXT (EFN ALREADY DEFINED) external file name and SIZE are mutually exclusive in WKDISK ERROR IN COMFILE PARAMETER : DEVCLASS MAY NOT APPEAR IN THE PRESENT CONTEXT WITHOUT MEDIA ERROR IN WKTAPE PARAMETER : USEOUT MAY NOT APPEAR IN THE PRESENT CONTEXT OF NON NATIVE OUTFILE USEOUT is restricted to native tapes. ERROR IN WKTAPE PARAMETER : USEOUT MAY NOT APPEAR IN THE PRESENT CONTEXT OF REPLOUT USEOUT and REPLOUT are mutually exclusive.
113	MANDATORY PARAMETER RECSIZE MISSING MANDATORY PARAMETER BLOCKSZ MISSING MANDATORY PARAMETER DEVCLASS MISSING MANDATORY PARAMETER SUBFILE MISSING MANDATORY PARAMETER H200FORM MISSING MANDATORY PARAMETER INFILE MISSING MANDATORY PARAMETER OUTFILE MISSING MANDATORY PARAMETER COMFILE MISSING MANDATORY PARAMETER EFN MISSING EFN stands for external file name MANDATORY PARAMETER LOGFILE MISSING
126	CONTRADICTORY VALUES : MEMORY, BUFFER-SIZE,DATABUF,BPB (MEMORY,BUFFER) form a group of parameters exclusive from the group (SIZE,DATABUF,BPB).

Table C-2 shows the messages that can result from errors in the $MERGE JCL statement.

Table C-2. ERROR MESSAGES FROM $MERGE JCL STATEMENT

Error number	Text and Explanation
106	CONTRADICTORY VALUES IN LABEL AND FILEFORM PARAMETERS Verify both parameters. A HFAS fileform implies a H200 type label, a standard label implies a native fileform. CONTRADICTORY VALUES IN SIZE AND FILESTAT PARAMETERS
107	ERROR IN FILESTAT PARAMETER;ILLEGAL VALUE TEMPRY FOR H200 FILES The concept of temporary file does not exist for HFAS files. ERROR IN COMFILE PARAMETER;ILLEGAL VALUE INPUT-ENCLOSURE-NAME Verify the syntax of the input enclosure name. ERROR IN COMFILE PARAMETER;ILLEGAL VALUE OF MOUNT OR EXPDATE Neither MOUNT not EXPDATE are allowed for COMFILE. ERROR IN COMFILE PARAMETER;ILLEGAL VALUE LABEL OR MEDIANAME TABLE LABEL is not a valid keyword in COMFILE description. The library is assumed to be native. The library cannot be a multivolume file. ERROR IN COMFILE PARAMETER;ILLEGAL VALUE SORT DEFINITION FORBIDDEN Parameters of sort definition are not allowed for COMFILE which must be an already created library. ERROR IN PRTFILE PARAMETER;ILLEGAL VALUE SORT DEFINITION FORBIDDEN The PRTFILE is either created and there is no need to specify sort-definition parameters, or the PRTFILE is a tape file to be created by the sort and default values will be used.

Table C-2.(cont.) ERROR MESSAGES FROM $MERGE JCL STATEMENT

Error number	Text and explanation
107 (cont.)	ERROR IN COMFILE PARAMETER:ILLEGAL VALUE DATABUF,BPB OR SIZE None of these keywords is allowed in COMFILE. ERROR IN PRTFILE PARAMETER:ILLEGAL VALUE DATABUF,BPB OR SIZE None of these keywords is allowed in PRTFILE. ERROR IN DATABUF PARAMETER:ILLEGAL VALUE MUST BE EQUAL TO 1 OR 2 ERROR IN BPB PARAMETER:ILLEGAL VALUE MUST BE GREATER OR EQUAL TO 1
108	ERROR IN LABEL PARAMETER CONTRADICTORY VALUES IN INPUT AND OUTPUT FILES HFAS input must produce HFAS output. HFAS output can only be obtained from HFAS input.
110	CONTRADICTORY PARAMETERS RESIDENT,DEVCLASS When a resident volume is described, the device class should not be specified.
111	MEDIA PARAMETER MAY APPEAR ONLY IF DEVCLASS EXPLICITLY GIVEN
112	ERROR IN NOBSN PARAMETER : NOBSN MAY NOT APPEAR IN THE PRESENT CONTEXT OF DISK FILE NOBSN is meaningful only for tape files. ERROR IN BANCHAR PARAMETER : BANCHAR MAY NOT APPEAR IN THE PRESENT CONTEXT OF DISK OR NON H200 TAPE FILE BANCHAR is meaningful only for HFAS tape files. ERROR IN H200FORM PARAMETER : H200FORM MAY NOT APPEAR IN THE PRESENT CONTEXT OF DISK OR NON H200 TAPE FILE H200FORM is meaningful only for HFAS tape files. ERROR IN COMFILE PARAMETER : DEVCLASS MAY NOT APPEAR IN THE PRESENT CONTEXT WITHOUT MEDIA
113	MANDATORY PARAMETER RECSIZE MISSING

Table C-2. (cont.) ERROR MESSAGES FROM $MERGE JCL STATEMENT

Error number	Text and Explanation
113 (cont.)	MANDATORY PARAMETER BLOCKSZ MISSING MANDATORY PARAMETER SUBFILE MISSING MANDATORY PARAMETER H2OOFORM MISSING MANDATORY PARAMETER OUTFILE MISSING MANDATORY PARAMETER COMFILE MISSING
126	ERROR IN INPUT FILE;ILLEGAL VALUE SIZE The SIZE keyword cannot appear in an input file description since the file already exists.

ERROR MESSAGE FORMAT FOR SORT/MERGE UTILITY

The error message format is as follows:

* * * {SORT / MERGE} YYY error message [return code] "secondary information"

YYY is a 3-digit number associated with a given message.

Error message - a phrase that describes the diagnosis or error. For example:

, KEYWORD INVALID IN RECORD

Return code - a class of error messages which can be identified as beginning with the words SYSTEM ERROR ... occur when the return code from a primitive execution cannot be interpreted by the Sort/Merge program. Such error messages are followed by a hexadecimal character representation of the specific return code.

Secondary information - secondary information, when supplied provides data that allows the user to be more readily aware of the language element that has been analyzed as erroneous. There are, however, situations where an error in a parameter string inhibits intelligent analysis of the string in the vicinity of that error. When this occurs, the secondary information may be a string of parameters that has not been analyzed. In addition some processing error messages are also supplemented by a secondary message.

Example:

RECORD: INCD = 10 2 EQ 20 2 SBIN AND 20 4 EQ CHR 'ABCD'

INCLUDE = 12 A EQ 40 3 PDEC AND 90 1 GT 1001 PDEC

Error messages

* * * SORT020 SYNTAX ERROR, INVALID KEYWORD
"INCD = 10 2 EQ 20 2 SBIN AND 20 4 EQ CHAR 'ABCD'"

* * * SORT0024 SYNTAX ERROR, INVALID LENGTH PARAMETER
"12 A EQ 40 3 PDEC"

Note that in the example the second condition-element of the INCLUDE= does not appear in the secondary information, since the presence of the word AND allowed analysis to correctly interpret the parameters. Usually, secondary information will be terminated by an element that immediately precedes a key-word, a paragraph header or, in the case of tests such as INCLUDE, the next condition element. It can also be terminated by the last element of the parameter string. Table C-3 lists the DSL dianostic messages, and explains the circumstances which give rise to their appearance.

Table C-3. DSL DIAGNOSTICS

YYY	Text and Explanation
001	SYNTAX ERROR,INITIAL PARAGRAPH HEADER MISSING Message issued when the first DSL word is not a valid paragraph header.
002	SYNTAX ERROR, 'RECORD' PARAGRAPH MISSING Message issued when the mandatory RECORD paragraph does not appear in the DSL statements.
003	INVALID COLLATING SEQUENCE FOR HFAS FILE Message issued when the ASCII or G100 collating sequence is requested for an HFAS file. If it is necessary to sort an HFAS file with a collating sequence which is neither EBCDIC nor H200, the collating sequence required should be entered as a private one.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
004	SYNTAX ERROR,KEYWORD INVALID IN 'FUNCTION' PARAGRAPH Message issued when an unexpected word appears in the FUNCTION paragraph. Two cases are frequent: . A punch error was made (e.g. COLATE instead of COLLATE) . The RECORD paragraph definition was omitted.
005	SYNTAX ERROR, KEYWORD INVALID IN 'RECORD' PARAGRAPH Messages issued when an invalid word appears in the RECORD paragraph. Here invalid means either: . Unknown, for example KAYS was specified instead of KEYS, or . Contextually non-significant, e.g, KEYS = INCLUDE = The INCLUDE keyword is not valid in this context.
007	SYNTAX ERROR,INVALID PARAGRAPH HEADER Message issued when an expected paragraph header is not found in the DSL file.
008	SYNTAX ERROR,'END' MISSING The DSL commands are not terminated by an END: which indicates the end of DSL specification. The DSL file should be corrected and the step re-submitted.
012	SYNTAX ERROR,PARAMETERS FOLLOW 'END' Characters other than blanks or comments are not allowed after the END: expression which marks the end of DSL specification. Verify the DSL subfile or input enclosure.
020	SYNTAX ERROR,INVALID KEYWORD Message issued when a keyword appears at a place where it should not. FUNCTION: ASCII will produce such a message.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
020 (cont.)	SYNTAX ERROR,EQUAL MISSING Message issued eacn time an expected equal sign is not found. OUTPUT ADDATA will produce such a messaqe
022	SYNTAX ERROR,INVALID PARAMETER messaqe issued when an unexpected value is specified for a parameter which is not a self identifying value, i.e., a parameter with the form keyword = value. SORTSIZE = 16A will produce such a message.
023	SYNTAX ERROR,INVALID POSITION PARAMETER Message issued when, in eitner KEYS, SUM, ARRANGE, INCLUDE or OMIT, the starting position of a field is not an unsigned numeric. SUM = 1A 4 SBIN for example will produce such a message since 1A is not a numeric value.
024	SYNTAX ERROR,INVALID LENGTH PARAMETER Messaqe issued when, in either KEYS, SUM, ARRANGE, INCLUDE or OMIT, the length of a field is not an unsigned numeric or is equal to 0. ARRANGE = 12 2C will produce such a messaqe since 2C is not a numeric value.
025	SYNTAX ERROR,INVALID OPERATOR Self explanatory.
026	SYNTAX ERROR,'AND' MISSING Diagnosed in an INCLUDE or OMIT clause when two condition elements are not related by an AND operator.
027	SYNTAX ERROR,'INCLUDE'/'OMIT' Messaqe issued when the syntax of an INCLUDE or OMIT clause is not understandable.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
028	SYNTAX ERROR,INVALID DATATYPE PARAMETER Self explanatory.
029	SYNTAX ERROR,INVALID LITERAL PARAMETER Message issued when a literal parameter has not the expected value, i.e., has either an invalid length (too long or too short) or is not in the desired format (i.e. non numeric when numeric implied by data type, etc...).
038	'KEY' OVERLAP IN CASE OF A NON STANDARD COLLATING SEQUENCE Message produced when a forbidden case of overlapping keys is detected. Overlapping keys are not allowed for HFAS files when COLLATE is not equal to H200, for other organizations when COLLATE is not equal to EBCDIC.
039	SYNTAX ERROR,'KEY' MISSING KEY is mandatory in SORT and MERGE.
040	'SUM' OVERLAP Fields defined in the SUM clause are not allowed to overlap.
041	'SUM' AND 'KEY' OVERLAP No overlapping is allowed between fields used for summation and fields used for key comparison.
042	'SUM' OMITTED IN 'ARRANGE' Message issued when the fields specified in the SUM clause are excluded from the output record by the ARRANGE clause.
051	DUPLICATE KEYWORD ENTRY Issued when the same keyword is encountered twice and duplication is not allowed. For example: FUNCTION : SORTSIZE = 10000 DESCEND OUTPUT = DATA DESCEND

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
058	INVALID COLLATING SEQUENCE LENGTH Issued when a private collating sequence is specified. For HFAS files, 64 positions must be specified, 256 for other organizations.
070	PARAMS INCONSISTENT,DATA TYPE - LENGTH Where a data type can be specified, the field length must be compatible with Level 64 data types (e.g. SBIN allows for a length of 2 or 4 only)
071	PARAMS INCONSISTENT,LITERAL-DATA TYPE In the INCLUDE or OMIT clauses, when a relational expression involves a literal and a data type, the literal and the data type must be consistent. For example: INCLUDE = 1,4 GT SBIN 1A is inconsistent since 1A is not a signed binary value.
073	PARAMS INCONSISTENT,LITERAL-LENGTH In the INCLUDE or OMIT clauses, when a relational expression involves a literal, the literal length must be consistent with the length of the field to which it is compared.
076	PARAMS INCONSISTENT, 'OUTPUT' - 'SUM' Message issued when summation is requested along with an output record of the form KEYADDR or ADDROUT.
077	PARAMS INCONSISTENT,'OUTPUT'-'ARRANGE' Message issued when output record arrangement is requested and the output record format is either KEYADDR or ADDROUT.
078	PARAMS INCONSISTENT,'SUM'-'DELETE' Message issued when summation is requested along with duplicate record deletion.
080	TOTAL NUMBER OF CONDITIONALS GT 16 No more than 16 condition elements are allowed in INCLUDE/OMIT.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
081	TOTAL LITERAL LENGTH GT 512 The total number of literal characters used in INCLUDE/OMIT must not exceed 512.
082	TOTAL NUMBER OF 'KEY' FIELDS GT 64 The total number of key fields specified for record ordering must not exceed 64.
083	TOTAL NUMBER OF 'SUM' FIELDS GT 16 The total number of fields specified in the SUM clause must not exceed 16.
084	TOTAL NUMBER OF 'ARRANGE' FIELDS GT 128 The total number of fields specified in the ARRANGE clause must not exceed 128.
200	TOTAL NUMBER OF PARAMETER CHARACTERS GT 4500 The total number of significant characters in the input exclosure must not exceed 4500 (several contiguous spaces are counted as one character). This limit can be increased on request. (The characters in comments are considered significant).
500	SYSTEM ERROR - INPUT FILE This message is followed by a secondary message giving the system return code which caused the message to be issued and an internal sort identifier for debugging purposes. The system return code will have been set by open or retrieve file definition functions.
501	I/O ERROR - INPUT FILE This message is output when a read record on the input file is not successful. It is followed by a secondary message that has the same format as for message 500.
502	SYSTEM ERROR - ON CLOSE INPUT FILE Wrong termination of the close function for the input file. The secondary message that follows has the same format as for message 500.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
503	SYSTEM ERROR - OUTPUT FILE Message followed by a secondary message giving the system return code which caused the message to be output. The return code will have been set by open or retrieve file definition functions.
504	I/O ERROR - OUTPUT FILE An error occurred while writing a record onto the output file. The secondary message that follows has the same format as for message 503.
505	SYSTEM ERROR - ON CLOSE OUTPUT FILE An error occurred while closing the output file. The secondary message that follows has the same format as for message 503.
506	SYSTEM ERROR - WORK FILE Message followed by a secondary message which may have the same format as for message 500 or 503.
507	I/O ERROR - WORK FILE Same secondary message as for message 500 or 503.
508	SYSTEM ERROR - ON WRITE WORK FILE Same secondary message as for message 500 or 503.
509	SYSTEM ERROR - ON CLOSE WORK FILE Same secondary message as for message 500 or 503.
510	INVALID RECORD FORMAT FOR INPUT FILE SORT and MERGE accept F, FB, V, or VB record formats. In case of MERGE, this message can be output when one of the merged files has not the same record format as the master file.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
511	INVALID RECORD SIZE FOR INPUT FILE This message is issued for SORT when the input record length exceeds the maximum record length. In the case of MERGE it is issued when variable length record files with different maximum record lengths are merged.
512	INVALID FILE ORGANIZATION FOR INPUT FILE Message issued when the input file organization is not supported by SORT or MERGE.
513	INVALID RECORD FORMAT FOR OUTPUT FILE Same as 510, but for output file.
514	INVALID RECORD SIZE FOR OUTPUT FILE When record size has not been specified for a file which is not preallocated, this message will be output.
515	INVALID FILE ORGANIZATION FOR OUTPUT FILE Same as 512 for the output file.
516	OUTPUT FILE OVERFLOW Message issued when the end of output file is reached before all the sorted records have been written.
518	INVALID DUMMY ASSIGN FOR INPUT FILE The assignment of a dummy file as input to the SORT and MERGE is considered as an error.
519	EXTERNAL FILE NAME UNKNOWN FOR INPUT FILE The file name specified for the input file has not been found on the input media. The JCL should be checked, since either the file name or the media name, or both, are not correctly specified.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
520	FIELD REFERENCED IN 'INCLUDE'/'OMIT' IS BEYOND INPUT RECORD LENGTH One or more fields specified in INCLUDE or OMIT DSL statements lie outside the record boundaries. The DSL command file should be suitably amended and the step resubmitted.
521	FIELD REFERENCED IN 'ARRANGE' IS BEYOND INPUT RECORD LENGTH. Same as 520, but for ARRANGE.
522	FIELD REFERENCED IN 'SUM' IS BEYOND INPUT RECORD LENGTH. Same as 520, but for SUM.
523	FIELD REFERENCED IN 'KEY' IS BEYOND INPUT RECORD LENGTH Same as 520, but for KEY.
525	INVALID DUMMY ASSIGNED FOR OUTPUT FILE The assignment of a dummy file as output of the SORT or MERGE is not allowed.
526	EXTERNAL FILE NAME UNKNOWN FOR OUTPUT FILE Same as 519, but for output file.
528	INVALID DUMMY ASSIGN FOR SYSIN FILE A dummy file is not allowed for the command file.
529	EXTERNAL FILE NAME UNKNOWN FOR SYSIN FILE Same as 519, but for the command file.
530	INSUFFICIENT SIZE GIVEN The SORT cannot be performed with the working set specified. Increase this value and re-submit the step.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanations
532	INSUFFICIENT WORK SPACE SPECIFIED The size of the disk work file is not sufficient to perform the sort. See the computation formula in Appendix A.
534	SYSTEM ERROR - SYSIN FILE A secondary message follows which gives the system return code that caused the message to be output. The error may occur during the open of the command file.
535	I/O ERROR - SYSIN FILE Same secondary message as for 534. Message issued when a read record operation on the command file terminates abnormally.
536	SYSTEM ERROR - ON CLOSE SYSIN FILE Same secondary message as for message 534. The close operation on the command file has been unsuccessful.
538	INVALID DUMMY ASSIGN FOR WORK FILE The assignment of a dummy file as sort disk work file is not allowed since the sort cannot be executed.
539	EXTERNAL FILE NAME UNKNOWN FOR WORK FILE Same as 519, but for work file.
541	SEQUENCE ERROR During the merging phase of the sort, a sequence error is found in a string. The error can be the result of a previous I/O error. The step can be re-submitted using another drive for the work file. If the error persists, refer to your supporting organization.
542	WORK FILE OVERFLOW Same as message 532.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
545	TOTAL NUMBER OF WORK FILE EXTENTS GT 16 The sort disk work file cannot extend over more than 16 extents. If such an error occurs, preallocate and format your disk work file again.
547	EXTERNAL FILE NAME UNKNOWN FOR LOG FILE Same as 519, but for the file described under LOGFILE in the JCL.
549	SYSTEM ERROR - ON WRITE LOG FILE Error when attempting to write a record onto the log file. A secondary message is displayed which shows the system return code that caused the message to be issued.
550	INCONSISTENT DATA DATA LOSS Message issued when sort discovers, at the end of the output file creation, that the number of output records is less than the number of input records (after substracting the number of omitted records, the number of deleted records and the number of invalid records). If such a message is issued, please inform your supporting organization.
551	INCONSISTENT DATA DATA GAIN Same as 550, but when the number of output records is greater than the number of input records.
552	FIRST RECORD TO BE PROCESSED IS BEYOND END OF FILE Check the START parameter in the $SORT statement. The value given is too high.
553	END OF FILE REACHED BEFORE LAST RECORD TO BE PROCESSED Warning message. The value given in HALT is too high. Sort process will be performed normally and terminated with a SEV1 status, i.e, all the records from the one specified in START up to the last one in the file will be sorted.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
554	SYSTEM ERROR - ON CLOSE LOG FILE The secondary message shows the system return code that caused the message to be printed.
555	LOG FILE OVERFLOW Physical end of log file has been reached before all the invalid record have been output. Specify a bigger file.
556	SYSTEM ERROR - ON OPEN LOG FILE A secondary message shows the system return code that caused the message to be printed.
557	INVALID ERROPT KEYWORD FOR HFAS FILE Only ERROPT = PRINTID is allowed for HFAS file. Another specification will cause this message be printed.
560	SYSTEM ERROR - ASSIGNMENT A secondary message shows the system return code that caused the assignment phase to terminate abnormally. In this case please inform your supporting organization.
561	SYSTEM ERROR - PRESORT Same as 560, but for the internal sort phase.
562	SYSTEM ERROR - MERGE/LAST PASS Same as 560, but for the external sort phase.
563	SYSTEM ERROR - COLLATE Same as 560, but for the merging of files invoked through $MERGE
580	SYSTEM ERROR - ON CHECKPOINT A secondary message shows the system return code given by the checkpoint function. Please inform your supporting organization.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
581	SIZE VALUE IS TOO SMALL The value specified in the SIZE keyword of $SORT (or the default value) is not sufficient. Refer to the SIZE parameter specification in Section II.
582	SIZE VALUE IS TOO LARGE Same as 581, but in this case a SIZE value that is too high was specified.
583	WORK FILE ILLEGAL DEVCLASS The file specified to the sort as disk work file, does not reside on a supported device class.
622	Same as 522
623	SYNTAX ERROR,KEYWORD INVALID FOR HFAS FILE A secondary message shows the invalid keyword.
624	SYNTAX ERROR,PARAMETER INVALID FOR HFAS FILE A secondary message shows the invalid parameter.
625	SYNTAX ERROR,DATA TYPE INVALID FOR HFAS FILE A secondary message shows the wrong sentence. Only characters are valid data types for HFAS files.
626	LITERAL-DATA TYPE INVALID IN 'INCLUDE'/'OMIT' FOR HFAS FILE A secondary message shows the wrong literal. Only characters are valid data types for HFAS files.
627	SYNTAX ERROR,OPERATOR INVALID FOR HFAS FILE A secondary message shows the invalid operator. Only character string operations are allowed for HFAS files.

Table C-3 (cont.) DSL DIAGNOSTICS

YYY	Text and Explanation
628	I/O ERROR - ON READ WORK FILE A secondary message gives the external device name (e.g. MT03 or MS02) where the unrecoverable I/O error was detected and a four-digit hexadecimal code representing the abnormal I/O completion code as returned by PIO. Such a message will be followed by the sort abortion with a SEV3 status, thus enabling the operator to dismount the work volume and mount it on another drive and restart the step from the last checkpoint. For disk sorts the relevant track address is printed: ADDRESS: CYL/TRACK.
629	I/O ERROR - ON WRITE WORK FILE Same as 628, but for write operations.
630	I/O ERROR - WRITE T.M. WORK FILE Same as 628, but when attempting to write a tape mark onto a work tape during a tape sort.
631	I/O ERROR - SKIP T.M. WORK FILE Same as 630, but when trying to skip to the next tape mark.
632	I/O ERROR - FORMAT WORK FILE Same as 628, but when attempting to format a track when a temporary file is used as sort disk work file.
633	I/O ERROR - SKIP TO BACK T.M. WORK FILE Same as 628, but when attempting to reposition a work tape after a restart. The operation of skip to tape mark and backspace is unsuccessful.
634	I/O ERROR - FORWARD SPACE BLOCK WORK FILE Same as 628, but when attempting to reposition a work tape after a restart. A forward space block command terminates abnormally.

In addition to those in Table C-3, certain other messages may be issued in exceptional circumstances. In this case, the organization supporting the installation should be informed. The messages concerned are listed below:

010 SYNTAX ERROR, KEYWORD INVALID IN 'SORT' PARAGRAPH
011 SYNTAX ERROR, KEYWORD INVALID IN 'MERGE' PARAGRAPH
031 SYNTAX ERROR, 'FILES' MISSING
517 INPUT FILE NOT ASSIGNED
524 OUTPUT FILE NOT ASSIGNED
527 SYSIN FILE NOT ASSIGNED
531 INSUFFICIENT WORKTAPES SPECIFIED
537 WORK FILE NOT ASSIGNED
543 INVALID VOLUME OR DEVICE ASSIGNMENT FOR INPUT FILE
544 INVALID VOLUME OR DEVICE ASSIGNMENT FOR OUTPUT FILE
548 LOG FILE NOT ASSIGNED

WARNINGS

Warnings are given when the function requested by the user leads to a situation that SORT judges semi-normal. These cases are:

- empty output files
- invalid records have been detected
- truncation may have occurred (maximum output record size is smaller than maximum input record size)
- end of file has been reached before the HALT value (leads to message number 553)

SECONDARY MESSAGES

Usually the secondary message appears in conjunction with a parameter error diagnostic; however, exceptions exist for messages 561, and 562. See Table C-4.

INVALID RECORDS

For records having an invalid length the following message is displayed:

INVALID RECORD cause : record number

Cause may be INCLUDE/OMIT or KEY/ARRANGE/SUM depending on the unsatisfied criteria. Record number is the record serial number maintained by SORT. SORT counts the records when reading the input file starting at 1 for the first record.

Table C-4. System Error Secondary Messages.

Message No.	Secondary Message Text
561,562	WRK DISK - READ
561,562	WRK DISK - WRITE
561,562	WRK DISK - READ BUF
561,562	WRK DISK - WRITE BUF
562	WRK TAPE - READ
561,562	WRK TAPE - WRITE
561,562	WRK TAPE - WRITE TM
561,562	WRK TAPE - SKIP TM
561,562	WRK TAPE - WRITE ID
562	WRK TAPE - READ ID
562	WRK TAPE - READ BUF
561,562	WRK TAPE - WRITE BUF
562	READ SEMAPHORE LINK
561,562	WRITE SEMAPHORE LINK

★

APPENDIX D

DECLARING WORK SPACE WITH $SORTWORK

The extended JCL statement $SORTWORK is placed inside the step enclosure of a job step, which invokes the SORT facility in a dynamic manner (for example, the SORT verb in COBOL). The purpose of $SORTWORK is to assign to the job step the required work file space.

As with $SORT, there are three different work space possibilities:

1) TAPE BASED: a minimum of three and a maximum of six WORK tapes are specified.

2) DISK BASED: a permanent or temporary disk file on one or more volumes is specified. If a temporary file is used then the size, in cylinders must be declared. This temporary file will always be given the name H_SRTWKD. ★

3) If neither disk nor tape information appears in the $SORTWORK statement, a temporary file of 10 cylinders on a resident volume is allocated, with the name H-SRTWKD.

```
$SORTWORK  { WKTAPE[S] =(NBDV=n,DEVCLASS=device-class-name)
           {
           { WKDISK[S] =( {external-file-name}
           {              {SIZE = cylinders   }
           {
           {              [,FILESTAT = {CAT  } ]
           {                           {UNCAT}
           {
           {              [,CATALOG = n]
           {
           {            [,{RESIDENT                                  }]])
           {              {DEVCLASS= device-class-name                }
           {              { [,MEDIA =({WORK                          })]}
           {              {          {media-name [,...]              }  }
```

Syntax

The parameters and their values are as described in Section II of this manual.

$SORTWORK Example

```
$JOB  CLS, USER=PJJC,PROJECT=LX28;

$COMM  'THE NEXT STATEMENT EXECUTES THE  COBOL
        COMPILER RESULTING IN THE TRANSLATION
        OF THE COBOL PROGRAM CONTAINED IN
        THE INPUT ENCLOSURE SAMPLESORT' ;

$COBOL  SOURCE = *SAMPLESORT  , XREF  ;

$COMM  'THE NEXT STATEMENT EXECUTES THE LINKER
        WHICH, USING THE RESULT OF THE COMPILATION,
        WILL BUILD A LOAD MODULE NAMED SAMPLESORT
        AND STORE IT IN A LIBRARY NAMED PLML'  ;

$LINKER SAMPLESORT, OUTLIB=(PLML,DEVCLASS=MS,MEDIA=BD54);

$COMM  'THE FOLLOWING STEP ENCLOSURE EXECUTES
        THE LOAD MODULE SAMPLESORT. SINCE THE PROGRAM
        SAMPLESORT USES THE SORT VERB, A $SORTWORK
        STATEMENT IS REQUIRED WITHIN THE STEP. NOTE
        ALSO THAT TWO $ASSIGN STATEMENTS ARE PRESENT,
        ONE WITH INTERNAL FILE NAME SAMPLE, THE OTHER
        WITH INTERNAL FILE NAME SORTOUT.'  ;


$STEP       SAMPLESORT,FILE=(PLML,DEVCLASS=MS,MEDIA=BD54);

$ASSIGN     SAMPLE,INVMAST,FILESTAT=UNCAT,DEVCLASS=MT,
            MEDIA=TV46 ;

$ASSIGN     SORTOUT,PWFILE,FILESTAT=UNCAT,DEVCLASS=MS,
            MEDIA=BD22;

$SORTWORK   WKDISK =(PRISRTWK,DEVCLASS=MS,MEDIA=BD14);

$ENDSTEP;

$COMM       'THERE NOW FOLLOWS THE INPUT ENCLOSURE SAMPLESORT
            WHICH CONTAINS THE COBOL SOURCE PROGRAM'  ;

$INPUT      SAMPLESORT , TYPE = COBOL  ;

          IDENTIFICATION DIVISION.
            PROGRAM-ID.  SAMPLESORT.
          ENVIRONMENT DIVISION.
```

APPENDIX E

COLLATING SEQUENCES

The tables below show the collating sequences available with the sort utility. The tables show the characters in ascending order, from top to bottom, and left to right. They begin with the "low-value" and finish with the "high-value".

Table E-1. EBCDIC (Series 60 Level 64) Character Set

Hexa-decimal	Graphic	H36 punch	Hexa-decimal	Graphic	H36 punch
00	low-val	12-0-1-8-9	20		11-0-1-8-9
01		12-1-9	21		0-1-9
02		12-2-9	22		0-2-9
03		12-3-9	23		0-3-9
04		12-4-9	24		0-4-9
05		12-5-9	25		0-5-9
06		12-6-9	26		0-6-9
07		12-7-9	27		0-7-9
08		12-8-9	28		0-8-9
09		12-1-8-9	29		0-1-8-9
0A		12-2-8-9	2A		0-2-8-9
0B		12-3-8-9	2B		0-3-8-9
0C		12-4-8-9	2C		0-4-8-9
0D		12-5-8-9	2D		0-5-8-9
0E		12-6-8-9	2E		0-6-8-9
0F		12-7-8-9	2F		0-7-8-9
10		12-11-1-8-9	30		12-11-0-1-8-9
11		11-1-9	31		1-9
12		11-2-9	32		2-9
13		11-3-9	33		3-9
14		11-4-9	34		4-9
15		11-5-9	35		5-9
16		11-6-9	36		6-9
17		11-7-9	37		7-9
18		11-8-9	38		8-9
19		11-1-8-9	39		1-8-9
1A		11-2-8-9	3A		2-8-9
1B		11-3-8-9	3B		3-8-9
1C		11-4-8-9	3C		3-8-9
1D		11-5-8-9	3D		5-8-9
1E		11-6-8-9	3E		6-8-9
1F		11-7-8-9	3F		7-8-9

Table E-1 (cont). EBCDIC (Series 60 Level 64) Character Set

Hexa-decimal	Graphic	H36 punch
40	space	no punches
41		12-0-1-9
42		12-0-2
43		12-0-3-9
44		12-0-4-9
45		12-0-5-9
46		12-0-6-9
47		12-0-7-9
48		12-0-8-9
49		12-1-8
4A		12-2-8
4B	.	12-3-8
4C	<	12-4-8
4D	(	12-5-8
4E	+	12-6-8
4F	¦	12-7-8
50	&	
51		12-11-1-9
52		12-11-2-9
53		12-11-3-9
54		12-11-4-9
55		12-11-5-9
56		12-11-6-9
57		12-11-7-9
58		12-11-8-9
59		11-1-8
5A	!	11-2-8
5B	$	11-3-8
5C	*	11-4-8
5D	)	11-5-8
5E	;	11-6-8
5F		11-7-8

Hexa-decimal	Graphic	H36 punch
60	-	11
61	/	0-1
62		11-0-2-9
63		11-0-3-9
64		11-0-4-9
65		11-0-5-9
66		11-0-6-9
67		11-0-7-9
68		11-0-8-9
69		0-1-8
6A	¦	12-11
6B	,	0-3-8
6C	%	0-4-8
6D	_	0-5-8
6E	>	0-6-8
6F	?	0-7-8
70		12-11-0
71		12-11-0-1-9
72		12-11-0-2-9
73		12-11-0-3-9
74		12-11-0-4-9
75		12-11-0-5-9
76		12-11-0-6-9
77		12-11-0-7-9
78		12-11-0-8-9
79	`	1-8
7A	:	2-8
7B	#	3-8
7C	@	4-8
7D	'	5-8
7E	=	6-8
7F	"	7-8

Table E-1 (cont). EBCDIC (Series 60 Level 64) Character Set

Hexa-decimal	Graphic	H36 punch
80		12-0-1-8
81	a	12-0-1
82	b	12-0-2
83	c	12-0-3
84	d	12-0-4
85	e	12-0-5
86	f	12-0-6
87	g	12-0-7
88	h	12-0-8
89	i	12-0-9
8A		12-0-2-8
8B		12-0-3-8
8C		12-0-4-8
8D		12-0-5-8
8E		12-0-6-8
8F		12-0-7-8
90		12-11-1-8
91	j	12-11-1
92	k	12-11-2
93	l	12-11-3
94	m	12-11-4
95	n	12-11-5
96	o	12-11-6
97	p	12-11-7
98	q	12-11-8
99	r	12-11-9
9A		12-11-2-8
9B		12-11-3-8
9C		12-11-4-8
9D		12-11-5-8
9E		12-11-6-8
9F		12-11-7-8

Hexa-decimal	Graphic	H36 punch
A0		11-0-1-8
A1	~	11-0-1
A2	s	11-0-2
A3	t	11-0-3
A4	u	11-0-4
A5	v	11-0-5
A6	w	11-0-6
A7	x	11-0-7
A8	y	11-0-8
A9	z	11-0-9
AA		11-0-2-8
AB		11-0-3-8
AC		11-0-4-8
AD		11-0-5-8
AE		11-0-6-8
AF		11-0-7-8
B0		12-11-0-1-8
B1		12-11-0-1
B2		12-11-0-2
B3		12-11-0-3
B4		12-11-0-4
B5		12-11-0-5
B6		12-11-0-6
B7		12-11-0-7
B8		12-11-0-8
B9		12-11-0-9
BA		12-11-0-2-8
BB		12-11-0-3-8
BC		12-11-0-4-8
BD		12-11-0-5-8
BE		12-11-0-6-8
BF		12-11-0-7-8

Table E-1(cont). EBCDIC (Series 60 Level 64) Character Set

Hexa-decimal	Graphic	H36 punch
C0	{	12-0
C1	A	12-1
C2	B	12-2
C3	C	12-3
C4	D	12-4
C5	E	12-5
C6	F	12-6
C7	G	12-7
C8	H	12-8
C9	I	12-9
CA		12-0-2-8-9
CB		12-0-3-8-9
CC		12-0-4-8-9
CD		12-0-5-8-9
CE		12-0-6-8-9
CF		12-0-7-8-9
D0	}	11-0
D1	J	11-1
D2	K	11-2
D3	L	11-3
D4	M	11-4
D5	N	11-5
D6	O	11-6
D7	P	11-7
D8	Q	11-8
D9	R	11-9
DA		12-11-2-8-9
DB		12-11-3-8-9
DC		12-11-4-8-9
DD		12-11-5-8-9
DE		12-11-6-8-9
DF		12-11-7-8-9

Hexa-decimal	Graphic	H36 punch
E0	\	0-2-8
E1		11-0-1-9
E2	S	0-2
E3	T	0-3
E4	U	0-4
E5	V	0-5
E6	W	0-6
E7	X	0-7
E8	Y	0-8
E9	Z	0-9
EA		11-0-2-8-9
EB		11-0-3-8-9
EC		11-0-4-8-9
ED		11-0-5-8-9
EE		11-0-6-8-9
EF		11-0-7-8-9
F0	0	0
F1	1	1
F2	2	2
F3	3	3
F4	4	4
F5	5	5
F6	6	6
F7	7	7
F8	8	8
F9	9	9
FA		12-11-0-2-8-9
FB		12-11-0-3-8-9
FC		12-11-0-4-8-9
FD		12-11-0-5-8-9
FE		12-11-0-6-8-9
FF		12-11-0-7-8-9

Table E-2 below shows the collating sequence for the Series 100 character set. The "Hexadecimal" column gives the original internal value of the character in a Series 100 system. The rightmost column shows the hexadecimal value which the character has after translation into the Series 60 Level 64 environment.

Table E-2. Series 100 Character Set

Hexadecimal pre-translation	Graphic	H14 punch	H36 punch	EBCDIC
00	low-val			00
40	0	0	0	F0
41	1	1	1	F1
42	2	2	2	F2
43	3	3	3	F3
44	4	4	4	F4
45	5	5	5	F5
46	6	6	6	F6
47	7	7	7	F7
48	8	8	8	F8
49	9	9	9	F9
4A	[	8-2	12-8-2	4A
4B	#	8-3	8-3	7B
4C	@	8-4	8-4	7C
4D	:	8-5	8-2	7A
4E	>	8-6	0-8-6	6E
4F	?	8-7	0-8-7	6F
50	space	no punches	no punches	40
51	A	12-1	12-1	C1
52	B	12-2	12-2	C2
53	C	12-3	12-3	C3
54	D	12-4	12-4	C4
55	E	12-5	12-5	C5
56	F	12-6	12-6	C6
57	G	12-7	12-7	C7
58	H	12-8	12-8	C8
59	I	12-9	12-9	C9
5A	&	12	12	50
5B	.	12-8-3	12-8-3	4B
5C	]	12-8-4	11-8-2	4F
5D	(	12-8-5	12-8-5	4D
5E	<	12-8-6	12-8-4	4C
5F	\	12-8-7	0-8-2	E0

Table E-2(cont). Series 100 Character Set

Hexadecimal pre-translation	Graphic	H14 punch	H36 punch	EBCDIC
A0		11-0	11-8-7	5F
A1	J	11-1	11-1	D1
A2	K	11-2	11-2	D2
A3	L	11-3	11-3	D3
A4	M	11-4	11-4	D4
A5	N	11-5	11-5	D5
A6	O	11-6	11-6	D6
A7	P	11-7	11-7	D7
A8	Q	11-8	11-8	D8
A9	R	11-9	11-9	D9
AA	-	11	11	60
AB	$	11-8-3	11-8-3	5B
AC	*	11-8-4	11-8-4	5C
AD	)	11-8-5	11-8-5	5D
AE	;	11-8-6	11-8-6	5E
AF	'	11-8-7	8-5	7D
B0	+	12-0	12-8-6	4E
B1	/	0-1	0-1	61
B2	S	0-2	0-2	E2
B3	T	0-3	0-3	E3
B4	U	0-4	0-4	E4
B5	V	0-5	0-5	E5
B6	W	0-6	0-6	E6
B7	X	0-7	0-7	E7
B8	Y	0-8	0-8	E8
B9	Z	0-9	0-9	E9
BA	_	0-8-2	0-8-5	6D
BB	,	0-8-3	0-8-3	6B
BC	%	0-8-4	0-8-4	7C
BD	=	0-8-5	8-6	7E
BE	"	0-8-6	8-7	7F
BF	!	0-8-7	11-8-2	5A
FF	hi-val			FF

NOTE: When two entries appear in the graphic column in Table E-2, the first graphic is the original series 100 character and the second is the graphic value after translation into EBCDIC.

Table E-3 below shows the effect of COLLATE=H200 on a Series 200/2000 disk or tape file. The octal values are those which exist in a Series 200/0 system.

Table E-3. H200 Collating Sequence for Series 200/0 Files

Graphic	Octal	Hex	Graphic	Octal	Hex
0	00	00	-	40	20
1	01	01	J	41	21
2	02	02	K	42	22
3	03	03	L	43	23
4	04	04	M	44	24
5	05	05	N	45	25
6	06	06	O	46	26
7	07	07	P	47	27
8	10	08	Q	50	28
9	11	09	R	51	29
`	12	0A	#	52	2A
=	13	0B	$	53	2B
:	14	0C	*	54	2C
Blank	15	0D	"	55	2D
>	16	0E		56	2E
&	17	0F	!	57	2F
+	20	10	<	60	30
A	21	11	/	61	31
B	22	12	S	62	32
C	23	13	T	63	33
D	24	14	U	64	34
E	25	15	V	65	35
F	26	16	W	66	36
G	27	17	X	67	37
H	30	18	Y	70	38
I	31	19	Z	71	39
;	32	1A	@	72	3A
.	33	1B	,	73	3B
)	34	1C	(	74	3C
%	35	1D		75	3D
	36	1E		76	3E
?	37	1F		77	3F

Table E-4 shows the effect of COLLATE = H200 on EBCDIC files.

When two graphics are present (special characters), the first is the Series 200 value and the second is the equivalent in Series 60 Level 64.

Table E-4. H200 Collating Sequence for EBCDIC Files

Graphic	Hexa-decimal	Graphic	Hexa-decimal	Graphic	Hexa-decimal	Graphic	Hexa-decimal
0	F0	A a	C1 81	J j	D1 91	/	61
1	F1	B b	C2 82	K k	D2 92	S s	E2 A2
2	F2	C c	C3 83	L l	D3 93	T t	E3 A3
3	F3	D d	C4 84	M m	D4 94	U u	E4 A4
4	F4	E e	C5 85	N n	D5 95	V v	E5 A5
5	F5	F f	C6 86	O o	D6 96	W w	E6 A6
6	F6	G g	C7 87	P p	D7 97	X x	E7 A7
7	F7	H h	C8 88	Q q	D8 98	Y y	E8 A8
8	F8	I i	C9 89	R r	D9 99	Z z	E9 A9
9	F9	:	5E	#	7B	@	7C
`	7D	.	4B	$	5B	,	6B
=	7E	)	5D	*	5C	(	4D
;	7A	%	6C	"	7E	;	4F
space	40		5F		E0	_	6D
>	6E	?	6F	!	5A		4A
&	50	-)	60 D0	<	4C		
+ (	4E C0						

Note: all other hexadecimal values have the same collating sequence value as hexadecimal 5F (graphic). Double entries in the table such as "A a C1 81", indicate that the values are equivalent for collating purposes.

Table E-5 shows the effect of COLLATE=ASCII for an EBCDIC file.

Table E-5. ASCII Collating Sequence

Graphic	Hexa-decimal	Graphic	Hexa-decimal	Graphic	Hexa-decimal	Graphic	Hexa-decimal
NUL	00	space	40	@	7C		79
SOH	01	!	4F	A	C1	a	81
STX	02	"	7F	B	C2	b	82
ETX	03	#	7B	C	C3	c	83
EOT	37	$	5B	D	C4	d	84
ENQ	2D	%	6C	E	C5	e	85
ACK	2E	&	50	F	C6	f	86
BEL	2F	'	7D	G	C7	g	87
BS	16	(	4D	H	C8	h	88
HT	05	)	5D	I	C9	i	89
LF	25	*	5C	J	D1	j	91
VT	0B	+	4E	K	D2	k	92
FF	0C	,	6B	L	D3	l	93
CR	0D	-	60	M	D4	m	94
SO	0E	.	4B	N	D5	n	95
SI	0F	/	61	O	D6	o	96
DLE	10	0	F0	P	D7	p	97
DC1	11	1	F1	Q	D8	q	98
DC2	12	2	F2	R	D9	r	99
TM	13	3	F3	S	E2	s	A2
DC4	3C	4	F4	T	E3	t	A3
NAK	3D	5	F5	U	E4	u	A4
SYN	32	6	F6	V	E5	v	A5
ETB	26	7	F7	W	E6	w	A6
CAN	18	8	F8	X	E7	x	A7
EM	19	9	F9	Y	E8	y	A8
SUB	3F	:	7A	Z	E9	z	A9
ESC	27	;	5E		4A	{	C0
IFS	1C	<	4C	\	E0	:	6A
IGS	1D	=	7E	!	5A	}	D0
IRS	1E	>	6E		5F	~	A1
IUS	1F	?	6F	-	6D	DEL	07
DS	20		41		76		B8
SOS	21		42		77		B9
FS	22		43		78		BA
	23		44		80		BB
BYP	24		45		8A		BC
NL	15		46		8B		BD
LC	06		47		8C		BE
IL	17		48		8D		BF

Table E-5(cont). ASCII Collating Sequence

Graphic	Hexa-decimal	Graphic	Hexa-decimal	Graphic	Hexa-decimal	Graphic	Hexa-decimal
	28		49		8E		CA
	29		51		8F		CB
SM	2A		52		90		CC
CU2	2B		53		9A		CD
	2C		54		9B		CE
RLF	09		55		9C		CF
SMM	0A		56		9D		DA
CU1	1B		57		9E		DB
	30		58		9F		DC
	31		59		A0		DD
CC	1A		62		AA		DE
	33		63		AB		DF
PN	34		64		AC		EA
RS	35		65		AD		EB
UC	36		66		AE		EC
	08		67		AF		ED
	38		68		B0		EE
	39		69		B1		EF
	3A		70		B2	LVM	FA
CU3	3B		71		B3		FB
RF	04		72		B4		FC
RES	14		73		B5		FD
	3E		74		B6		FE
	E1		75		B7		FF

APPENDIX F

PERFORMANCE IMPROVEMENT

The user wanting to get the best possible performance from a sort application should note the following:

- Using Checkpoint facilities has a significant impact on elapsed time. Performance degradation can be roughly evaluated between 15% and 30%.

- File placement : avoid placing the input and output files on the same pack as the disk work files.

- Give the SORTSIZE parameter in the DSL. An exact value will produce a well tailored sort. In case of doubt it is better to supply a value that is too big than none.

- Sort performances are improved if more memory is supplied for the execution, so the SIZE keyword is of prime importance. A default value of 53 K bytes is allocated if the keyword is absent from $SORT. The upper and lower values are explained hereunder:

  - lower value : 22K + MAX (input file buffers, output file buffers)

    input file buffers = total size allocated for buffers to process the input file. If B represents the input file block size then the total input file buffers size is computed as:

    $$DATABUF * B * BPB$$

    If standard values for number of buffers and number of blocks per buffer are used then the input file buffers size is:

    $$2 * B$$

    output file buffers = same computation as for input file buffers.

For example : Suppose the input and output files have the same block size of 4K. When using standard options for input and output file processing, the minimum value for SIZE is 30. If you use single buffering for input and output, you can reduce this value to 26.

- upper value = 512 bytes

The BLKSIZE keyword should be specified in the OUTFILE description when the output file and the input file have not the same block size. The keyword specification serves two purposes:

- if the input block size is larger than the output block size, then performances will be better when specifying BLKSIZE in the OUTFILE.

- if the output block size is larger than the input block size, then if the BLKSIZE value is not specified, SORT may exceed the specified working set.

There is no need to specify this keyword if the input file and the output file have the same block size.

The REPORT option has an influence on performance.

The selection of PARAMS or ALL will slightly degrade performance if the value of SIZE is high, but degradation can be more important for small working sets. PARAMS or ALL should be used for debugging the sort application. Once debugged the AUDIT option (default JCL value) provides enough information.

APPENDIX G

SORTING CARD FILES AND SUBFILES

CARD FILES

- The card file can be transferred to a sequential disk file, which can then be sorted in the normal manner
- The SYSIN facility can be used
- The cards can be directly input to the SORT.

Using SYSIN

The card deck to be sorted is placed between $INPUT and $ENDINPUT cards. The JCL $SORT statement refers to the input file thus:

```
... ,INFILE = (input-enclosure-name,FILESTAT = IN), ...
```

This statement will produce a warning message at JCL translation time but no corrective action is necessary. The input enclosure will be written in SARF format.

Using Direct Input

The card deck to be sorted is prepared with a $EOS card at the end. In the $SORT statement, INFILE describes a card file, with RECSIZE and BLKSIZE (at least) specified. Records may be one per card (REFORM = F) or several per card (RECFORM = FB). Blocksize should be either 80 or 51, depending on the type of cards used. The DEVCLASS keyword must specify a card reader (i.e. CD/R/C80/MB).
51/MI

Cards are read in SARF format and the punch code must be H36. (Since the cards are read in SARF format, command cards such as $HOL or

$BIN are not allowed in the stream). At the beginning of the sort, a message is sent to the operator to mount the deck of cards named in the input-enclosure-name of INFILE on the card reader.

The input-enclosure-name is purely for operator identification. SORT cannot check the identity of the deck. When the operator switches the card reader from standby to ready, the sort starts to read the cards, stopping at the $EOS card.

Since blocks are very small (80 or 51) the reading speed can be increased by specifying BPB = 5 in the INFILE description. That will result in the reading of 5 cards at each I/O operation. Using the default value of 2 for buffers (DATABUF keyword) will give a reading speed close to the maximum device speed.

SUBFILE SORT

The sorting of subfiles is treated in exactly the same way as sequential files (except that the SUBFILE keyword must be present), provided that the subfile is in SARF format. Sort always assumes that data input to it is in SARF format. Care must be taken, since subfiles are generally library members, so:

- a special control record exists at the beginning of the subfile
- the data in the record is preceded by an SSF header of 8 bytes

It is then recommended to first convert the subfile from SSF to SARF before sorting it.
Otherwise the control record will be input to the sort (unless omitted by OMIT) and the other records processed with their SSF headers.

APPENDIX H

SORTING RECORDS OVER 1K BYTES IN LENGTH

The input records to the sort utility have the following size limitations:

- When work files are on tapes, maximum record size is 16000 bytes.
- When work files are on MSU0310, maximum record size is 7000 bytes
- When work files are on MSU0350 or MSU0400, maximum record size is 13000 bytes.

When sorting large records, care must be taken to supply SORT with sufficient memory. Insufficient memory allocation results in message number 581 being issued. The minimum value for SIZE given in Section II is

22K + max (input buffers, output buffers).

This is valid for records up to 1K bytes. When the records are larger, the minimum value for SIZE in disk sorts is

14 + 7 * RECSIZE + MAX(input buffers, output buffers).

For example, a sort of records of 5K bytes on a file which has a blocking factor of 1, where double buffering is used on input and output requires a minimum value of

14 + 7 + 10 = 59 K bytes for SIZE.

Clearly, 59K will not yield the best performance for the sort, but it is adequate.

For tape sorts, calculate the minimum value of SIZE as follows:

14 + (NBDV + 4) * RECSIZE + MAX(input buffers, output buffers)

where NBDV is the number of work tapes.

APPENDIX I

CHECKPOINT/RESTART IN SORT

If the REPEAT keyword is specified in the $SORT JCL statement, the checkpoint mechanism is selected. This enables the SORT to be restarted after an abnormal termination from which recovery is possible (I/O errors). When such an error is detected by the SORT, the process is aborted with a SEV3 status. (Sometimes, however, when reading the input file or writing the output file, sort may terminate with SEV4, even when the abnormal condition is an I/O error.) The following message is output on the operator console:

AV18 ddnn I/O ERROR ON volume-name FOR ron

The operator can then decide whether or not to change the device of the involved volume and restart the sort from the last checkpoint.

If the abnormal condition is a system crash, the sort process can also be restarted from the last checkpoint.

Note that no checkpoint is taken during the reading of the input file or the creation of the output file.

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE: SERIES 60 (LEVEL 64) SORT/MERGE

ORDER NO.: AQ85, REV. 1

DATED: SEPTEMBER 1978

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here and furnish complete mailing address below. ☐

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

FIRST CLASS
PERMIT NO. 39531
WALTHAM, MA
02154

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

21612, 1.5878, Printed in U.S.A.

AQ85, Rev. 1