

NOTE ...

this manual was written a while ago, but most of the information is (probably) still relevant

my current address, affiliation, and contact information are as follows ...

Mike Zdeb

Assistant Professor

U@Albany School of Public Health

1 University Place

Rensselaer, NY 12144

518.402.4479

msz03@albany.edu

**A GUIDE TO SyncSort
FOR DATA ANALYSTS AND RESEARCHERS**

Mike Zdeb

New York State Department of Health
Information Systems and Health Statistics
Concourse - Room C-144
Albany, NY 12237-0044

(518)474-2377

May 1996

NOTE This guide assumes a few prerequisites. You must know how to edit a file on VM. You must know how to submit a job from VM to the MVS operating system. You must understand certain concepts about files, records, and fields within records. If any or all of these terms are new to you, the material contained in this guide may be difficult to understand.

Even if you have the prerequisites, some or all of the material concerning SyncSort may be new to you. If you do not understand something in the introductory sections of the guide, DO NOT DESPAIR. There are a number of examples in later sections that show you how to use SyncSort to accomplish tasks that are common to the work of most data analysts or researchers. Also, a number of examples are available from current SyncSort users that might help in your understanding of how SyncSort can assist in your work.

TABLE OF CONTENTS

Section	Page
1.0	INTRODUCTION 1
1.1	SyncSort Statements 1
1.2	Using SyncSort Statements 2
1.2.1	Selecting Records 2
1.2.2	Selecting Fields from Records 3
1.2.3	Sorting / Copying Records 4
1.2.4	Formatting Output from SORT 5
1.2.5	Combining Records 7
2.0	SyncSort ON MVS 8
2.1	Copying Records 8
2.1.1	Tape-to-Tape 8
2.1.1a	Entire contents of a tape containing only one file 8
2.1.1b	Entire contents of a tape containing multiple files 9
2.1.1c	Selected records from one tape 10
2.1.1d	Selected records from several tapes 10
2.1.1e	Selected records from two tapes with fixed length, but with different record lengths 11
2.1.2	Tape-to-VM via the Reader 11
2.1.2a	Selected portions of selected records 11
2.1.2b	Multiple output files 12
2.2	Counting Records 14
2.2.1	Counts Output to a Single File in the VM Reader 14
2.2.2	Counts Output to a Temporary Disk File on MVS - Used as Input to a SAS Job 15
2.2.3	Multiple Output Files Used to Produce Two Sets of Counts - Sent to VM Reader as One File 15
3.0	SyncSort ON VM 16
3.1	Starting SyncSort on VM 16
3.2	SyncSort Commands Specific to VM 16
3.3	Passing File Information to SyncSort on VM 17
3.3.1	Single Input File and Single Output File 17
3.3.1a	Direct entry on the command line 17
3.3.1b	Using FILEDEFS 17
3.3.1c	Combination of direct entry and FILEDEFS 18
3.3.2	Multiple Input and/or Output Files 18
3.3.2a	Direct entry on the command line - multiple input, single output 18
3.3.2b	Using FILEDEFS - multiple input, single output 18
3.3.2c	Combination of direct entry and FILEDEFS - single input, multiple output 18

Appendices

A	SyncSort References	19
B	The Flow of SyncSort	20
C	Allocating Temporary Disk Space on MVS	21
	BASICS	21
	EXAMPLE	22
D	Examples of Uses on SyncSort on MVS	23
	Example 1 Extract and Copy Records - Simple Record Selection	23
	Example 2 Extract and Copy Records - Complex Record Selection	24
	Example 3 Extract and Count Records	24
	Example 4 Extract and Count Records - Use Output from SyncSort as SAS Input	25
	Example 5 Extract and Count Records when Multiple Fields on a Record Contain Data that are to be Combined into One Count	26
	Example 6 Extract and Count Records Using Two Fields to Accumulate Counts - Use Counts as Input for SAS	28
E	Examples of Uses of SyncSort on VM	30
	SORT RECORDS	30
	EXTRACT RECORDS	30
	COUNT RECORDS	30
	EXTRACT DATA	31
	COMBINE SPECIFIED RECORDS FROM SEVERAL FILES	31
	SORT A FILE ON SEVERAL FIELDS	31
	ELIMINATE RECORDS WITH DUPLICATE DATA IN SPECIFIED POSITIONS	32
	PRODUCE SEVERAL OUTPUT FILES CONTAINING SPECIFIED RECORDS	32
F	Using ETHERNET to Send SyncSort Output from MVS Directly to a PC	33
G	Some Other Useful SyncSort Statement	35

1.0 INTRODUCTION

SyncSort is a utility available on both MVS and VM that can be used to copy, sort, merge, or count records. Many of the tasks performed by SyncSort can also be performed using other software. However, SyncSort has a distinct advantage over other software in that it performs tasks very fast, and requires very few statements to perform tasks.

SyncSort also has disadvantages. When compared to other software packages, e.g. SAS, that perform tasks similar to those performed by SyncSort, the SyncSort command set can appear limited and cryptic. Also, the reference manuals for SyncSort on both the VM and MVS operating systems are not written for the non-programmer. The terminology used can be intimidating, and it is difficult to determine what you have to know and what you can ignore.

This guide is intended to explain SyncSort to data analysts and researchers, i.e., those whose main task is not programming. It does not cover all the tasks performed by SyncSort. Rather, it explains some of the features of SyncSort that have been found useful in processing departmental data on both VM and MVS. Other information contained in this guide includes JCL necessary to use SyncSort on MVS; examples of how to use SyncSort to extract and/or count records; examples of using SyncSort to pre-process data prior to using SAS for further data analysis or reporting.

A complete description of SyncSort capabilities can be found in the *Programmer's Guides* (see appendix A). Portions of this guide are based upon material found in the *SyncSort OS Applications Workshop*, a publication that also goes beyond the reference manuals in trying to explain SyncSort to non-programmers.

1.1 SyncSort Statements

The control statements (sometimes referred to later in this guide as the command set) used to process records with SyncSort are as follows:

INCLUDE/OMIT	select records for processing
INREC	select fields to include from records to be processed
SORT	sort records by specified fields (also referred to as sort keys), or copy records
OUTREC	position and format data on output records
OUTFIL	an alternative to OUTREC (allows for multiple file output and for editing of output records)
SUM	combines records that contain equal sort keys, or delete records with equal sort keys

These statements are used in conjunction with the following logical expressions and relative conditions to select records for processing:

logical expressions	AND	OR
relative conditions	EQ (equal)	NE (not equal)
	GT (greater than)	GE (greater than or equal)
	LT (less than)	LE (less than or equal)

When constructing SyncSort statements to select records for processing, the following rules apply:

- conditions within internal parentheses are evaluated first
- AND conditions are evaluated next
- OR conditions are evaluated last

1.2 Using SyncSort Statements

A SyncSort job consists of combinations of the control statements, logical expressions, and relative conditions shown in section 1.1. The following sections describe how to use the SyncSort command set. In using these commands, it is important to know the flow of a SyncSort job, i.e. the order in which SyncSort processes commands. The illustration in appendix B can be used in conjunction with this section to understand how SyncSort processes data.

1.2.1 Selecting Records

The INCLUDE/OMIT control statements allows the specification of conditions for the selection or omission of records for processing with SyncSort. The following rules apply:

- a field may be compared to a constant or another field within a record
- multiple editing conditions may be formed by using ANDs, ORs, and parentheses as needed
- only ONE INCLUDE/OMIT statement is allowed per sort

The form of the statement is

INCLUDE(or OMIT) COND=(logical expression)[,FORMAT=f]

where the logical expression uses the format

relative condition [(,AND(or OR),) relative condition ...]

and the relative condition(s) are expressed as

field ,EQ(or NE GT GE LT LE), constant(or field)

For example:

- INCLUDE COND=(5,2,CH,EQ,C'70')
accepts all records where positions 5-6 contain the literal string 70
- INCLUDE COND=(5,2,GE,C'01',AND,5,2,LE,C'61'),FORMAT=CH
accepts all records where positions 5-6 contain a literal string in the range 01 to 61
- OMIT COND=(1,2,EQ,5,2),FORMAT=CH
deletes all records where the literal strings in positions 1-2 and 5-6 are equal
- OMIT COND=(10,1,ZD,EQ,1,OR,(20,2,CH,GT,C'09',AND,20,2,CH,LT,C'50'))
deletes all records where position 10 contains a numeric entry 1, or where positions 20-21 contain a literal string in the range 10-49

NOTE: The above statements illustrate that SyncSort uses 'positional notation' to refer to location of data within records, i.e. data are referred to by the starting position of the data followed by the length of the field containing the data.

NOTE: The INCLUDE/OMIT statement contributes to the EFFICIENCY of a SyncSort job by reducing the number of records to be processed by subsequent SyncSort statements. More efficient jobs use fewer system resources (disk space and computer time). As will be shown later, records can also be included in an output section. However, record selection on output does not contribute to efficiency.

Several different data formats are addressed in these examples. They are not exhaustive of the formats that SyncSort can handle. However, they should suffice for most data dealt with by Department data analysts and researchers. The CH data format in examples 'a' through 'd' is used for CHARACTER data. This format is used when data are to be treated as literal strings. The format ZD in example 'd' is used for zoned-decimal data. Though this term may not be familiar, it can be thought of as the format to use when data are to be treated as numeric.

NOTE: When record positions are referred to in any of the examples, a RECORDTYPE of FIXED is assumed. If the RECORDTYPE is VARIABLE, SyncSort considers the 4-byte header (the place where information as to record length is stored) as the first 4 record positions. Thus, with a variable length records, data that appears to be in position 1 would have to be described to SyncSort as in position 5, with every subsequent data element moved over 4 positions. To AVOID CONFUSION, it is easier to work with FIXED LENGTH RECORDS.

As noted in the examples, there are two types of comparisons that can be made in the edit statements: FIELD-to-FIELD, FIELD-to-CONSTANT. The following types of comparisons are allowed.

FIELD-to-FIELD

between two fields with the same format

FIELD-to-CONSTANT

between ZD and a numeric constant
between CH and a character literal

In those cases where the length of the fields or constants being compared differ, the following rules apply:

FIELD-to-FIELD

the shorter field is ZERO-padded in its own format
CH padded on the right ZD padded on the left

FIELD-to-CONSTANT

the constant length is adjusted
numeric constants are padded or truncated on the left - padding with ZERO if the field is ZD, or with NULLS in all other cases character literals are padded or truncated on the right - padding with SPACES

NOTE: The SAFEST way to make comparisons is to try to KEEP FIELDS and CONSTANTS the SAME LENGTH, thus AVOIDING any PADDING or TRUNCATION.

1.2.2 Selecting Fields from Records

INREC allows selection of fields from records. The form of the statement is

INREC FIELDS=(field,<field,field,...>)

The following general rules apply:

- input fields are defined by position,length (as in examples 'a' through 'e')
- data may be repositioned with column pointers, by inserting spaces or literals (as in examples 'b' through 'e')

For example:

- a. INREC FIELDS=(1,2,100,50)
inputs data from positions 1 and 2, and 100 through 149
all data from the input file in positions 1-2 remain in those positions, while data from positions 100-149 now reside in positions 3-52
- b. INREC FIELDS=(10,5,8:20,3)
inputs data from positions 10 through 14, and 20 through 22
data from positions 10-14 now reside in positions 1-4, while data from positions 20-22 have been placed in positions 8-10 by using the column pointer 8:
- c. INREC FIELDS=(40,9,C'NYS')
inputs data from positions 40 through 48
these data now reside in positions 1-9, and the literal NYS resides in positions 10-12
- d. INREC FIELDS=(5,2,3X,7,2)
inputs data from positions 5 and 6, and 7 and 8
data from positions 5-6 now reside in positions 1-2, while data from positions 7-8 have been placed in positions 6-7 by using the 3X to insert three SPACES
- e. INREC FIELDS=(135,2,8,4,C'000001')
inputs data from positions 135 and 136, and 8 through 11
data from positions 135 and 136 now reside in positions 1-2, while data from positions 8 through 11 have been placed in positions 3 through 6, while the literal string 000001 has been added to each record

NOTE: The INREC statement REPOSITIONS data from the input record. Since the INREC statement REFORMATS the input records, any SyncSort control statements (e.g. SORT) subsequent to INREC must take into account the new positions of the input data.

NOTE: The INREC statement contributes to the EFFICIENCY of a SyncSort job. INREC allows a user to reduce the size of the records that will be processed by subsequent steps in a SyncSort job. Reducing the size of records will cut down on the system resources used by the job. Therefore, if only certain fields from the input records are of interest, an INREC statement should be used.

1.2.3 Sorting / Copying Records

The SORT command can be used to sort, copy, or merge records (merge not covered in this guide). Records are sorted after INREC and INCLUDE/OMIT steps are executed. Thus, if an INREC statement is used, the SORT statement must take into account any reformatting of records that may have resulted from INREC processing.

The form of the SORT command is

```
SORT FIELDS=(field[,field,...]) [,FORMAT=f]
```

The fields used in the sort are expressed as:

Position,Length,Format,Collating Sequence (as in examples 'a' and 'b')

-or-

Position,Length,Collating Sequence (as in example 'c' --- where the format of all data in the edit is the same and the format is placed after the closing parenthesis)

NOTE: Collating sequences are either 'A' for ascending order, or 'D' for descending order.

For example:

- a. SORT FIELDS=(5,2,CH,A)
sorts the input file on positions 5 and 6, containing character data, in ascending order
- b. SORT FIELDS=(1,5,CH,A,10,2,ZD,D)
sorts the input file on positions 1 through 5, containing character data, in ascending order, and positions 10 and 11, containing zoned-decimal data, in descending order within
- c. SORT FIELDS=(10,2,A,30,2,A),FORMAT=CH
sorts the input file on positions 10 and 11, in ascending order, and positions 30 and 31, in ascending order, where both fields contain character data

NOTE: Up to sixty-four fields can be specified.

NOTE: When data are sorted on more than one field, the first sort is referred to as the PRIMARY sort, and the remaining sorts are referred to as SECONDARY sorts. SECONDARY sorts will sort data within like data of the PRIMARY sort

There are several options that can be used with the SORT statement. Consult the SyncSort references referred to previously for information on these options.

The SORT command can also be used to copy records, as in:

```
SORT FIELDS=COPY
```

No sort fields are used when the SORT command is used to copy records. No rearranging of records takes place, i.e. records are copied in the order they appear in the input file.

1.2.4 Formatting Output from SORT

Output from SORT is formatted using either an OUTREC statement (one output file) or a series of OUTFIL statements (multiple output files). Either OUTREC or OUTFIL can be used to:

- select output fields
- reposition data
- insert literal fields or spaces

As with the SORT, OUTREC and OUTFIL process records after INREC and INCLUDE/OMIT. Therefore, any reformatting of data performed with INREC must be taken into account when using OUTREC and OUTFIL.

The form of OUTREC is similar to that of the INREC statement

```
OUTREC FIELDS=(field,<field,field,...>)
```

The following general rules apply:

- output fields are defined by position,length (as in examples 'a' through 'd')
- data may be repositioned with column pointers, by inserting spaces or literals (as in examples 'b' through 'd')

For example:

- a. OUTREC FIELDS=(1,30)
outputs data from the first 30 positions of the records that result from processing prior to the SORT command
- b. OUTREC FIELDS=(11,2,21,2,31,10)
outputs data from positions 11 and 12, 21 and 22, and 31 through 40 of the records that result from processing prior to the SORT command
- c. OUTREC FIELDS=(1,5,7:C'00001')
outputs data from positions 1 through 5 of the records that result from processing prior to the SORT command, and adds the literal string 00001 to each record starting in column seven, using the column pointer 7:
- d. OUTREC FIELDS=(1,5,2X,6,5)
outputs data from positions 1 through 5, and 6 through 10 of the records that result from processing prior to the SORT command, inserting two spaces between the two ranges of data using 2X

If multiple output files are desired, OUTFIL statements are used instead of OUTREC. The form of the OUTFIL statement is

```
OUTFIL FILES=xx [,INCLUDE(or OMIT)=logical expression]
              [,OUTREC=(field,<field,field,...>)
```

where 'xx' refers to a portion of a DDname cited in the JCL accompanying the SyncSort job.

NOTE: When using multiple output files, it is necessary to 'connect' the SyncSort control statements (OUTFIL) with the actual output files defined in the JCL. This is shown in examples in later sections.

The rules for INCLUDE/OMIT from section 1.2.1 apply in the OUTFIL statement. The only exception is that the FORMAT= statement (when all data formats are the same in the logical expressions) cannot be used. The rules for OUTREC are the same as when only one output file is produced.

For example:

- a. OUTFIL FILES=1,INCLUDE=(1,2,CH,GE,C'01',AND,1,2,CH,LE,C'61')
OUTFIL FILES=2,INCLUDE=(1,2,CH,EQ,C'70')
two output files are produced, each edited on data in positions 1 and 2 of records resulting from processing prior to the SORT command, where the first file contains character data in the range 01 through 61, and the second contains character data equal to 70 - the entire record resulting from processing prior to the SORT command is output
- b. OUTFIL FILES=1,INCLUDE=(1,2,CH,GE,C'01',AND,1,2,CH,LE,C'61'),
OUTREC=(1,100)

OUTFIL FILES=2,INCLUDE=(1,2,CH,EQ,C'70'),OUTREC=(101,100)

the INCLUDE conditions are the same as example 'a', however the first output file will contain data from positions 1 through 100 and the second file from positions 101 through 200 from records resulting from processing prior to the SORT command

- c. OUTFIL FILES=1,INCLUDE=(1,2,CH,GE,C'01',AND,1,2,CH,LE,C'61'),
OUTREC=(1,5,C'UPS')

OUTFIL FILES=2,INCLUDE=(1,2,CH,EQ,C'70'),OUTREC=(1,5,C'NYC')

the INCLUDE conditions are the same as example 'a', however the first output file will contain the literal string UPS in positions 6 through 8, while the second file will contain the literal string NYC in positions 6 through 8

NOTE: There are many other options available on the OUTREC and OUTFIL statements. Discussion of these options can be found in the SyncSort references cited previously.

1.2.5 Combining Records

The SUM statement can be used to combine records that have equal sort keys. The SUM statement results in a single record for all records with equal sort keys, where the single record contains the sum of specified numeric (e.g. zoned-decimal) data. Records are summed after INREC and INCLUDE/OMIT steps are executed. Thus, if an INREC statement is used, the SUM statement must take into account any reformatting of records that may have resulted from INREC processing.

The form of the SUM statement is

SUM FIELDS=(field<,field,field,...>) [,FORMAT=f]

The following general rules apply:

- SUM fields are defined by position,length (as in example 'a')
- the format of SUM fields may be expressed with a FORMAT= if the format of all fields being summed are the same
- the sum resulting from the combination of records with equal SORT keys must fit within the width of the field(s) specified in the SUM command; if the total becomes too large to be accommodated by the field width, more than one record will result from the SUM

For example:

- a. SORT FIELDS=(1,4,CH,A)
SUM FIELDS=(6,5,ZD)

totals zoned-decimal data in positions 6 through 10 for records containing like data in positions 1 through 4

- b. SORT FIELDS=(10,3,CH,D)
SUM FIELDS=(6,5,101,3),FORMAT=ZD

totals zoned-decimal data in positions 6 through 10, and 101 through 103 for records that contain like data in positions 10 through 12

2.0 SyncSort ON MVS

Using SyncSort on MVS requires not only a knowledge of the SyncSort command set, but also a knowledge of JCL and the rules for moving data from MVS to VM. The following sections contain complete examples of SyncSort jobs that run on the MVS operating system, showing JCL and addressing system rules for moving data from MVS to VM. In addition to the following 'generic' examples, a number of examples based on a given data set are shown in Appendix E.

2.1 Copying Records

SyncSort can be used to copy records from one or more cartridge to another cartridge or cartridges (remaining in the MVS environment for both input and output), or to copy records from one or more cartridges to your reader (input data on MVS, output data to VM). Entire files, selected records, and selected portions of records can be copied. The SyncSort statement SORT FIELDS=COPY is used when copying records. The numeric NOTES included on the right of JCL and SyncSort lines are NOT part of the code needed to run a job. They merely reference explanations of statements following each job. Not every job has a note on each statement. Refer to previous examples if a later example does not seem to contain enough of an explanation of either a JCL or SyncSort statement.

NOTE: though the term 'cartridge' is used in examples, any storage device (e.g., disk or tape) can be substituted

2.1.1 Cartridge-to-Cartridge

There are several JCL and SyncSort rules to follow when copying records from cartridge-to-cartridge. The following examples illustrate how to follow these rules.

2.1.1a Entire contents of a cartridge containing only one file

	NOTES
//XXXXXSYN JOB 99999999,'YOUR NAME',CLASS=B,TIME=1	1
/*ROUTE PRINT DOHVM1.XXXXX	2
//STEP1 EXEC PGM=SORT,REGION=2400K	3
//SYSOUT DD SYSOUT=A	4
//SORTIN DD UNIT=CART,DISP=SHR,	5
// DSN=U9999.TESTSTPE.XXXXX.CARTIN,	
// DCB=(RECFM=FB,LRECL=200),	
// LABEL=(,NL,EXPDT=98000),	
// VOL=SER=A12345	
//SORTOUT DD UNIT=CART,DISP=(NEW,KEEP,DELETE),	6
// DSN=U9999.TESTSTPE.XXXXX.CARTOUT,	
// LABEL=EXPDT=98030,	
// DCB=(RECFM=FB,LRECL=200)	
//SYSIN DD *	7
SORT FIELDS=COPY	8
/*	

- 1 Run this job in the 1-queue on MVS (one of the MVS queues for jobs that require cartridges). Replace the XXXXX with your user ID - replace the 99999999 with your charge code
- 2 Send all printouts associated with this job back to your reader on VM. Again, replace the XXXXX with your user ID.
- 3 Execute the SyncSort program.
- 4 All information provided by SyncSort as to errors, number of records processed, etc. will be returned to your reader on VM in a file with class A (i.e. in a file together with the log of your

JCL, files allocated, etc.). If this output is to be separated from the log, etc., use another letter, e.g. SYSOUT=B.

- 5 Define the input file --- in this case, a 'foreign', unlabeled cartridge with a record length of 200 (rules for defining cartridges for MVS jobs can be found in ISHS Technical Booklet #1, *Tape Management System*).
- 6 Define the output file.
- 7 SyncSort statements follow (you MUST HAVE a //SYSIN DD * statement)
- 8 Tell SyncSort to copy the entire cartridge.

2.1.1b Entire contents of a cartridge containing multiple files

NOTES

```

//XXXXXSYN JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD UNIT=CART,DISP=OLD,
//      DSN=U9999.TESTSTPE.XXXXX.CARTIN,
//      DCB=(RECFM=FB,LRECL=200),
//      LABEL=(1,NL,EXPDT=98000),
//      VOL=(,RETAIN,SER=012345)
//SORTOUT DD UNIT=CART,DISP=(NEW,PASS,DELETE),
//      DSN=U9999.TESTSTPE.XXXXX.CARTOUT,
//      DCB=(RECFM=FB,LRECL=200),
//      LABEL=(1,SL,EXPDT=98030),VOL=RETAIN
//SYSIN DD *
        SORT FIELDS=COPY
/*
//STEP2 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD UNIT=CART,DISP=OLD,
//      DSN=*.STEP1.SORTIN,
//      DCB=(RECFM=FB,LRECL=200),
//      LABEL=(2,NL,EXPDT=98000),
//      VOL=(,RETAIN,REF=*.STEP1.SORTIN)
//SORTOUT DD UNIT=CART,DISP=(NEW,KEEP,DELETE),
//      DSN=*.STEP1.SORTOUT,
//      LABEL=(2,SL,EXPDT=98030),
//      VOL=(,RETAIN,REF=*.STEP1.SORTOUT)
//SYSIN DD *
        SORT FIELDS=COPY
/*

```

- 1 A RETAIN parameter is used on the VOL statement to avoid having to remount the input and output cartridges in each successive job step.
- 2 The DSN, DCB, and VOL statements reference (REF) the parameters used in the first job step.

Add another job step for each file to be copied, repeating the JCL used in STEP2, and changing the file number used in the LABEL statement. You need not use the retain in the final job step and can change the VOL parameters as follows:

```
in SORTIN    VOL=REF=*.STEP1.SORTIN
in SORTOUT  VOL=REF=*.STEP1.SORTOUT
```

2.1.1c Selected records from one cartridge

NOTES

```
//XXXXXSYN JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD UNIT=CART,DISP=SHR,
//          VOL=SER=123456,DSN=POPULT.A12345
//SORTOUT DD UNIT=CART,DISP=(NEW,KEEP,DELETE),
//          DSN=U9999.TESTSTPE.XXXXX.CARTOUT,LABEL=EXPDT=98030
//SYSIN DD *
//          INCLUDE COND=(5,2,CH,EQ,C'70')
//          SORT FIELDS=COPY
/*
```

- 1 No DCB parameters are required if the SORTIN cartridge has a standard label and you want to copy the entire record. The SORTOUT cartridge will have the same record type, record length and blocksize as the SORTIN cartridge.
- 2 There is only one condition set for selecting records from the input cartridge --- each record must contain a 70 in positions 5 and 6. You could have used an OMIT statement instead of the INCLUDE: OMIT COND=(5,2,CH,NE,C'70').

You can add as many editing conditions as you like, following the rules for editing shown in section 1. It is possible to have so many edits that you run out of space for SyncSort to hold the edits in memory. However, this is a rare event and may be circumvented by using multiple job steps.

2.1.1d Selected records from several cartridges

NOTES

```
//XXXXXTAP JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD UNIT=CART,DISP=SHR,DSN=POPULT.A38274(0)
//          DD UNIT=AFF=SORTIN,DISP=SHR,DSN=POPULT.A38275(0)
//          DD UNIT=AFF=SORTIN,DISP=SHR,DSN=POPULT.A38276(0)
//          DD UNIT=AFF=SORTIN,DISP=SHR,DSN=POPULT.A38277(0)
//          DD UNIT=AFF=SORTIN,DISP=SHR,DSN=POPULT.A38278(0)
//SORTOUT DD UNIT=CART,DISP=(NEW,KEEP,DELETE),
//          DSN=U9999.TESTSTPE.XXXXX.MAT,LABEL=EXPDT=98030
//SYSIN DD *
//          INCLUDE COND=(37,2,GE,C'10',AND,37,2,LE,C'49',AND
//          39,1,EQ,C'2',AND,22,1,EQ,C'8'),FORMAT=CH
//          SORT FIELDS=COPY
/*
```

- 1 If the input cartridges contain fixed length records, they all must have the same record length. If the cartridges do not have the same blocksize, the with the largest blocksize must appear first in the list. Fixed length record cartridges with varying record sizes can be accommodated, as in the next example. Note, the affinity parameter (AFF) is used to use the same cartridge drive for all input cartridges.

- 2 Since there is only one statement to process all of the input cartridges, all data to be processed must lie in the same record position on each input cartridge.

2.1.1e Selected records from two cartridges with fixed length, but with different record lengths

NOTES

```
//XXXXXSYN JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//*****THIS CARTRIDGE HAS A RECORD LENGTH OF 90
//SORTIN DD UNIT=CART,DISP=SHR,DSN=POPULT.A31472(0)
//SORTOUT DD UNIT=CART,DISP=(NEW,KEEP,DELETE),
//          DSN=U9999.TESTSTPE.XXXXX.CARTOUT,
//          DCB=(RECFM=FB,LRECL=90),
//          LABEL=EXPDT=98030,
//          VOL=RETAIN
//SYSIN DD *
          INCLUDE COND=((5,2,GE,C'01',AND,5,2,LE,C'70'),AND,
                        (23,1,GE,C'2',AND,23,1,LE,C'5')),FORMAT=CH
          SORT FIELDS=COPY
/*
//STEP2 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//*****THIS CARTRIDGE HAS A RECORD LENGTH OF 100
//SORTIN DD UNIT=CART,DISP=SHR,DSN=POPULT.A31489(0)
//SORTOUT DD UNIT=CART,DISP=MOD,
//          DSN=*.STEP1.SORTOUT,
//          VOL=REF=*.STEP1.SORTOUT
//SYSIN DD *
          INCLUDE COND=((5,2,GE,C'01',AND,5,2,LE,C'70'),AND,
                        (23,1,GE,C'2',AND,23,1,LE,C'5')),FORMAT=CH
          SORT FIELDS=COPY
/*
```

- 1 The first cartridge read in this job has a record length of 90. The DCB statement contains parameters to give the output cartridge a record length of 90 and blocksize of 9000.
- 2 On the output tape, the disposition (DISP) is set to MOD. This will add new records from this job step to those already put onto the cartridge in step 1. The DSN, DCB, and VOL statements all refer to the SORTOUT statement in step 1 and get their settings from step 1. Note, even though the input tape has a record length of 100, only the first 90 characters will be copied to the output cartridge since the DCB parameter LRECL was set to 90 in step 1.

2.1.2 Cartridge-to-VM via the Reader

SyncSort can be used to bring back data from MVS to VM via your reader on VM. The following examples illustrate this process. Note that the SORTIN statements for input follow all the rules illustrated in section 2.1. However, there are new considerations for output to your reader. Since it is rare that you would want to copy an entire cartridge back to your reader, all the following examples involve selecting records with SyncSort edits.

2.1.2a Selected portions of selected records written to an output file

NOTES

```
//XXXXXSYN JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
```

```

//SORTIN DD UNIT=CART,DISP=SHR,DSN=POPULT.A12345(0)
//SORTOUT DD UNIT=SYSDA,DISP=(NEW,PASS),DSN=&&TEMP1,      1
//          DCB=(RECFM=FB,LRECL=30),
//          SPACE=(TRK,(10,1),RLSE)
//SYSIN DD *
          INCLUDE COND=(5,2,CH,NE,C'00')                2
          INREC FIELDS=(1,20,50,5,100,10)                3
          SORT FIELDS=COPY
          OUTREC FIELDS=(1,35)                            4
/*
//STEP9 EXEC PGM=IJKEFT01,DYNAMNBR=20                    5
//SYSTSPRT DD SYSOUT=*
//INFILE DD DISP=OLD,DSN=&&TEMP1                          6
//SYSTSIN DD *
TRANSMIT DOHVM1.XXXXX DDNAME(INFILE) NOLOG              7
/*

```

- 1 Portions of selected records are to be brought back to your reader on VM. The records are first written to disk on MVS (UNIT=SYSDA) where they are held temporarily in a file given the name &&TEMP1. The SORTOUT card shows one method of allocating temporary disk space on MVS to hold the SyncSort output file from STEP 1. (see appendix C for a discussion on allocating temporary disk space on MVS)
- 2 Records are selected that do not have a 00 in positions 5 and 6.
- 3 Data from positions 1 through 20, 50 through 54, and 100 through 109 are to be copied to the temporary file on MVS.
- 4 This statement is OPTIONAL since you have already specified that 35-characters are to be copied when the INREC statement was used earlier.
- 5 NETDATA transfer is used to move the file from temporary disk on MVS to your reader on VM (via the program IJKEFT01). The INFILE statement refers to the temporary disk file created in the SyncSort step (&&TEMP1). This method of moving a file from MVS-to-VM works with any record length - RECEIVE is used to move the file from your reader to your A-disk or some other disk on VM.
- 6 This statement identifies the file to be transferred from MVS disk space to your reader. The file will show up in your reader with a CLASS of K.
- 7 The XXXXX is replaced with your user ID on VM.

If the records to be sent to your reader have a length of 205 characters or less, you can use a different SORTOUT statement:

```
//SORTOUT DD SYSOUT=K
```

If this method is used, you should NOT use all the statements starting with //STEP9 through the end of the job since the records are written directly to your reader without using temporary disk. The records appear in your reader with a class of K.

2.1.2b Multiple output files returned to your reader

NOTES

```
//XXXXXB86 JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
```

```

//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD UNIT=CART,DISP=SHR,DSN=POPULT.A27113.CY1986(0)
//SORTOF1 DD UNIT=SYSDA,DISP=(NEW,PASS),DSN=&&TEMP1,           1
//          DCB=(RECFM=FB,LRECL=4),
//          SPACE=(TRK,(13,1),RLSE)
//SORTOF2 DD UNIT=SYSDA,DISP=(NEW,PASS),DSN=&&TEMP2,           1
//          DCB=(RECFM=FB,LRECL=4),
//          SPACE=(TRK,(13,1),RLSE)
//SYSIN DD *
      INCLUDE COND=(21,1,CH,EQ,C'2')
      INREC FIELDS=(1,2,48,2,253,4)
      SORT FIELDS=COPY
      OUTFIL FILES=1,OUTREC=(5,4),                               2
            INCLUDE=(1,2,CH,EQ,C'70')
      OUTFIL FILES=2,OUTREC=(1,4),
            INCLUDE=(1,2,CH,GE,C'01',AND,1,2,CH,LE,C'61')
/*
//STEP2 EXEC PGM=IJKEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//INFILE DD DISP=OLD,DSN=&&TEMP1
//SYSTSIN DD *
TRANSMIT DOHVM1.XXXXXX DDNAME(INFILE) NOLOG
/*
//STEP3 EXEC PGM=IJKEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//INFILE DD DISP=OLD,DSN=&&TEMP2
//SYSTSIN DD *
TRANSMIT DOHVM1.XXXXXX DDNAME(INFILE) NOLOG
/*

```

-
- 1 Two output files are defined on MVS disk. The first, &&TEMP1, is assigned the DD name SORTOF1 - the second, &&TEMP2, is assigned the DD name SORTOF2.
 - 2 The first output file (&&TEMP1) will contain data that had a 70 in positions 1 and 2, and will contain data from positions 253 through 256 on the original file. The second output file will contain data from positions 1 through 2, and 48 through 49 for records that a 01 through 61 in positions 1 and 2.

If there were more output files to be specified, the convention to follow is OUTFIL FILES=x referencing a JCL DDname of SORTOFx.

As stated previously, it is possible to direct files back to a DD name of SYSOUT, but only if they have a record length of 205 or less. Since there are two files to be sent to VM in this example, the SORTOF DD statements could read as follows:

```

//SORTOF1 DD SYSOUT=1
//SORTOF2 DD SYSOUT=2

```

Again, you would not use the statements that perform the NETDATA transfer of a file from MVS disk to your reader since the files are written directly to your reader without any use of temporary disk space. The files appear in your reader with a CLASS equal to whatever you assigned in the statement that specified the output SORTOF files, with class 1 and class 2 used in the above example.

2.2 Counting Records

As stated in section 1.2.5, SyncSort can combine records with equal sort keys, summing over specified numeric fields. If the numeric field contains the number 1, summing fields with SyncSort results in counts of records with equal sort keys.

2.2.1 Counts Output to a Single File in the VM Reader

NOTES

```

//XXXXXD86 JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD UNIT=CART,DISP=SHR,DSN=POPULT.A38362(0)
//SORTOUT DD SYSOUT=M
//SYSIN DD *
    INCLUDE COND=(22,1,CH,EQ,C'8')
    INREC FIELDS=(8,4,C'00001')
    SORT FIELDS=(1,4,CH,A)
    SUM FIELDS=(5,5,ZD)
    OUTREC FIELDS=(1,4,X,5,5,ZD)
/*

```

- 1 Counts will be returned to the VM reader in a file with class M.
- 2 Only records containing an 8 in position 22 of the input record are included
- 3 Data from positions 8 through 11 are retained; in addition to these data, the literal 00001 is added to each record in positions 5 through 9. The numeric field is defined as being 5-digits wide to make sure it will accommodate the largest sum of records across records with equal sort keys. The record that SyncSort will process from now on is:
 ABCD00001 where ABCD represents data from positions 8 through 11 in records from the input file
- 4 The data are sorted on positions 1 through 4.
 NOTE: Positions 1 through 4 contain the data from the input records originally contained in positions 8 through 11.
- 5 The data in positions 5 through 9 are summed for records containing like data in positions 1 through 4.
 NOTE: The data to be counted must be referred to as numeric, ZD (zoned-decimal) in this example, even though they were placed on each record as CH (character literal 00001) in the INREC statement.
- 6 Record counts are output to a reader file with class M, each record containing a count of records containing like data in positions 8 through 11 of the original records; a space is placed between the sort key and the sum by using an X. The ZD placed after the 5,5 removes leading zeroes from the counts. The ZD could have been left off the end of the OUTREC statement, resulting in leading zeroes on all counts. If no space were desired, as placed with the 'X', and leading zeroes were permissible, no OUTREC statement would be necessary, or the OUTREC statement could have been written as follows:

```
OUTREC FIELDS=(1,9)
```

2.2.2 Counts Output to a Temporary Disk File on MVS - Used as Input to a SAS Job

NOTES

```

//XXXXXD86 JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD UNIT=CART,DISP=SHR,DSN=POPULT.A38362(0)
//SORTOUT DD UNIT=SYSDA,DISP=(NEW,PASS),DSN=&&TEMP,      1
//          DCB=(RECFM=FB,LRECL=9),
//          SPACE=(TRK,(33,1),RLSE)
//SYSIN DD *
          INCLUDE COND=(22,1,CH,EQ,C'8')
          INREC FIELDS=(8,4,C'00001')
          SORT FIELDS=(1,4,CH,A)
          SUM FIELDS=(5,5,ZD)
          OUTREC FIELDS=(1,9)
/*
//STEP2 EXEC SAS,REGION=2400K
//FILE1 DD UNIT=SYSDA,DISP=SHR,DSN=&&TEMP      2
//SYSIN DD *
DATA NEW;
INFILE FILE1;
INPUT
@001 FIELD1 $CHAR4.
@005 FREQ1 5.      3
;
(more SAS statements)

```

- 1 SORTOUT is a file on MVS temporary disk (SYSDA) - the disk file is given the name &&TEMP (using the '&&' is a way to give the file a unique name since it adds as a prefix to TEMP the name assigned to the job as it runs on MVS)
- 2 The temporary disk file (&&TEMP) is referenced in the JCL as FILE1.
- 3 The numeric data in positions 5-9 are input by SAS - these data are counts of records that had the same data in positions 8 through 11 on the original file.

2.2.3 Multiple Output Files Used to Produce Two Sets of Counts - Sent to VM Reader as One File

NOTES

```

//XXXXXD86 JOB 99999999,'YOUR NAME',CLASS=B,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD UNIT=CART,DISP=SHR,DSN=POPULT.A38362(0)
//SORTOF1 DD UNIT=SYSDA,DISP=(NEW,PASS),DSN=&&TEMP1,      1
//          DCB=(RECFM=FB,LRECL=9),
//          SPACE=(TRK,(26,1),RLSE)
//SORTOF2 DD UNIT=SYSDA,DISP=(NEW,PASS),DSN=&&TEMP2,      1
//          DCB=(RECFM=FB,LRECL=9),
//          SPACE=(TRK,(26,1),RLSE)
//SYSIN DD *
          INCLUDE COND=(22,1,CH,EQ,C'8')
          SORT FIELDS=COPY
          OUTFIL FILES=1,INCLUDE=(1,2,CH,EQ,C'70'),      1
                   OUTREC=(1,2,48,2,C'00001')
          OUTFIL FILES=2,OMIT=(1,2,CH,EQ,C'70'),      1
                   OUTREC=(50,4,C'00001')

```

```

/*
//STEP2   EXEC PGM=SORT,REGION=2400K
//SYSOUT  DD SYSOUT=A
//SORTIN  DD UNIT=SYSDA,DISP=SHR,DSN=*&TEMP1           2
//        DD UNIT=SYSDA,DISP=SHR,DSN=*&TEMP2           2
//SORTOUT DD SYSOUT=M
//SYSIN   DD *
           SORT FIELDS=(1,4,CH,A)                       3
           SUM FIELDS=(5,5,ZD)                          3
           OUTREC FIELDS=(1,4,X,5,5,ZD)                  4
/*

```

- 1 Two output files are defined in the JCL (allocating temporary disk space to hold data that are to be counted later in the job). The SyncSort statements reference the two output files. The first file created contains records that have a 70 in positions 1 and 2, while the second file contains all other records. The output files contain data from different data from the input files in positions 1 through 4, but they both contain the literal 00001 in positions 5 through 9.
- 2 The input files for the second execution of SyncSort within the job are the two output files defined and created in the first execution of SyncSort.
- 3 The data are sorted on positions 1 through 4 and totaled on positions 5 through 9, creating counts of records with equal sort keys in positions 1 through 4.
- 4 The output file will appear in the VM reader with class M, containing the sort key in positions 1 through 5, and record counts in positions 7 through 11 (a space added in position 6 with the X), with no leading zeroes (the result of using the ZD format).

3.0 SyncSort ON VM

SyncSort may be used to accomplish the same tasks on VM as on MVS. Since most data analysts and researchers in the Department cannot use tapes on VM, SyncSort can be used to work with files maintained in personal storage on the A-disk or on temporary disk space allocated within a VM session. The majority of the SyncSort commands used on VM are the same as those used on MVS. The main difference between using SyncSort on the two operating systems are how a SyncSort job is started, how information concerning files to be used is conveyed to SyncSort, and how SyncSort is provided with information as to what commands are to be used.

Examples of uses of SyncSort on VM are shown in Appendix E.

3.1 Starting SyncSort on VM

A SyncSort job on VM is started using the command SSORT on the VM command line. For example:

```
SSORT <input> <output> <SyncSort instructions>
```

where <input> and <output> refer to files that are to be processed and produced, and <SyncSort instructions> usually refers to a file on VM that contains the SyncSort commands.

3.2 SyncSort Commands Specific to VM

There are several SyncSort commands that are used only on VM. Among them are FILESOUT and OPTION FILNM. They are used when multiple output files are to be produced.

The FILESOUT command is used with the SORT statement, for example:

```
SORT FIELDS=COPY,FILESOUT=3
```

would be used when 3 output files are to be produced.

The OPTION FILNM statement is used to pass SyncSort information as to the names of the output files, for example:

```
OPTION FILNM=(,OUT2,OUT3)
```

would be used when 3 output files are to be produced. The names OUT2 and OUT3 are DDnames that have previously been associated with VM file names through the use of FILEDEF statements (FILEDEF statements are discussed in subsequent sections). The name of the first output file would have appeared on the command line where the SSORT command was issued.

NOTE: Up to 9 output files can be produced with one SyncSort job.

3.3 Passing File Information to SyncSort on VM

There are several ways to tell SyncSort what files are to be used, direct entry on the command line, FILEDEFS, or combinations of direct entry and FILEDEFS. SyncSort must receive information about the name of the input file, output file, and file containing the SyncSort commands. If not provided with this information, SyncSort will look for files with default names, overwrite the input file with the output file, etc. This guide does not cover all the possibilities of passing file information to SyncSort. The guide does address the situation where all necessary information is passed to SyncSort, the safest method of using SyncSort on VM for the non-programmer.

3.3.1 Single Input File and Single Output File

The following sections illustrate two methods of using SyncSort on VM to process one input file, producing one output file.

3.3.1a Direct entry on the command line

```
SSORT MYINPUT DATA A MYOUTPUT DATA A MYSORT SORT A
```

MYINPUT DATA A is a VM file used by SyncSort as input

MYOUTPUT DATA A is a VM file produced by SyncSort

MYSORT SORT A is a VM file that contains SyncSort commands

NOTE: If an entry for the file that contains the SyncSort commands is left off the command line, SyncSort prompts the user for SyncSort commands. Unless the SyncSort job is simple, this is not a very efficient way in which to convey sort information to SyncSort.

3.3.1b Using FILEDEFS

```
SSORT DDN DDIN DDN DDOUT MYSORT SORT A
```

DDIN and DDOUT are DDnames produced by FILEDEF statements, e.g.

```
FILEDEF DDIN DISK MYINPUT DATA A
```

```
FILEDEF DDOUT DISK MYOUTPUT DATA A
```

NOTE: The names DDIN and DDOUT are arbitrary - any names can be used.

3.3.1c Combination of direct entry and FILEDEFS

SSORT DDN DDIN MYOUTPUT DATA A MYSORT SORT A

-or-

SSORT MYINPUT DATA A DDN DDOUT MYSORT SORT A

3.3.2 Multiple Input and/or Output Files

Up to 9 input files may be processed and up to 9 output produced with a single SSORT statement. Multiple input files may be referenced either directly on the command line or with FILEDEFS. Multiple output files must be referenced by a combination of direct command line entry, FILEDEFS, and entries in a VM file of SyncSort commands.

3.3.2a Direct entry on the command line - multiple input, single output

SSORT (MYIN1 DATA A MYIN2 DATA A) MYOUT DATA A MYSORT SORT A

MYIN1 DATA A and MYIN2 DATA A are input files (up to 9 may be used)

MYOUT DATA A is an output file

3.3.2b Using FILEDEFS - multiple input, single output

SSORT (DDN MY1 DDN MY2) DDN MY3 MYSORT SORT A

MY1 and MY2 and MY3 are DDnames produced by FILEDEF statements, e.g.

FILEDEF MY1 DISK MYIN1 DATA A

FILEDEF MY2 DISK MYIN2 DATA A

FILEDEF MY3 DISK MYOUT DATA A

3.3.2c Combination of direct entry and FILEDEFS - single input, multiple output

SSORT MYINPUT DATA A DDN OUT1 MYSORT SORT A

OUT1 is a DDname for the first output file produced by a FILEDEF statement, e.g.

FILEDEF OUT1 DISK MYOUT1 DATA A

NOTE: The other output files must also be assigned DDnames with FILEDEF statements, e.g.

FILEDEF OUT2 DISK MYOUT2 DATA A

FILEDEF OUT3 DISK MYOUT3 DATA A

and the file of SyncSort commands must contain the following statements:

SORT FIELDS=COPY,FILESOUT=3

and

OPTION FILNM=(,OUT2,OUT3)

APPENDIX A

SyncSort References

for VM SyncSort CMS Programmer's Guide

for MVS SyncSort OS Programmer's Guide

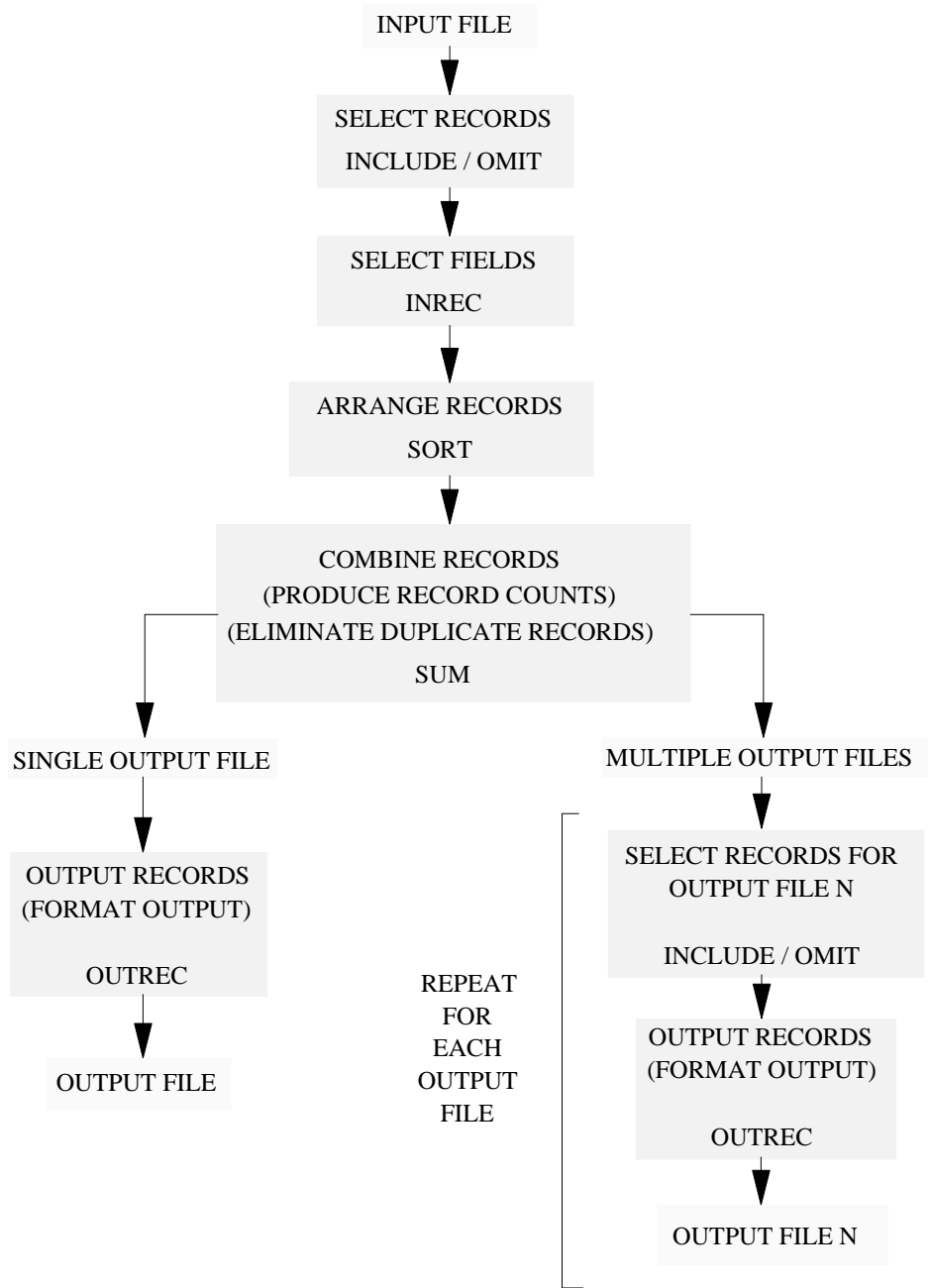
both available from

SyncSort Incorporated
50 Tice Boulevard
Woodcliff Lake, NJ 07675

(201)930-8230

APPENDIX B

The Flow of SyncSort
 (Adapted from *SyncSort OS Applications Workshop*)



APPENDIX C

Allocating Temporary Disk Space on MVS

NOTE: This topic is covered in more detail in Technical Booklet 5, *Using Disk on the MVS IBM*, available from ISHS on the concourse. Parts of this appendix are abstracted from Technical Book 5.

Temporary disk space may be allocated for use on MVS. Temporary means that the disk space is allocated for the duration of the job in which it is allocated. When the job is completed, the disk space is no longer allocated and any files written to the temporary space are erased. The main thing to remember in allocating temporary disk space is to allocate enough to hold all records written to files in the temporary space, but not to allocate so much as to be wasteful of system resources. This rule should be followed even though the space is only being allocated for the duration of a given job.

The JCL command SPACE is used to allocate temporary disk space on MVS. A full discussion of the SPACE statement and its various parameters can be found in the *OS/VS2 MVS JCL* manual.

Basics

Temporary disk space on MVS is referred to as SYSDA. A certain portion of the total disk space available on MVS has been designated as SYSDA, or scratch space.

The 3380 and 3390 disk drives used on MVS are divided into subunits called cylinders. Cylinders are composed of tracks. The number of cylinders and tracks on a disk are as follows:

3380	1 Disk ->	885 Cylinders			
		1 Cylinder	->	15 Tracks	
				1 Track	-> 47,476 Bytes (characters)
3390	1 Disk ->	2,226 Cylinders			
		1 Cylinder	->	15 Tracks	
				1 Track	-> 56,664 Bytes (characters)

Temporary disk space may be allocated in three ways: cylinders (CYL), tracks (TRK), or blocks (blocksize). In order to determine the amount of space required, it is recommended that you use the CALCDISK utility available on VM. To access it, type CALCDISK at ay command prompt.

Temporary space is allocated in terms of primary and secondary extents. The amount of space requested in the primary extent is allocated as soon as a job step starts, while space requested in the secondary extents is allocated as needed, up to 15 times the amount requested.

Temporary disk space should be assigned a name (JCL parameter DSN) that begins with the characters &&, e.g. DSN=&&TEMP. This naming convention will uniquely identify the file to the operating system, and allow for easy deletion once the job is completed.

NOTE: It is no longer necessary to specify a blocksize since the system will automatically determine the most efficient blocksize for a given record length and disk type.

There are a number of dispositions (JCL parameter DISP) that can be assigned to the file being written to temporary disk space. When using SyncSort, the dispositions that will be used are:

DISP=(NEW,PASS) when creating a file with SyncSort

DISP=OLD when reading a file created in a previous job step

DISP=MOD when adding records to a file created in a previous job step

Example

A file with a record length of 100 is to be produced by a SyncSort job step and used in a subsequent job step (e.g. another SyncSort job, SAS, etc.). Approximately 25,000 records are expected as output. The examples shown are for a 3390 disk.

1. Allocation by blocksize

```
//SORTOUT DD UNIT=SYSDA,DISP=(NEW,PASS),DSN=&&TEMP1,  
//          DCB=(RECFM=FB,LRECL=100,BLKSIZE=27900),  
//          SPACE=(27900,(86,4),RLSE)
```

The file is assigned the name &&TEMP1. The DCB statement shows that there will be 279 records in each block (27,900/100). Blocksize is calculated as (27,998/100), where 27,998 is the maximum recommended blocksize for a 3390 disk (or 23,476 for a 3380 disk). Space is allocated in terms of blocksize, with 24,000 records to be accommodated in the primary extent (279*86), and 1,116 records (279*4) each time a secondary extent is allocated. The RLSE parameter results in any unused space being released when the file &&TEMP1 is closed.

2. Allocation by tracks

(same first two lines as allocation by blocksize)
// SPACE=(TRK,(43,2),RLSE)

There is room for approximately 56,664 characters in a track. In this example, 2 blocks (558 records) would fit within a track (56,664/27,900). The primary extent will accommodate approximately 25,000 records (279*2*43), with 1,116 records (279*2*2) allowed in each secondary extent.

3. Allocation by cylinders

(same first two lines as allocation by blocksize)
// SPACE=(CYL,(3,1),RLSE)

Each cylinder is composed of 15 tracks. Since the track allocation for the primary extent was 43 in the previous example, 3 cylinders are allocated for the primary extent, and 1 for each secondary extent.

APPENDIX D

Examples of Uses of SyncSort on MVS

The following examples are based on a file that contains information on patients discharged from hospitals in New York State - CPU times shown following each example are actual times needed to complete the task on an IBM 3081 mainframe running release 3.3 of SyncSort under the MVS/XA operating system. There are approximately 2.5 million fixed length records (174 characters per record, blocksize of 23316) in the data base. The data are located on a 3380 disk having the following record layout (NOTE: not all data present in the data base are shown in the record layout):

position (from-to)	contents
8-11	hospital number
36-38	patient age
39-39	patient sex
46-47	county number - residence of patient
57-59	principal diagnosis
62-64	other diagnosis #1
67-69	other diagnosis #2
72-74	other diagnosis #3
77-79	other diagnosis #4
135-136	county number - location of hospital
171-174	length of stay in hospital (days)

NOTE: note the numbers on the right of each line in the examples are NOT part of the job - they merely refer to the explanations that follow each example

Example 1 Extract and Copy Records - Simple Record Selection

extract all the records from the file where the hospitals are located in Albany county (a county number equal to 01) and copy the selected records to another cartridge

```
//XXXXXXEX1 JOB 99999999,'YOUR NAME',CLASS=A,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD DISP=SHR,DSN=HOSPITAL.DISCHARG.CY1990 1
//SORTOUT DD DISP=(NEW,KEEP),UNIT=CART, 2
// DSN=U9999.TESTSTPE.XXXXX,HOSPALB,
// DCB=(RECFM=FB,LRECL=174),
// LABEL=EXPDT=98030
//SYSIN DD *
INCLUDE COND=(135,2,CH,EQ,C'01') 3
SORT FIELDS=COPY 4
/*
```

- 1 the file defined in the SORTIN DD statement is the 1990 hospital discharge file
- 2 the file defined in the SORTOUT DD statement is a cartridge, where records have a fixed length of 174 characters

- select those records where the hospital is located in Albany county - the county number is treated as character data - if you were sure that all county numbers were numeric, and had leading zeroes, you could also have written the INCLUDE statement as follows:

```
INCLUDE COND=(135,2,ZD,EQ,1)
```

- copy the selected records to the output cartridge

records read	2543621	records copied	50451	CPU time	5.83 seconds
--------------	---------	----------------	-------	----------	--------------

Example 2 Extract and Copy Records - Complex Record Selection

once again, extract all records for hospitals located in Albany county, but also restrict the selection to records where the primary diagnosis is in the the range of 630 through 676

use the same JCL and SyncSort statements that are used in example 1 - change the include statement to the following

```
INCLUDE COND=(135,2,EQ,C'01',AND,
              57,3,GE,C'630',AND,57,3,LE,C'676'),FORMAT=CH
```

notice that the FORMAT=CH placed at the end of the INCLUDE statement, and after the final parenthesis replaces the use of multiple CH's within the parentheses

records read	2543621	records copied	6828	CPU time	5.55 seconds
--------------	---------	----------------	------	----------	--------------

Example 3 Extract and Count Records

count the number of discharges from each hospital and send the counts back to your reader on VM

```
//XXXXXEX3 JOB 99999999,'YOUR NAME',CLASS=A,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD DISP=SHR,DSN=HOSPITAL.DISCHARG.CY1990
//SORTOUT DD SYSOUT=M
//SYSIN DD *
INREC FIELDS=(8,4,C'000001')
SORT FIELDS=(1,4,CH,A)
SUM FIELDS=(5,6,ZD)
OUTREC FIELDS=(1,4,X,5,6,ZD)
/*
```

- the output from this job will be sent to your reader on VM (if the /*ROUTE PRINT line is present in the job and contains your VM user ID replacing the XXXXX)
- only the positions containing the hospital number are read from the file, and a '000001' is added to each record (this field must be wide enough to accomodate the maximum number of discharges expected at any particular hospital in the data base) - since only data from positions 8 through 11 were read from the file, the data are 'moved' to the left, and the records seen by SyncSort in subsequent statements are 9 characters long, with the hospital number in positions 1 through 4, and a '000001' in positions 5 though 10 - e.g., if the hospital number was 1200, adding the 000001 would result in the a record that looked like this - 1200000001

- 3 sort the records by hospital number
- 4 sum the records over positions 5 through 10 (since these positions only contain a numeric 1, a count of records will be produced for records with the same hospital number in positions 1 through 4)
- 5 format the output sent to VM, with the hospital number in positions 1 through 4, a space (produced with the 'X'), and counts with leading zeroes suppressed (suppression caused by placing the ZD in the OUTREC statement - without the ZD, counts will have leading zeroes)

records read 2543621 records copied (counts) 266 CPU time 23.32 seconds

Example 4 Extract and Count Records - Use Output from SyncSort as SAS Input

extract all records from five hospitals, count the number of records by age and sex in the selected hospitals, hold the counts in a file and read that file with a SAS job that puts the discharges into age groups by hospital

```
//XXXXXEX4 JOB 99999999,'YOUR NAME',CLASS=A,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD DISP=SHR,DSN=HOSPITAL.DISCHARG.CY1990
//SORTOUT DD DISP=(NEW,PASS),UNIT=SYSDA,DSN=*&&TEMP, 1
// (DCB=FB,LRECL=14),
// SPACE=(TRK,(17,1),RLSE)
//SYSIN DD *
INCLUDE COND=(8,4,EQ,C'0001',OR, 2
           8,4,EQ,C'0004',OR,
           8,4,EQ,C'0005',OR,
           8,4,EQ,C'0128',OR,
           8,4,EQ,C'0267'),FORMAT=CH
INREC FIELDS=(8,4,39,1,36,3,C'000001') 3
SORT FIELDS=(1,8,CH,A) 4
SUM FIELDS=(9,6,ZD) 5
/*
//STEP2 EXEC SAS,REGION=2400K 6
//FILEIN DD DISP=SHR,UNIT=SYSDA,DSN=*&&TEMP 7
//SYSIN DD *

proc format; 8
value $sex '1'='MALES'
           '2'='FEMALES';
value $age low-'009'='less than 10'
           '010'-'019'='10-19'
           '020'-'064'='20-64'
           '065'-'084'='65-84'
           '085'-high='85 plus';
run;

data hosp; 9
infile filein;
input hosp $ 1-4 sex $ 5-5 age $ 6-8 freq 9-14;
run;

proc freq data=hosp; 10
tables age*sex/norow nocol nocum nopercen;
by hosp;
weight freq;
```

```
format sex $sex. age $age,;
run;
```

- 1 the SyncSort output is written to a TEMPORARY disk file with the name &&TEMP - each record written to the file will be 14 characters long (LRECL=14), which is equal to the 4 character hospital number + the 1 character patient sex + the 3 character patient age + a 6 digit numeric field that will be used to accumulate the totals

space must be allocated to hold the file - it is allocated in blocks - there are 5 hospitals, with 2 sexes and a maximum of 121 different ages in each hospital (allowable ages on the file are 0 to 120) - thus, the maximum number of records that will have to be stored temporarily is 1210 (or 5 x 2 x 121) - with a record length of 14 (LRECL=14) - Appendix 3 contains more information on allocating temporary disk space
- 2 records from 5 hospitals are extracted from the large file
- 3 select only those data elements that you need (hospital number, sex, age), and add a 000001 to each record to use for counting
- 4 sort the extracted records by a field composed of hospital number, sex, and age
- 5 use the 000001 field to count records having the same hospital number, sex, and age
- 6 run a SAS job that will use the data extracted via SyncSort
- 7 use as input the counts produced by SyncSort
- 8 write some formats to label and group the data
- 9 the SAS DATA STEP now has to read a maximum of 1210 records (actually 978), NOT the 2.5 million records on the original file
- 10 PROC FREQ now has to count a maximum of 1210 observations, not the total number of discharges from the 5 hospitals - the data have been 'pre-counted' with SyncSort, and the WEIGHT option is used with PROC FREQ to tell SAS that each observation represents not one, but many discharges

SyncSort STEP			
records read	2543621	records copied (counts)	978 CPU time 8.16 seconds
SAS STEP			
records read	978		CPU time 0.71 seconds

Example 5 Extract and Count Records when Multiple Fields on a Record Contain Data that are to be Combined into One Count

count the number of the various diagnoses made at hospital number 0001 - since there are five locations on each record that might contain a diagnosis (principal, plus four possible secondary), the counting must be done in two steps

```
//XXXXXXE5 JOB 99999999,'YOUR NAME',CLASS=A,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD DISP=SHR,DSN=HOSPITAL.DISCHARG.CY1990
```

```

//SORTOF1 DD DISP=(NEW,PASS),UNIT=SYSDA,DSN=&&TEMP1,           1
//          DCB=(RECFM=FB,LRECL=3),
//          SPACE=(TRK,(10,1),RLSE)
//SORTOF2 DD DISP=(NEW,PASS),UNIT=SYSDA,DSN=&&TEMP2,           1
//          DCB=(RECFM=FB,LRECL=3),
//          SPACE=(TRK,(10,1),RLSE)
//SORTOF3 DD DISP=(NEW,PASS),UNIT=SYSDA,DSN=&&TEMP3,           1
//          DCB=(RECFM=FB,LRECL=3),
//          SPACE=(TRK,(10,1),RLSE)
//SORTOF4 DD DISP=(NEW,PASS),UNIT=SYSDA,DSN=&&TEMP4,           1
//          DCB=(RECFM=FB,LRECL=3),
//          SPACE=(TRK,(10,1),RLSE)
//SORTOF5 DD DISP=(NEW,PASS),UNIT=SYSDA,DSN=&&TEMP5,           1
//          DCB=(RECFM=FB,LRECL=3),
//          SPACE=(TRK,(10,1),RLSE)
//SYSIN   DD *
          INCLUDE COND=(8,4,CH,EQ,C'0001')                       2
          SORT          FIELDS=COPY
          OUTFIL        FILES=1,OUTREC=(57,3),INCLUDE=(57,3,CH,NE,C' ') 3
          OUTFIL        FILES=2,OUTREC=(62,3),INCLUDE=(62,3,CH,NE,C' ') 3
          OUTFIL        FILES=3,OUTREC=(67,3),INCLUDE=(67,3,CH,NE,C' ') 3
          OUTFIL        FILES=4,OUTREC=(72,3),INCLUDE=(72,3,CH,NE,C' ') 3
          OUTFIL        FILES=5,OUTREC=(77,3),INCLUDE=(77,3,CH,NE,C' ') 3
/*
//STEP2   EXEC PGM=SORT,REGION=2400K
//SYSOUT  DD SYSOUT=A
//SORTIN  DD DISP=SHR,UNIT=SYSDA,DSN=&&TEMP1                     4
//          DD DISP=SHR,UNIT=SYSDA,DSN=&&TEMP2
//          DD DISP=SHR,UNIT=SYSDA,DSN=&&TEMP3
//          DD DISP=SHR,UNIT=SYSDA,DSN=&&TEMP4
//          DD DISP=SHR,UNIT=SYSDA,DSN=&&TEMP5
//SORTOUT DD SYSOUT=M
//SYSIN   DD *
          INREC        FIELDS=(1,3,C'000001')                       5
          SORT          FIELDS=(1,3,CH,A)                           6
          SUM          FIELDS=(4,6,ZD)                               7
/*

```

- 1 five different output files are defined, each of which is to be used to hold a diagnosis from a discharge record
- 2 records from hospital number 0001 are selected
- 3 five OUTFIL statements are used to direct the diagnoses on each record to the files defined in the JCL as SORTOF1, SORTOF2, etc. - the contents of each diagnosis field is checked and only non-blank diagnosis are written to the output files
- 4 the five files created in STEP1 are used as input to SyncSort in STEP2
- 5 a '000001' is added to each record on input - it will be used in producing counts of diagnoses
- 6 the records are sorted by diagnosis
- 7 counts are produced with the SyncSort SUM command (using the 000001 that was added to each record with the INREC statement) - since no OUTREC command was used, the counts will be returned to the reader on VM with leading zeroes, e.g., V30001256, where the first three characters are a diagnosis and the last six characters are a count

SyncSort STEP1

records read	2543621	records copied (counts)	CPU time	6.69 seconds
		SORTOF1		22727
		SORTOF2		18527
		SORTOF3		14352
		SORTOF4		10850
		SORTOF5		8199
SyncSort STEP2				
records read	74655		CPU time	1.55 seconds

Example 6 Extract and Count Records Using Two Fields to Accumulate Counts - Use Counts as Input for SAS

compute the average length of stay by hospital for patients discharged from hospitals in Manhattan (a county number equal to 60)

```
//XXXXXXEX6 JOB 99999999,'YOUR NAME',CLASS=A,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD DISP=SHR,DSN=HOSPITAL.DISCHARG.CY1990
//SORTOUT DD DISP=(NEW,PASS),UNIT=SYSDA,DSN=&&TEMP,
// DCB=(RECFM=FB,LRECL=18),
// SPACE=(TRK,(6,1),RLSE)
//SYSIN DD *
        INCLUDE COND=(135,2,CH,EQ,C'60') 1
        INREC FIELDS=(8,4,C'000001',C'0000',171,4) 2
        SORT FIELDS=(1,4,CH,A) 3
        SUM FIELDS=(5,6,ZD,11,8,ZD) 4
/*
//STEP2 EXEC SAS,REGION=2400K
//FILEIN DD DISP=SHR,UNIT=SYSDA,DSN=&&TEMP 5
//SYSIN DD *
options nocenter;
title1 'HOSPITALS IN COUNTY = 60';
title2 'DISCHARGES, DAYS, AVERAGE LENGTH OF STAY';
data _null_; 6
file print; infile filein;
input hosp $ 1-4 dis 5-10 days 11-18;
alos=days/dis;
put (hosp dis days alos) ($char4. 8. 10. 8.1);
run;
```

- 1 records are selected for hospitals in county 60
- 2 only the hospital number (postions 8-11) and the length of stay (positions 171-174) are read from the input records - a '000001' is added to be used in counting discharges - a '0000' is placed in front of the number of days (171,4) since the days are to be accumulated by hospital and the total number of patient days must fit in the positions being used for the count - adding four leading zeroes gives enough room to hold the total days per hospital
- 3 the records are sorted by hospital
- 4 the SUM statement is used to produce two counts - the number of discharges in postions 5 through 10, and the total patient days in positions 11 through 18
- 5 the counts produced by SyncSort are used as input to a SAS step

- 6 no SAS dataset is to be produced, so DATA _NULL_ is used - the data step is used to read the number of discharges and days in each hospital, to calculate the average length of stay (ALOS), then to print the results - SAS reads 28 records (one per hospital in Manhattan) instead of the 2.5 million records in the original file

SyncSort STEP

records read	2543621	records copied (counts)	28	CPU time	9.42 seconds
--------------	---------	-------------------------	----	----------	--------------

SAS STEP

records read	28	CPU time	0.28 seconds
--------------	----	----------	--------------

APPENDIX E

Examples of Uses of SyncSort on VM

NOTE: When record positions are referred to in any of the examples, a RECORDTYPE of FIXED is assumed. If the RECORDTYPE is VARIABLE, SyncSort considers the 4-byte header (the place where information as to record length is stored) as the first 4 record positions. Thus, with variable length records, data that appears to be in position 1 would have to be described to SyncSort as in position 5, with every subsequent data element moved over 4 positions.

SORT A FILE ON ONE FIELD IN ASCENDING ORDER

Sort all records in file MYFILE DATA A based upon data contained in positions 10-15 in ascending order.

First create a file MYSORT SORT A that contains the following SyncSort command:

```
SORT FIELDS=(10,6,CH,A)
```

then run SyncSort as follows:

```
SSORT MYFILE DATA A MYFILE DATA A MYSORT DATA A
```

EXTRACT RECORDS

Extract all records from file MYFILE DATA A that have a 2 in position 4, and write the records to file MYTWOS DATA T.

First create a file MYSORT SORT A that contains the following SyncSort commands:

```
INCLUDE COND=(4,1,CH,EQ,C'2')
SORT FIELDS=COPY
```

then run SyncSort as follows:

```
SSORT MYFILE DATA A MYTWOS DATA T MYSORT SORT A
```

COUNT RECORDS

Produce a file of counts for all possible combinations of characters in positions 5 and 6 of file MYFILE DATA A. Place the counts in MYCOUNTS DATA T.

First create a file MYSORT SORT A that contains the following SyncSort commands:

```
INREC FIELDS=(5,2,C'0001')
SORT FIELDS=(1,2,CH,A)
SUM FIELDS=(3,4,ZD)
OUTREC FIELDS=(1,2,X,3,4,ZD)
```

then run SyncSort as follows:

```
SSORT MYFILE DATA A MYCOUNTS DATA T MYSORT SORT A
```

EXTRACT DATA

Extract data from positions 1 through 10 and 20 through 25 from MYFILE DATA A and place these data in file MYREDUCE DATA T.

First create a file MYSORT SORT A that contains the following SyncSort commands:

```
INREC FIELDS=(1,10,20,6)  
SORT FIELDS=COPY
```

then run SyncSort as follows:

```
SSORT MYFILE DATA A MYREDUCE DATA T MYSORT SORT A
```

COMBINE SPECIFIED RECORDS FROM SEVERAL FILES

Combine records having a 70 in positions 1 and 2 from the three files MYIN1 DATA A, MYIN2 DATA A, MYIN3 DATA A. Place the output in MYTHREE SONS T.

First use FILEDEFS to assign DDnames to the input files:

```
FILEDEF IN1 DISK MYIN1 DATA A  
FILEDEF IN2 DISK MYIN2 DATA A  
FILEDEF IN3 DISK MYIN3 DATA A
```

then create a file MYSORT SORT A that contains the following SyncSort commands:

```
INFILE COND=(1,2,CH,EQ,C'70')  
SORT FIELDS=COPY
```

then run SyncSort as follows:

```
SSORT (DDN IN1 DDN IN2 DDN IN3) MYTHREE SONS T MYSORT SORT A
```

SORT A FILE IN ON SEVERAL FIELDS DESCENDING ORDER

Sort file MYDATA DATA A in ascending order of positions 1 through 5, descending order of positions 6 through 10, and ascending order of positions 21 through 25 Only retain the sorted portion of the file in an output file MYSRT DATA T.

First create a file MYSORT SORT A that contains the following SyncSort commands:

```
INREC FIELDS=(1,10,21,5)  
SORT FIELDS=(1,5,CH,A,6,5,CH,D,11,5,CH,A)
```

then run SyncSort as follows:

```
SSORT MYDATA DATA A MYSRT DATA T MYSORT SORT A
```

ELIMINATE RECORDS WITH DUPLICATE DATA IN SPECIFIED POSITIONS

Eliminate duplicate records from a file MYDUPS DATA A, using the data in positions 1 through 8 to check for duplicates, and creating a file MYNODUPS DATA T that has no duplicate records.

First create a file MYSORT SORT A that contains the following SyncSort commands:

```
SORT FIELDS=(1,8,CH,A)
SUM FIELDS=NONE
```

then run SyncSort as follows:

```
SSORT MYDUPS DATA A MYNODUPS DATA T MYSORT SORT A
```

PRODUCE SEVERAL OUTPUT FILES CONTAINING SPECIFIED RECORDS

Extract records from MYINPUT DATA A, creating three files based on the contents on positions 20 and 21: if 10 through 19, place records in MY1019 DATA T; if 20 through 29, place records in MY2029 DATA T; place all others in MOUTH DATA T.

First use FILEDEFS to assign DDnames to the output files:

```
FILEDEF OUT2 DISK MY1019 DATA T
FILEDEF OUT3 DISK MY2029 DATA T
```

then create a file MYSORT SORT A that contains the following SyncSort commands:

```
SORT FIELDS=COPY,FILESOUT=3
OUTFIL FILES=1,OMIT=(20,2,CH,GE,C'10',AND,20,2,CH,LE,C'29')
OUTFIL FILES=2,INCLUDE=(20,2,CH,GE,C'10',AND,20,2,CH,LE,C'19')
OUTFIL FILES=3,INCLUDE=(20,2,CH,GE,C'20',AND,20,2,CH,LE,C'29')
OPTION FILNM=(,OUT2,OUT3)
```

then run SyncSort as follows:

```
SSORT MYINPUT DATA A MOUTH DATA T MYSORT SORT A
```

Appendix F

Using ETHERNET to Send SyncSort Output from MVS Directly to a PC

It is now possible to run a job on MVS and have your output directed to a PC that is connected to the mainframe via ethernet. This allows you to bypass sending a file to VM, moving it from your reader, then to a T-disk, then downloading it to a PC. Any PC on the ethernet network can be used as a destination for the file(s). The PC must be placed in 'server mode' to receive the file.

Here are the steps you should follow:

1. Set up a job on to run on MVS that will write your data to a disk file.
NOTE: You can't write to a temporary file, e.g., &&TEMP. Your file must have a real name, e.g., XXXXX.TESTSDSK.XXXXX.MYDATA - you can write the file to SYSDA.
2. Add the file transfer commands as another job step. The content of these commands will vary according to: the 'ethernet address' of the PC that is to receive the data; whether or not the receiving PC is 'password protected' on the network; the name of the files on MVS and the destination directory and filename on the PC
3. Add a final job step that will delete your file from MVS disk after the file transfer.
4. Place the appropriate lab PC in 'server mode' by entering TN3270 at the DOS prompt.
4. Send the job from VM to MVS to create and download the data file.
5. After the job runs, take the PC out of 'server mode' by pressing ESC.

NOTE: This topic is covered in more detail in Technical Booklet 22, *Batch Submission of MVS FTP*, available from ISHS on the concourse.

The following is a complete example that uses the file described in Appendix D.

```
//XXXXXFTP JOB 99999999,'YOUR NAME',CLASS=A,TIME=1
/*ROUTE PRINT DOHVM1.XXXXX
//STEP1 EXEC PGM=SORT,REGION=2400K
//SYSOUT DD SYSOUT=A
//SORTIN DD DISP=SHR,DSN=HOSPITAL.DISCHARG.CY1990
//SORTOUT DD DISP=(NEW,CATLG),UNIT=SYSDA, 1
// DSN=XXXXX.TESTSDSK.XXXXX.TESTFTP,
// SPACE=(TRK,(68,1),RLSE,
// DCB=(RECFM=FB,LRECL=150)
//SYSIN DD *
INCLUDE COND=(8,4,EQ,C'0001',AND,36,3,GE,C'085'),FORMAT=CH 2
INREC FIELDS=(1,150) 3
SORT FIELDS=COPY
/*
//STEPFTP EXEC PGM=FTP,COND=(0,NE,STEP1) 4
//OUTPUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//INPUT DD *
<ethernet address of receiving PC> 5
<username password of receiving PC> 6
cd c:\ 7
put 'XXXXX.TESTSDSK.XXXXX.TESTFTP' MYPCFILE.DAT 8
quit 9
```

```
/*
//STEPDEL EXEC PGM=IEFBR14,COND=(0,NE,STEPFTP) 10
//SYSPRINT DD SYSOUT=*
//DELFILE DD DISP=(OLD,DELETE,DELETE), 11
// DSN=XXXXX.TESTSDSK.XXXXX.TESTFTP
```

- 1 an file is defined to receive the output from SyncSort - the file is CATALOGED
- 2 records are selected from the input file with a hospital number of 0001 and an age equal to or greater than 85
- 3 only 150 characters of each record read and copied
- 4 the FTP program on MVS is used to copy the output from SyncSort (now on disk) to a PC
- 5 you must provide the ethernet address for the receiving PC
- 6 if the receiving PC is password protected, you must supply both the appropriate username and password - if there is no password protection, any username will work
- 7 this line is OPTIONAL - it changes the directory on the receiving PC to C:\
- 8 the PUT command is used to tranfer the MVS file (in quotes) to the PC (no quotes on the PC filename)
- 9 QUIT ends the batch FTP session on MVS
- 10 another job step is added to delete the SyncSort output that us still on disk on MVS
- 11 the file to be deleted is specified

Appendix G

Some Other Useful SyncSort Statements

As stated in the introduction, this guide does not cover all the tasks that can be accomplished using SyncSort and all the statements and options available to a SyncSort user. The object of this guide is to point out some useful features of SyncSort in a 'non-intimidating' fashion. In addition to the information already covered, there are a few more SyncSort features that you may find useful.

SKIPREC=n this option can be used with the SORT statement to bypass 'n' records of the input file - the records are skipped prior to any decisions on record selection caused by an INCLUDE or OMIT statement, e.g.:

```
SORT FIELDS=COPY,SKIPREC=1000
SORT FIELDS=(1,5,CH,A),SKIPREC=250
```

STOPAFT=n this option can be used with the SORT statement to stop the input of records after 'n' - the decision to stop is made after considering record selection caused by an INCLUDE or OMIT statement, e.g.:

```
SORT FIELDS=COPY,STOPAFT=1000
SORT FIELDS=(1,5,CH,A),STOPAFT=250
```

Comments can be added to SyncSort jobs in several ways. The first is to place an asterisk in column one. This is a good way to either add some documentation to your job, or to tell SyncSort to ignore one or more lines of SyncSort statements.

Example 1 add some documentation

```
* this job extracts records for Albany County residents
INCLUDE COND=(46,2,CH,EQ,C'01')
```

Example 2 tell SyncSort to ignore a statement

```
* INCLUDE COND=(46,2,CH,EQ,C'01')
INCLUDE COND=(36,3,GE,C'065',AND,36,3,LE,C'084'),FORMAT=CH
SORT FIELDS=COPY
```

Another way to add comments is to leave some spaces at the end of a SyncSort statement before adding a comment.

Example 3 add another type of comment

```
INCLUDE COND=(46,2,EQ,C'01',AND, Albany County residents
              (36,3,GE,C'065',AND,
              36,3,LE,C'084')),FORMAT=CH age 65 through 84
SORT FIELDS=COPY
```

Example 4 use comments to show the layout of extracted data (refer to the description in Appendix D to see the record layout and positions of the data being read) - numbers and description on the right are the record layout and contents of the record copied with the SORT FIELDS=COPY statement

```
* new record layout      pos      contents
INREC FIELDS=(8,4,      1,4      hospital number
              46,2,      5,6      county of residence
              171,4)     7,10     length of stay
SORT FIELDS=COPY
```

Please contact me with any comments, suggestions,
etc. in any of the following ways:

PHONE (518)474-2079

FAX (518)473-2015

E-MAIL msz03@health.state.ny.us

MAIL Mike Zdeb
NYS Department of Health
Division of Family Health
ESP - Tower - Room 890
Albany, NY 12237-0657
