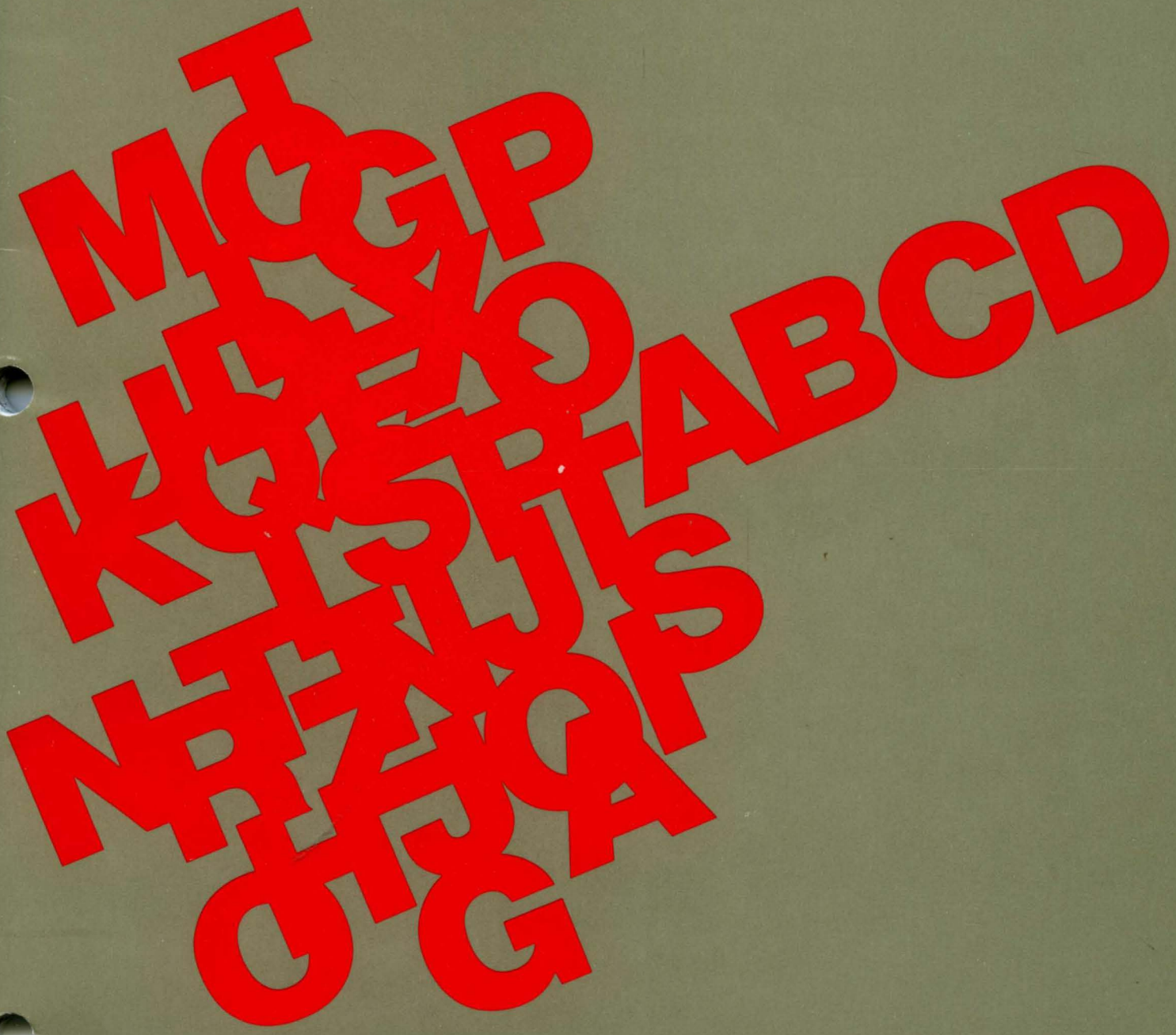
IBM

Getting Started with DFSORT

Program Product



Order Number:
SC26-4109-2

Program Number:
5740-SM1

Release Number:
8.0

Third Edition (March 1986)

This is a major revision of, and makes obsolete, SC26-4109-1.

This edition applies to Release 8.0 of IBM DFSORT, Program Product 5740-SM1, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

Because this book is primarily designed for new users of this product, specific changes normally indicated by a vertical bar to the left of the change do not appear in this edition.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

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

Preface

The objective of this book is to help you get started using the DFSORT program. It shows you step-by-step how to write sort, merge and copy applications, and teaches you some techniques for using DFSORT most efficiently.

Organization

This book has eleven chapters and two appendixes:

Chapter 1

"What is DFSORT?" gives an overview of what DFSORT can do.

Chapter 2

"Doing a Simple Sort" shows you how to do basic sort applications. For these applications and for those in Chapters 4 through 7 and 10, you will learn how to execute DFSORT by using job control language (JCL).

Chapter 3

"Tailoring the Input File" explains how to select from the input file only the records that are relevant to your application.

Chapter 4

"Summing Values in Records" explains how to sum the values in two or more records.

Chapter 5

"Reformatting Records Using OUTREC" shows you how to reformat records *after* they are processed (sorted, merged, or copied), for example, how to insert blanks between fields to make a printout more legible.

Chapter 6

"Reformatting Records Using INREC" explains how to reformat records *before* they are processed; for example, you will see how to eliminate unnecessary fields to make the processing more efficient.

Chapter 7

"Merging Files" explains how to merge presorted files.

Chapter 8

"Calling DFSORT from a Program" explains how to call DFSORT from a program written in PL/I or COBOL.

Chapter 9

"Overriding Installation Defaults" explains how to override defaults established when DFSORT was installed.

Chapter 10

"Copying Files" explains how to specify copy applications on the SORT, MERGE, or OPTION statement.

Chapter 11

"Using DFSORT Efficiently" explains ways to get the best performance from DFSORT.

Appendix A

"Sample File" illustrates a file used throughout the book as input for the examples. The illustration folds out so that you may easily refer to both the input file and the examples at the same time.

Appendix B

"Processing Order of Control Statements" contains a flowchart showing the order in which DFSORT control statements are processed.

Related Publications

For more detailed information on DFSORT, see

- *DFSORT Application Programming: Guide*, SC33-4035

For an explanation of DFSORT defaults, see

- *DFSORT Planning and Installation*, SC33-4034

For a quick reference, see

- *DFSORT Reference Summary*, SX33-8001

In addition, you may want to refer to your JCL reference manual and the programming guides for COBOL and PL/I.

Preface

Summary of Amendments

Release 8.0, March 1986

A main feature added to DFSORT that affects this manual is the ability to:

- Copy data sets without performing any sorting or merging operation. You can use COPY with most of the same control statements, exits, and options available when sorting or merging.

Release 7.1, June 1985

New features added to DFSORT that affect this manual are the ability to:

- Keep the first record when summarizing identically collating records when doing a sort or merge (see the chapter on "Summing Values in Records").
- Call DFSORT from VS COBOL II programs (see the chapter on "Calling DFSORT from a Program").
- Use the VS COBOL II FASTSRT compile time option, which enhances performance (see the chapters on "Calling DFSORT from a Program" and "Using DFSORT Efficiently").

Contents

Contents

Figures

1 What Is DFSORT?

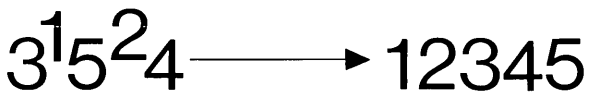
1 What Is DFSORT?

DFSORT (Data Facility Sort) is a program developed by IBM to help you organize information. You can use it for a task as simple as alphabetizing a list of names, or as an aid in complex tasks such as taking inventory or running a billing system. Through its record-level editing capability, DFSORT gives you the ability to perform data management functions at the record level.

Sorting Records

The primary function of DFSORT is to sort records. *Sorting* is arranging records in either ascending or descending order within a file.

Ascending Order



Descending Order

Records are sorted using either EBCDIC, the standard DFSORT collating sequence, or the ISCII/ASCII sequence established by the International Organization of Standards and the American National Standards Institute.

The EBCDIC and ISCII/ASCII collating sequences for the most common symbols are:

EBCDIC

Space
¢ . < (+ | & ! $ *) ; ¬
- / , % _ > ? : # @ ' = "
a through z
A through Z
0 through 9

ISCII/ASCII

Space
| " # $ % & ' ( ) * + , - . /
0 through 9
: ; < = > ? @
A through Z
[\ ] ^ _ `
a through z
{ | } ~

The data in the records can be in any of the formats used with the IBM System/370 (for example, EBCDIC character, ISCII/ASCII character, decimal, and binary).

1 What Is DFSORT?

While you are sorting, you can:

- Include or exclude certain records from the input file.
- Reformat records.
- Sum values in records.
- Alter the collating sequence.

Merging Records

Another major function of DFSORT is to merge records.

Merging is combining two or more files of sorted records to form a single file of sorted records.

While you are merging, you can:

- Include or exclude certain records from the input file.
- Reformat records.
- Sum values in records.
- Alter the collating sequence.

Copying Records

A third major function of DFSORT is to copy records without any sorting or merging taking place.

You can copy files in much the same way that you would sort or merge them.

While copying, you can:

- Include or exclude certain records from the input file.
- Reformat records.

What Do You Code?

You can do all of the above by coding a small set of DFSORT control statements that take the place of programs with complex logic. In fact, you can do a basic sort, merge, or copy by writing only one control statement (or, if calling DFSORT from a COBOL program, you can do a basic sort or merge by writing a COBOL SORT or MERGE statement).

You can execute DFSORT either by using the job control language (JCL) EXEC statement or by calling it from a program written in COBOL, PL/I, or assembler.

2 Doing a Simple Sort

2 Doing a Simple Sort

You can begin by doing a basic sort and using the JCL EXEC statement to execute it.

First of all, turn to the sample file in Appendix A. All the examples in this manual will refer to this file as input, so you should become familiar with it. (Note that the examples are for fixed-length records only; for information on processing variable-length records, see *DFSORT Application Programming: Guide*).

You will see that each record in the bookstore file has 12 fields (book title, author's last name, and so on).

To sort records, you choose one or more fields that you want ordered. These fields are called *control fields* (or, in COBOL, *keys*).

For example, if you choose the course department field as a control field and specify ascending order, the records will be sorted as shown in Figure 1. For the sake of brevity, not all the record fields are shown in Figure 1; however, notice that each *entire* record is sorted, not just the control field.

Also notice that records having equal control fields (in this case, having the same department) appear in their original order. For example, within the Computer Science department (COMP), the title *Video Game Design* still appears before *Computers: An Introduction.*

Whether records with equal control fields should appear in their original order or whether DFSORT may randomly order them can be determined at either installation time (as a default) or at execution time. In our examples, we'll assume that the default is for records with equal control fields to appear in their original order.

2 Doing a Simple Sort

	Book Title		Course Dept.		Price
BYTE	1	75	110	114	170 173
	LIVING WELL ON A SMALL BUDGET				9900
	PICK'S POCKET DICTIONARY				295
	INTRODUCTION TO BIOLOGY		BIOL		2350
	SUPPLYING THE DEMAND		BUSIN		1925
	STRATEGIC MARKETING		BUSIN		2350
	COMPUTER LANGUAGES		COMP		2600
	VIDEO GAME DESIGN		COMP		2199
	COMPUTERS: AN INTRODUCTION		COMP		1899
	NUMBERING SYSTEMS		COMP		360
	SYSTEM PROGRAMMING		COMP		3195
	INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL		595
	EDITING SOFTWARE MANUALS		ENGL		1450
	MODERN ANTHOLOGY OF WOMEN POETS		ENGL		450
	THE COMPLETE PROOFREADER		ENGL		625
	SHORT STORIES AND TALL TALES		ENGL		1520
	THE INDUSTRIAL REVOLUTION		HIST		795
	EIGHTEENTH CENTURY EUROPE		HIST		1790
	CRISES OF THE MIDDLE AGES		HIST		1200
	INTRODUCTION TO PSYCHOLOGY		PSYCH		2200
	ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH		2600

Figure 1. Sort by Course Department in Ascending Order

Writing the SORT Statement

When executing DFSORT with the JCL EXEC statement, you describe the control fields, and the order in which you want them sorted, by using a SORT control statement.

To write a SORT statement that sorts the bookstore records by the department field (as shown in Figure 1), you:

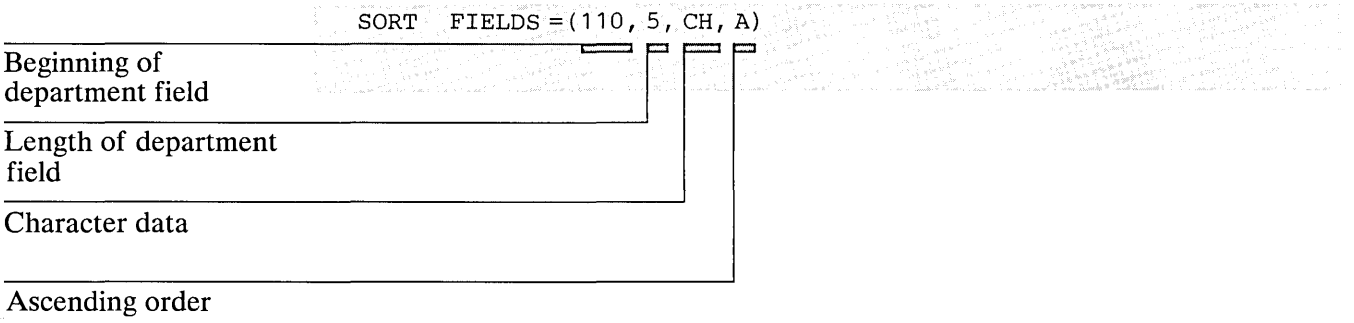
- Leave at least one blank, and write SORT.
- Leave at least one blank and write FIELDS=.

- Write, in parentheses, and separated by commas:
  - Where the department field begins, relative to the beginning of the record (the first position is byte 1). The department field begins at byte 110.
  - The length of the department field in bytes. The department field is 5 bytes long.
  - A code for the data format. The department field contains character data, which you specify as CH.

    (Figure 2 shows the codes for the most common data formats).

  - The letter A, which means you want ascending order.

Note: Make sure that the statement is coded between columns 2 through 71.



The most common data formats and their codes are:

Data Format	Code
EBCDIC Character	CH
ISCII/ASCII Character	AC
Binary	BI
Zoned Decimal	ZD
Packed Decimal	PD

Figure 2. Data Format Codes

Doing a Simple Sort

Sorting by Multiple Fields

Within each department, you can further sort the records, by specifying more control fields.

When you specify two or more control fields, you specify them in the order of greater to lesser priority.

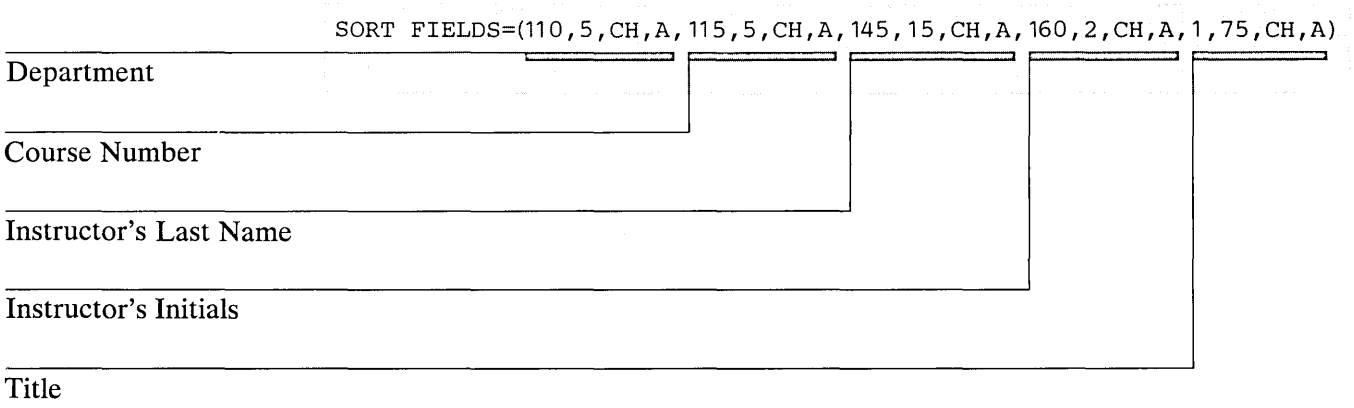
Note: Control fields may overlap or be contained within other control fields.

Figure 3 shows how the records would be sorted if you specified the following control fields in the order they are listed:

- Course department
- Course number
- Instructor's last name
- Instructor's initials
- Book title

So, if two records have the same course department, they are sorted by course number; if they also have the same course number, they are sorted by instructor's last name; if they also have the same last name, they are sorted by initials; and, if they also have the same initials, they are sorted by title.

To write a SORT statement that sorts the records as they are shown in Figure 3, specify the location, length, data format, and order for each of the control fields, as follows:



Book Title	Course Dept.	Course No.	Instr. Last Name	Instr. Init.	Price
BYTE 1 75	110 114	115 119	145 159	160 161	170 173
LIVING WELL ON A SMALL BUDGET					9900
PICK'S POCKET DICTIONARY					295
INTRODUCTION TO BIOLOGY	BIOL	80521	GREENBERG	HC	2350
STRATEGIC MARKETING	BUSIN	70124	LORCH	MH	2350
SUPPLYING THE DEMAND	BUSIN	70251	MAXWELL	RF	1925
NUMBERING SYSTEMS	COMP	00032	CHATTERJEE	AN	360
COMPUTER LANGUAGES	COMP	00032	CHATTERJEE	CL	2600
COMPUTERS: AN INTRODUCTION	COMP	00032	CHATTERJEE	CL	1899
SYSTEM PROGRAMMING	COMP	00103	SMITH	DC	3195
VIDEO GAME DESIGN	COMP	00205	NEUMANN	LB	2199
SHORT STORIES AND TALL TALES	ENGL	10054	BUCK	GR	1520
EDITING SOFTWARE MANUALS	ENGL	10347	MENDOZA	VR	1450
THE COMPLETE PROOFREADER	ENGL	10347	MENDOZA	VR	625
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	ENGL	10856	FRIEDMAN	KR	595
MODERN ANTHOLOGY OF WOMEN POETS	ENGL	10856	FRIEDMAN	KR	450
THE INDUSTRIAL REVOLUTION	HIST	50420	GOODGOLD	ST	795
CRISES OF THE MIDDLE AGES	HIST	50521	WILLERTON	DW	1200
EIGHTEENTH CENTURY EUROPE	HIST	50632	BISCARDI	HR	1790
INTRODUCTION TO PSYCHOLOGY	PSYCH	30016	ZABOSKI	RL	2200
ADVANCED TOPICS IN PSYCHOANALYSIS	PSYCH	30975	NAKATSU	FL	2600

Figure 3. Sort by Multiple Fields

You can shorten this last statement and still achieve the same result, by specifying the department and course number together as one field, and the instructor's last name and initials together as one field. You can specify fields together whenever they are next to each other and have the same data format.

```
SORT FIELDS=(110,10,CH,A,145,17,CH,A,1,75,CH,A)
```

Department and Course Number

Instructor's Last Name and Initials

Title

Also, if all the control fields have the same type of data format, you can specify the data format just once, using the FORMAT= parameter, like this:

```
SORT FIELDS=(110,10,A,145,17,A,1,75,A),FORMAT=CH
```

Continuing a Statement

If you can't fit your SORT statement (or any other DFSORT statement that we'll discuss later) between columns 2 through 71, you can continue it on the next line. If you end a line with a comma followed by a blank, DFSORT will assume that the next line is a continuation.

The continuation can begin anywhere between columns 2 through 71.

For example:

```
SORT FIELDS=(110,10,A,145,17,A,
             1,75,A),FORMAT=CH
```

Sorting in Descending Order

To sort the records in descending order, specify D instead of A. You can have some control fields in descending order and others in ascending order. For example, to sort the departments in ascending order and the prices for each department in descending order, you write:

```
SORT FIELDS=(110,5,CH,A,170,4,BI,D)
```

Department

Price

Figure 4 shows the result.

	Book Title	Course Dept.	Price
BYTE	1 75	110 114	170 173
	LIVING WELL ON A SMALL BUDGET		9900
	PICK'S POCKET DICTIONARY		295
	INTRODUCTION TO BIOLOGY	BIOL	2350
	STRATEGIC MARKETING	BUSIN	2350
	SUPPLYING THE DEMAND	BUSIN	1925
	SYSTEM PROGRAMMING	COMP	3195
	COMPUTER LANGUAGES	COMP	2600
	VIDEO GAME DESIGN	COMP	2199
	COMPUTERS: AN INTRODUCTION	COMP	1899
	NUMBERING SYSTEMS	COMP	360
	SHORT STORIES AND TALL TALES	ENGL	1520
	EDITING SOFTWARE MANUALS	ENGL	1450
	THE COMPLETE PROOFREADER	ENGL	625
	INKLINGS: AN ANTHOLOGY OF YOUNG POETS	ENGL	595
	MODERN ANTHOLOGY OF WOMEN POETS	ENGL	450
	EIGHTEENTH CENTURY EUROPE	HIST	1790
	CRISES OF THE MIDDLE AGES	HIST	1200
	THE INDUSTRIAL REVOLUTION	HIST	795
	ADVANCED TOPICS IN PSYCHOANALYSIS	PSYCH	2600
	INTRODUCTION TO PSYCHOLOGY	PSYCH	2200

Figure 4. Sort by Department in Ascending Order and Price in Descending Order

2 Doing a Simple Sort

Writing the JCL

The JCL you need to do a sort depends on whether you execute DFSORT with the JCL EXEC statement or call DFSORT from a program. For now, we'll limit our discussion to executing DFSORT with the JCL EXEC statement.

The JCL statements you need are described below.

//jobname JOB

The JOB statement signals the beginning of a job. At your installation, you may be required to specify information such as your name and account number on the JOB statement.

//stepname EXEC

The EXEC statement signals the beginning of a job step and tells the operating system what program to execute. To execute DFSORT, write the EXEC statement like this:

```
//stepname  EXEC  PGM=SORT
```

//STEPLIB DD

The STEPLIB DD statement defines the library containing the DFSORT program. If your DFSORT program is in a system library, you can omit the STEPLIB statement.

//SYSOUT DD

The SYSOUT DD statement defines the output data set for messages.

//SORTIN DD

The SORTIN DD statement defines the input data set.

//SORTWKnn DD

The SORTWKnn DD statement defines a work storage data set. For most applications, one work storage data set is sufficient. (Increasing the number of work storage data sets does *not* improve performance.)

//SORTOUT DD

The SORTOUT DD statement defines the output data set.

//SYSIN DD

The SYSIN DD statement precedes the DFSORT statements.

2 Doing a Simple Sort

Below is some sample JCL that will execute DFSORT. It is assumed that the input data set is cataloged and that the output data set will be cataloged. It is also assumed that the input and output record lengths are the same. Later, in Chapter 5, we'll show you how to modify the SORTOUT DD statement if the record length is changed.

```
//EXAMP    JOB  A492,PROGRAMMER
//SORT     EXEC PGM=SORT
//STEPLIB  DD   DSN=A492.SM,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//SORTIN   DD   DSN=BOOKS.INPUT,DISP=OLD
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTOUT  DD   DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(1,1)),UNIT=SYSDA
//SYSIN    DD   *
    SORT  FIELDS=(110,10,A,145,17,A,1,75,A),
          FORMAT=CH
/*
```

For more detailed information about executing DFSORT with JCL, see *DFSORT Application Programming: Guide.*

3 Tailoring the Input File

Often, only a subset of the records in a file is needed for an application. This chapter explains how to tailor the input file, selecting only the records that you need.

By tailoring the file, you can increase the speed of the sort, because the unneeded records are deleted before the sort begins; the fewer the records, the less time it takes to sort them.

You tailor an input file by either:

- Specifying on an INCLUDE control statement the records you want included, or
- Specifying on an OMIT control statement the records you want omitted.

Your choice of the INCLUDE or the OMIT statement depends on which is easier and more efficient to write for a given application. *You may not use both statements together.*

You select the records you want included or omitted by comparing the contents of a record field with either:

- Another field (for example, you can select records for which the author's last name is the same as the instructor's last name); or,
- A constant. The constant may be a character string, a decimal number, or a hexadecimal string (for example, you can select records that have the character string "HIST" in the department field).

You may also have two or more conditions combined by logical ANDs and ORs. For example, you can select records that have either "HIST" or "PSYCH" in the department field.

Note: DFSORT follows these rules for padding and truncation:

- In a field-to-field comparison, the shorter field is padded as appropriate (with blanks or zeros).
- In a field-to-constant comparison, the constant is padded or truncated to the length of the field. Decimal constants are padded or truncated on the left; character and hexadecimal constants are padded or truncated on the right.

Writing the INCLUDE Statement

Suppose that it's the end of the year, and you want to sort by title *only the books that you need to order more copies for the coming year.* If the number of copies sold this year for a particular book is greater than the number in stock, you can assume you need to order more copies.

To write an INCLUDE statement that selects only the books you need to order:

- Leave at least one blank and write INCLUDE.
- Leave at least one blank and write COND=.
- Write, in parentheses, and separated by commas:
  - The location, length, and data format of the number sold field.
  - The comparison operator GT (comparison operators are shown in Figure 5).
  - The location, length, and data format of the number in stock field.

(You can use FORMAT= when fields have the same data format.)

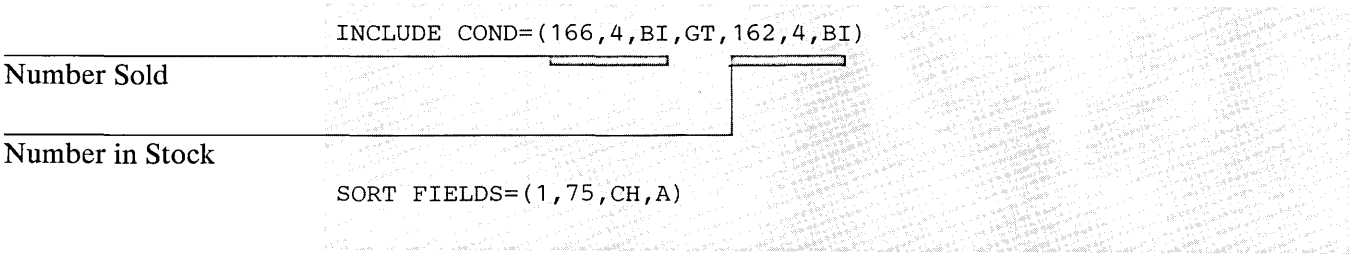
You may select from the following comparison operators:

EQ	Equal to
NE	Not equal to
GT	Greater than
GE	Greater than or equal to
LT	Less than
LE	Less than or equal to

Figure 5. Comparison Operators

You then sort the tailored file by title in ascending order by using the SORT statement.

You can place the SORT statement either before or after the INCLUDE statement. Control statements don't have to be in any specific order; however, it is good documentation practice to code them in the order in which they are processed. For a flowchart showing the order in which all the control statements are processed, see Appendix B.



The sorted file is shown in Figure 6.

	Book Title	No. in Stock	No. Sold Y-to-D	Price
BYTE	1 75	162 165	166 169	170 173
	ADVANCED TOPICS IN PSYCHOANALYSIS	1	12	2600
	COMPUTER LANGUAGES	5	29	2600
	COMPUTERS: AN INTRODUCTION	20	26	1899
	CRISES OF THE MIDDLE AGES	14	17	1200
	EDITING SOFTWARE MANUALS	13	32	1450
	INKLINGS: AN ANTHOLOGY OF YOUNG POETS	2	32	595
	INTRODUCTION TO BIOLOGY	6	11	2350
	MODERN ANTHOLOGY OF WOMEN POETS	1	26	450
	NUMBERING SYSTEMS	6	27	360
	STRATEGIC MARKETING	3	35	2350
	SUPPLYING THE DEMAND	0	32	1925
	SYSTEM PROGRAMMING	4	23	3195
	THE COMPLETE PROOFREADER	7	19	625

Figure 6. Books for which Number Sold Is Greater than Number in Stock

Now, suppose you want to tailor the input file even further, to sort only the books you need to order from COR Publishers. In this case, two conditions must be true: 1) the number sold is greater than the number in stock, and 2) the book is published by COR. To add this second condition, you can expand the above INCLUDE statement by adding a logical AND, and comparing the contents of the publisher field to the character string "COR" (the last section in this chapter shows how to specify constants). Because the publisher field is four bytes long, "COR" will be padded on the right with one blank.

```
INCLUDE COND=(166,4,BI,GT,162,4,BI,AND,106,4,CH,EQ,C'COR')
SORT FIELDS=(1,75,CH,A)
```

The sorted file is shown in Figure 7.

Book Title		Publisher		No. in Stock		No. Sold Y-to-D		Price	
BYTE 1	75	106	109	162	165	166	169	170	173
CRISES OF THE MIDDLE AGES		COR			14		17		1200
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		COR			2		32		595
MODERN ANTHOLOGY OF WOMEN POETS		COR			1		26		450
SUPPLYING THE DEMAND		COR			0		32		1925

Figure 7. Books by COR for which Number Sold Is Greater than Number In Stock

As another example, you can sort only the books for courses 00032 and 10347 by writing the INCLUDE and SORT statements as follows:

```
INCLUDE COND=(115,5,CH,EQ,C'00032',OR,115,5,CH,EQ,C'10347')
SORT FIELDS=(115,5,CH,A)
```

Note: Be aware of the rules for padding and truncation of constants and fields. In the above example, you cannot substitute C'32' for C'00032', because character constants are padded on the right with blanks.

Writing the OMIT Statement

Suppose that you want to sort by title all the books used for courses, but not those for general reading. In this case, you can write an OMIT statement that excludes records containing a blank in the course department field.

The format of the OMIT statement is the same as that of the INCLUDE statement. So, to exclude the general reading books, you write:

```
OMIT COND=(110,5,CH,EQ,C' ')
SORT FIELDS=(1,75,CH,A)
```

The sorted file is shown in Figure 8.

BYTE

Book Title 1 75	Course Dept. 110 114	Price 170 173
ADVANCED TOPICS IN PSYCHOANALYSIS	PSYCH	2600
COMPUTER LANGUAGES	COMP	2600
COMPUTERS: AN INTRODUCTION	COMP	1899
CRISES OF THE MIDDLE AGES	HIST	1200
EDITING SOFTWARE MANUALS	ENGL	1450
EIGHTEENTH CENTURY EUROPE	HIST	1790
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	ENGL	595
INTRODUCTION TO BIOLOGY	BIOL	2350
INTRODUCTION TO PSYCHOLOGY	PSYCH	2200
MODERN ANTHOLOGY OF WOMEN POETS	ENGL	450
NUMBERING SYSTEMS	COMP	360
SHORT STORIES AND TALL TALES	ENGL	1520
STRATEGIC MARKETING	BUSIN	2350
SUPPLYING THE DEMAND	BUSIN	1925
SYSTEM PROGRAMMING	COMP	3195
THE COMPLETE PROOFREADER	ENGL	625
THE INDUSTRIAL REVOLUTION	HIST	795
VIDEO GAME DESIGN	COMP	2199

Figure 8. Sort, Omitting Books Not Required for Classes

Allowable Comparisons for INCLUDE and OMIT

Figures 9 and 10 show the allowable field-to-field and field-to-constant comparisons for INCLUDE and OMIT.

Field Format	BI	CH	ZD	PD	AC
BI	X	X			
CH	X	X			
ZD			X	X	
PD			X	X	
AC					X

Figure 9. Allowable Field-to-Field Comparisons

Field Format	Character String	Hexadecimal String	Decimal Number
BI	X	X	
CH	X	X	
ZD			X
PD			X
AC	X	X	

Figure 10. Allowable Field-to-Constant Comparisons

For example, if you want to sort by author's last name and include only those books whose author's last name begins with "M", you can compare the contents of byte 76 (the first byte of the author's last name), which is in character format, with either a character or hexadecimal string:

```
INCLUDE COND=(76,1,CH,EQ,C'M')
SORT FIELDS=(76,15,CH,A)
```

or

```
INCLUDE COND=(76,1,CH,EQ,X'D4')
SORT FIELDS=(76,15,CH,A)
```

Another example: If you want to sort by number in stock only the books for which the number in stock is less than 10, you can compare the contents of the number in the stock field, which is in binary format, to a hexadecimal string:

```
INCLUDE COND=(162,4,BI,LT,X'0000000A'
SORT FIELDS=(162,4,BI,A)
```

Again, remember the padding and truncation rules. If you specified X'0A', the string would be padded on the right instead of the left.

Formats for Writing Constants

The formats for writing character strings, hexadecimal strings, and decimal numbers are shown below.

Character Strings

The format for writing a character string is:

C'x...x'

where x is an EBCDIC character. For example, C'FERN'.

If you want to include a single apostrophe in the string, you must specify it as two single apostrophes. For example, *O'NEILL* must be specified as C'O''NEILL'.

Hexadecimal Strings

The format for writing a hexadecimal string is:

X'yy...yy'

where yy is a pair of hexadecimal digits. For example, X'7FB0'.

Decimal Numbers

The format for writing a decimal number is:

n...n **or** ±n...n

where n is a decimal digit. Examples are 24, +24, and -24.

Decimal numbers must not contain commas or decimal points.

4 Summarizing Values in Records

4 Summing Values in Records

Suppose that the English department wants to know the total price of books for all its courses.

You can tailor the file to include only the English department's records by using the INCLUDE statement, and sum the book prices by using the SORT and SUM statements.

On the SUM control statement, you specify one or more numeric fields that are to be summed whenever records have equal control fields (control fields are specified on the SORT statement). The numeric fields can be in binary, packed decimal, or zoned decimal format.

So, to sum the prices for all English department's records, you specify the price field on the SUM statement and the department field on the SORT statement. By the time SUM and SORT are processed, INCLUDE will have tailored the file to contain only the English department's records, making the department field equal for all the records, and allowing the prices to be summed. (For a flowchart showing the order in which the INCLUDE, SUM, and SORT statements are processed, see Appendix B.)

When you sum records, keep in mind that two types of fields are involved:

- *Control fields*, which are specified on the SORT statement, and
- *Summary fields*, which are specified on the SUM statement.

The contents of the summary fields are summed only when the contents of the control fields are equal.

Writing the SUM Statement

To write a SUM statement that sums the prices for the English department:

- Leave at least one blank and write SUM.
- Leave at least one blank and write FIELDS=.
- Write, in parentheses, and separated by commas:
  - The location, length, and data format of the price field.

The INCLUDE, SORT, and SUM statements are shown below:

```
INCLUDE COND=(110,5,CH,EQ,C'ENGL')
SORT FIELDS=(110,5,CH,A)
SUM FIELDS=(170,4,BI)
```

Price

4 Summing Values in Records

When the prices are summed, the final sum is placed in the price field of one record, and the other records are deleted. Therefore, the result (shown in Figure 11) is only one record, containing the sum. Whether you know which record will be kept depends on whether you specified that records should keep their original order. Remember that, for our examples, we assumed that the installation default is for records with equal control fields to appear in their original order. When summing records keeping the original order, DFSORT chooses the first record to contain the sum.

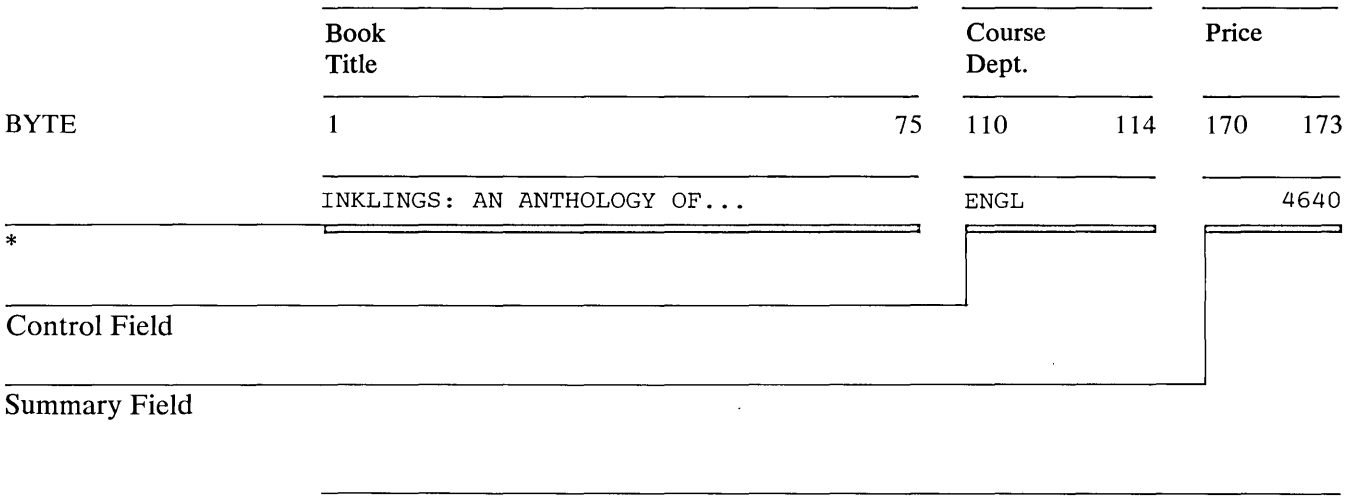


Figure 11. Sum of Prices for English Department

* Some of the fields in your summation record may not be meaningful, such as the book title field in Figure 11. In the next chapter, you'll discover how to omit the fields that are not meaningful.

4 Summing Values in Records

Let's suppose now that the English department wants to know the total price of books for *each* of its courses. In this case, you still select only the English department's records using INCLUDE, and specify the price field on the SUM statement, but you specify the *course number* on the SORT statement.

```
INCLUDE COND=(110,5,CH,EQ,C'ENGL')
SORT FIELDS=(115,5,CH,A)
SUM FIELDS=(170,4,BI)
```

Price

The result, one record per course, is shown in Figure 12.

Book Title		Course No.		Price	
BYTE 1	75	115	119	170	173
SHORT STORIES AND TALL TALES			10054		1520
EDITING SOFTWARE MANUALS			10347		2075
INKLINGS: AN ANTHOLOGY OF...			10856		1045

Figure 12. Sum of Prices for Each English Class

4 Summing Values in Records

For an example using two summary fields, assume that for inventory purposes you want to sum separately the number of books in stock and the number sold for each of the publishers.

For this application, specify the publisher as the control field on the SORT statement, and the number in stock and number sold as summary fields on the SUM statement.

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,166,4),FORMAT=BI
```

Number in Stock

Number Sold

The result, one record per publisher, is shown in Figure 13.

Book Title	Publisher	No. in Stock	No. Sold Y-to-D	Price
BYTE 1 75	106 109	162 165	166 169	170 173
LIVING WELL ON A . . .	COR	103	161	9900
COMPUTER LANGUAGES	FERN	19	87	2600
VIDEO GAME DESIGN	VALD	42	97	2199
COMPUTERS: AN INTRO. . .	WETH	62	79	1899

Figure 13. Sums of Number in Stock and Number Sold for Each Publisher

Summing Values in Records

Overflow

When a sum becomes larger than the space available for it, *overflow* occurs. For example, if you have a 2-byte binary field (unsigned) containing X'FFFF' and you add X'0001' to it, overflow will occur, because the sum requires more than two bytes.

```
 FFFF
 0001
-----
10000
```

If overflow occurs, the two records involved (two records are summed at a time) will be left unsummarized; that is, their contents will be left undisturbed, and neither record will be deleted. However, other records will continue to be summarized. If overflow occurs, DFSORT will give you a warning message.

In some cases, you can correct overflow by padding the summary fields with zeros, using the INREC control statement. You'll learn how to do this in Chapter 6.

Deleting Records with Duplicate Control Fields

Apart from summing values, another function of SUM is to delete records with duplicate control fields.

For example, you may want to list the publishers in ascending order, but you want each publisher to appear only once. If you used only the SORT statement, COR would appear seven times (because seven books in the file are published by COR), FERN would appear four times, VALD five times, and WETH four times.

By specifying FIELDS=NONE on the SUM statement, as shown below, only one record per publisher will be written.

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=NONE
```

The result is shown in Figure 14.

4 Summing Values in Records

Book Title		Publisher		Price	
BYTE	1 75	106	109	170	173
LIVING WELL ON A SMALL BUDGET		COR			9900
COMPUTER LANGUAGES		FERN			2600
VIDEO GAME DESIGN		VALD			2199
COMPUTERS: AN INTRODUCTION		WETH			1899

Figure 14. List of Publishers, Deleting Duplicates

5 Reformatting Records Using OUTREC

After the records are sorted and before they are written, you can reformat them by using the OUTREC control statement.

Using OUTREC, you can:

- Delete fields
- Rearrange the order of fields
- Insert zeros before, between, or after fields
- Insert blanks before, between, or after fields

Note: If you use OUTREC to change the record length (by deleting fields or inserting blanks), be sure to specify the new record length on the SORTOUT DD statement (using the DCB parameter).

Writing the OUTREC Statement

Deleting Fields

In the last chapter, you used the SUM statement to sum the books in stock and the books sold for each publisher. Now, using the OUTREC statement, you can delete all the fields that aren't needed for the application (fields whose contents are not meaningful in a summation record). Only the publisher, number in stock, and number sold fields will be written, reducing the output record length to 12 bytes.

To write the OUTREC statement, you:

- Leave at least one blank and write OUTREC.
- Leave at least one blank and write FIELDS=.
- Write, in parentheses, and separated by commas:
  - The location and length of the publisher field.
  - The location and length of the number in stock field.
  - The location and length of the number sold field.

Because the number in stock and number sold fields are next to each other, you can also specify them together as one field (they need not have the same data format).

Note that on this statement you do *not* specify the data format.

Reformatting Records Using OUTREC

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,162,4,166,4)
```

Publisher

Number in Stock

Number Sold

Or:

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,162,8)
```

Publisher

Number in Stock
and Number Sold

Because the record length was changed, the new length must be specified on the SORTOUT DD statement. For example:

```
//SORTOUT DD DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(1,1),UNIT=SYSDA,
//          DCB=(LRECL=12,BLKSIZE=12,RECFM=F)
```

The output is shown in Figure 15.

	Publisher	No. in Stock	No. Sold Y-to-D
BYTE	1 4	5 8	9 12
	COR	103	161
	FERN	19	87
	VALD	42	97
	WETH	62	79

Figure 15. Writing Only Publisher, Number in Stock, and Number Sold Fields

Reformatting Records Using OUTREC

Reordering Fields

The fields always appear in the order in which you specify them. So, if you wanted the number sold to appear before the number in stock, as shown in Figure 16, you would merely reverse their order on the OUTREC statement.

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,166,4,162,4)
```

	Publisher	No. Sold Y-to-D	No. in Stock
BYTE	1 4	5 8	9 12
	COR	161	103
	FERN	87	19
	VALD	97	42
	WETH	79	62

Figure 16. Reordering the Fields

Inserting Binary Zeros

Building on the last example, assume you want to reformat the records to include a new 4-byte binary field after the number in stock (beginning at byte 13). In this case, you could insert binary zeros as place holders for the new field (to be filled in with data at a later date).

To insert the zeros, write 4Z after the last field, as shown below.

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,166,4,162,4,4Z)
```

This time, you must specify on the SORTOUT DD statement that the new record length is 16 bytes:

```
//SORTOUT DD DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(1,1),UNIT=SYSDA,
//           DCB=(LRECL=16,BLKSIZE=16,RECFM=F)
```

The result is shown in Figure 17.

Reformatting Records Using OUTREC

	Publisher	No. Sold Y-to-D	No. in Stock	X'0...0'
BYTE	1 4	5 8	9 12	13 16
	COR	161	103	0...0
	FERN	87	19	0...0
	VALD	97	42	0...0
	WETH	79	62	0...0

Figure 17. Inserting Binary Zeros

You can insert binary zeros before, between, or after fields.

Inserting Blanks

If an output data set contains *only character data*, you can print it by writing the SORTOUT DD statement, as follows:

```
//SORTOUT DD SYSOUT=A
```

You can make the printout more legible by using the OUTREC statement to separate the fields with blanks and to create a margin.

For example, assume you want to print just the publisher and title fields, and want the publisher field to appear first. Because most of the publishers' names fill up the entire 4-byte publisher field, if you don't separate the two fields with blanks, the publishers' names will run into the titles; also, without a margin, the publishers' names will begin at the edge of the paper.

The printout can be made more legible by separating the fields with 10 blanks and creating a margin of 20 blanks.

To insert the blanks, write 10X between the two fields, and 20X before the first field, as shown below. The SORT statement sorts the records by title in ascending order (remember that SORT or MERGE is always required).

```
SORT FIELDS=(1,75,CH,A)
OUTREC FIELDS=(20X,106,4,10X,1,75)
```

The output is shown in Figure 18.

```
FERN        ADVANCED TOPICS IN PSYCHOANALYSIS
FERN        COMPUTER LANGUAGES
WETH        COMPUTERS: AN INTRODUCTION
COR         CRISES OF THE MIDDLE AGES
VALD        EDITING SOFTWARE MANUALS
WETH        EIGHTEENTH CENTURY EUROPE
COR         INKLINGS: AN ANTHOLOGY OF YOUNG POETS
VALD        INTRODUCTION TO BIOLOGY
COR         INTRODUCTION TO PSYCHOLOGY
COR         LIVING WELL ON A SMALL BUDGET
COR         MODERN ANTHOLOGY OF WOMEN POETS
FERN        NUMBERING SYSTEMS
COR         PICK'S POCKET DICTIONARY
VALD        SHORT STORIES AND TALL TALES
VALD        STRATEGIC MARKETING
COR         SUPPLYING THE DEMAND
WETH        SYSTEM PROGRAMMING
FERN        THE COMPLETE PROOFREADER
WETH        THE INDUSTRIAL REVOLUTION
VALD        VIDEO GAME DESIGN
```

20 Blanks

10 Blanks

Figure 18. Output After Inserting Blanks

You can insert blanks before, between, or after fields.

6 Reformatting Records Using INREC

6 Reformatting Records Using INREC

You have just discovered how to reformat records using the OUTREC control statement. Another way to reformat records is to use the INREC control statement. With INREC, you can also delete fields, insert blanks or zeros, and reorder fields.

The difference between OUTREC and INREC is that, whereas OUTREC reformats records *after* they are sorted, INREC reformats them *before* they are sorted.

Because shorter records take less time to sort, you should generally use INREC to delete fields, and OUTREC to insert blanks or zeros. However, both these functions are provided on each statement for flexibility. Because reordering fields doesn't affect record length, it doesn't matter which statement you use for this function.

Note: Be sure to specify the *final* record length on the SORTOUT DD statement (that is, the length after INREC if you are using just INREC, or the length after OUTREC if you are using just OUTREC or both INREC and OUTREC).

Writing the INREC Statement

The INREC statement has the same format as the OUTREC statement. So, in the first example of Chapter 5, where you used OUTREC to write only the publisher, number in stock, and number sold fields, you could use INREC instead, as shown below.

```
INREC FIELDS=(106,4,162,4,166,4)
```

Publisher

Number in Stock

Number Sold

Or:

```
INREC FIELDS=(106,4,162,8)
```

Publisher

Number in Stock and
Number Sold

Reformatting Records Using INREC

Writing Other Statements with INREC

Because INREC reformats the records *before* they are sorted, the SORT and SUM statements must refer to the *reformatted* rather than the original records.

Thus, after INREC, the input records are 12 bytes long (see Figure 19):

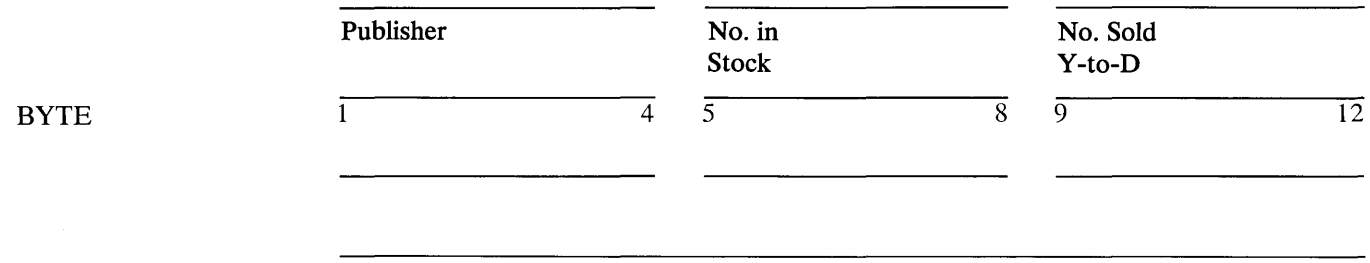


Figure 19. Input Records After INREC

You write the SORT and SUM statements like this:

```
SORT FIELDS=(1,4,CH,A)
SUM FIELDS=(5,4,BI,9,4,BI)
```

The final result is shown in Figure 20.

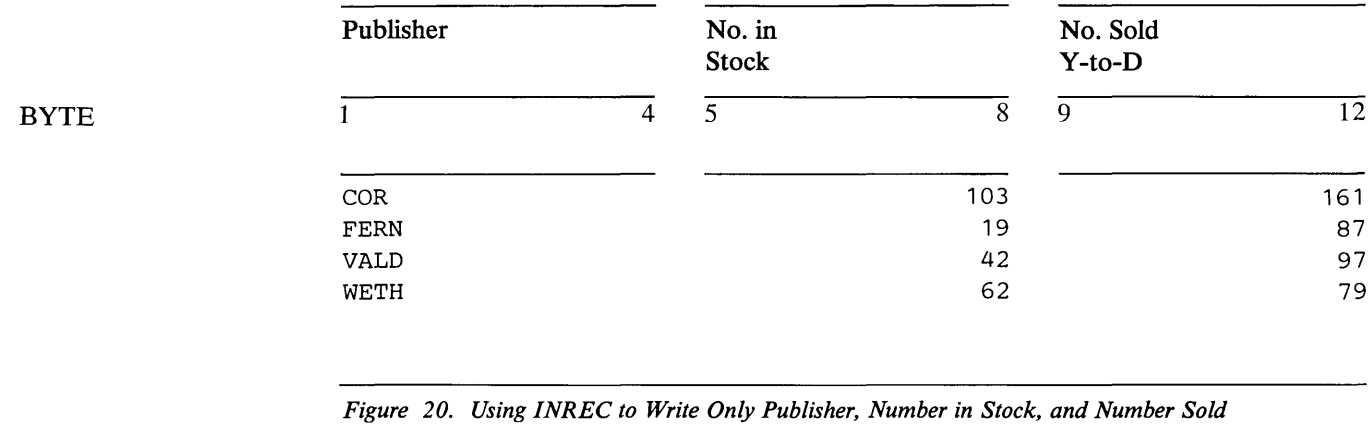


Figure 20. Using INREC to Write Only Publisher, Number in Stock, and Number Sold

6 Reformatting Records Using INREC

If you turn to the flowchart in Appendix B, you'll see that the INREC statement is processed *before* SORT, SUM, and OUTREC, but *after* INCLUDE and OMIT. Therefore, when using the INREC statement, SORT, SUM, and OUTREC must refer to the *reformatted* records, and INCLUDE and OMIT must refer to the *original* records.

Note: The flowchart in Appendix B also includes MERGE and OPTION COPY in the processing order.

Preventing Overflow When Summing Values

In some cases, you can prevent overflow by using INREC to pad summary fields with zeros. However, this method can't be used for negative fixed-point binary data, because padding with zeros rather than with ones would change the sign.

If the summary fields in the last example were overflowing, you could pad each of them on the *left* with four bytes (binary fields must be two, four, or eight bytes long), as shown in Figure 21.

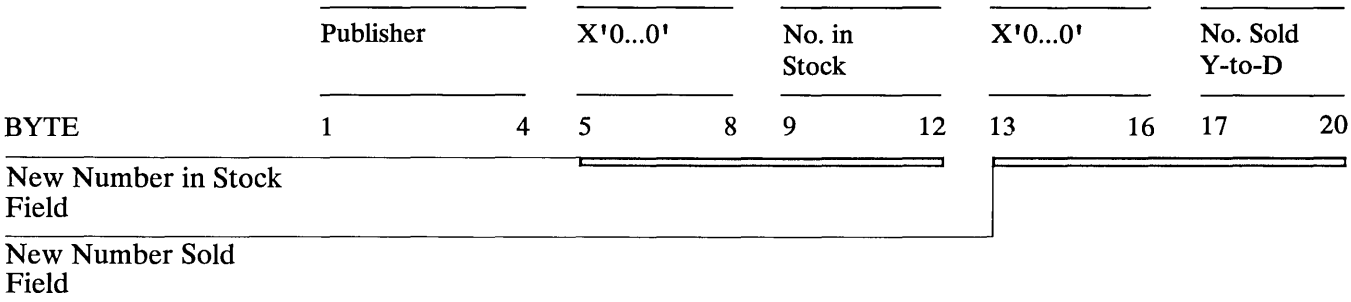


Figure 21. Padding Summary Fields

```
INREC FIELDS=(106,4,4Z,162,4,4Z,166,4)
SORT FIELDS=(1,4,CH,A)
SUM FIELDS=(5,8,BI,13,8,BI)
```

New Number in Stock Field

New Number Sold Field

The output records, each 20 bytes long, are shown in Figure 22.

	Publisher	New No. in Stock	New No. Sold Y-to-D
BYTE	1 4	5 12	13 20
	COR	103	161
	FERN	19	87
	VALD	42	97
	WETH	62	79

Figure 22. New Number in Stock and Number Sold Fields

Note that you cannot use OUTREC to prevent overflow, because it is processed *after* summarization.

7 Merging Files

Generally, the reason for merging files is to add more records to a file that is already sorted.

For example, assume that the bookstore file is already sorted by department and title (see Figure 23), and you want to update it by merging it with a file that contains five new records, also sorted by department and title (see Figure 24). Figure 25 shows the resulting file, which is now the updated bookstore file.

	Book Title		Course Dept.		Price	
BYTE	1	75	110	114	170	173
	LIVING WELL ON A SMALL BUDGET					9900
	PICK'S POCKET DICTIONARY					295
	INTRODUCTION TO BIOLOGY		BIOL			2350
	STRATEGIC MARKETING		BUSIN			2350
	SUPPLYING THE DEMAND		BUSIN			1925
	COMPUTER LANGUAGES		COMP			2600
	COMPUTERS: AN INTRODUCTION		COMP			1899
	NUMBERING SYSTEMS		COMP			360
	SYSTEM PROGRAMMING		COMP			3195
	VIDEO GAME DESIGN		COMP			2199
	EDITING SOFTWARE MANUALS		ENGL			1450
	INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL			595
	MODERN ANTHOLOGY OF WOMEN POETS		ENGL			450
	SHORT STORIES AND TALL TALES		ENGL			1520
	THE COMPLETE PROOFREADER		ENGL			625
	CRISES OF THE MIDDLE AGES		HIST			1200
	EIGHTEENTH CENTURY EUROPE		HIST			1790
	THE INDUSTRIAL REVOLUTION		HIST			795
	ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH			2600
	INTRODUCTION TO PSYCHOLOGY		PSYCH			2200

Figure 23. Bookstore File Sorted by Department and Title

	Book Title		Course Dept.		Price	
BYTE	1	75	110	114	170	173
	INTERNATIONAL COOKBOOK					1450
	WORLD JOURNEYS BY TRAIN					1099
	ARTS AND CRAFTS OF ASIA		ART			1545
	BIOCHEMISTRY		BIOL			2150
	BEHAVIORAL ANALYSIS		PSYCH			1060

Figure 24. Five New Records Sorted by Department and Title

	Book Title		Course Dept.		Price	
BYTE	1	75	110	114	170	173
	INTERNATIONAL COOKBOOK					1450
	LIVING WELL ON A SMALL BUDGET					9900
	PICK'S POCKET DICTIONARY					295
	WORLD JOURNEYS BY TRAIN					1099
	ARTS AND CRAFTS OF ASIA		ART			1545
	BIOCHEMISTRY		BIOL			2150
	INTRODUCTION TO BIOLOGY		BIOL			2350
	STRATEGIC MARKETING		BUSIN			2350
	SUPPLYING THE DEMAND		BUSIN			1925
	COMPUTER LANGUAGES		COMP			2600
	COMPUTERS: AN INTRODUCTION		COMP			1899
	NUMBERING SYSTEMS		COMP			360
	SYSTEM PROGRAMMING		COMP			3195
	VIDEO GAME DESIGN		COMP			2199
	EDITING SOFTWARE MANUALS		ENGL			1450
	INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL			595
	MODERN ANTHOLOGY OF WOMEN POETS		ENGL			450
	SHORT STORIES AND TALL TALES		ENGL			1520
	THE COMPLETE PROOFREADER		ENGL			625
	CRISES OF THE MIDDLE AGES		HIST			1200
	EIGHTEENTH CENTURY EUROPE		HIST			1790
	THE INDUSTRIAL REVOLUTION		HIST			795
	ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH			2600
	BEHAVIORAL ANALYSIS		PSYCH			1060
	INTRODUCTION TO PSYCHOLOGY		PSYCH			2200

Figure 25. Updated Bookstore File

7 Merging Files

To merge files, you write a MERGE control statement and several JCL statements. Whenever you merge files, you must make sure that their records have the same format and that they have been previously sorted by the same control fields. You can merge up to 16 files at a time.

Except for SORT, all statements described so far (INCLUDE, OMIT, SUM, OUTREC, and INREC) can also be used for a merge application.

Writing the MERGE Statement

The format of the MERGE statement is the same as that of the SORT statement.

So, to merge the bookstore master file with the file containing the five new records, you write the MERGE statement as shown below:

```
MERGE FIELDS=(110,5,A,1,75,A),FORMAT=CH
```

Department

Title

Writing the JCL

As in a sort, the JCL you need depends on whether you execute DFSORT with the JCL EXEC statement or call it from a program. This chapter discusses only executing DFSORT with the JCL EXEC statement.

The JCL DD statements for a merge are the same as those for a sort, with the following exceptions:

- You do *not* use the SORTWKnn DD statement.
- Instead of the SORTIN DD statement, you use SORTINnn DD statements to define the input files. You need one SORTINnn DD statement for each file that will be merged. The value nn can be a number from 01 through 16.

7 Merging Files

To merge the bookstore master file and the file containing the new records, you can code the following JCL statements (it is assumed that the input files are cataloged and that the output file will be cataloged):

```
//EXAMP    JOB  A492,PROGRAMMER
//SORT     EXEC PGM=SORT
//STEPLIB  DD   DSN=A492.SM,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//SORTIN01 DD   DSN=BOOKS.INPUT1,DISP=OLD
//SORTIN02 DD   DSN=BOOKS.INPUT2,DISP=OLD
//SORTOUT  DD   DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(1,1)),UNIT=SYSDA
//SYSIN    DD   *
    MERGE  FIELDS=(110,5,A,1,75,A),FORMAT=CH
/*
```

In the next chapter, you'll learn how to call DFSORT from a program.

Calling DFSORT from a Program

8 Calling DFSORT from a Program

You can call DFSORT from programs written in COBOL, PL/I, or Assembler language. Here, we'll limit our discussion to sorting and merging using COBOL and sorting using PL/I. For information on restrictions when using these languages and on calling DFSORT from an assembler program, see *DFSORT Application Programming: Guide.*

Passing Control Statements

When using OS/VS COBOL, VS COBOL II, or PL/I, you can pass the INCLUDE, OMIT, SUM, INREC, and/or OUTREC control statement (these program products create a RECORD and a SORT or MERGE control statement for you) to DFSORT by using the SORTCNTL DD statement. For example, you can pass the INCLUDE control statement that will select only the English department books, as follows:

```
//EXAMP    JOB A492,PROGRAMMER
  .
  .
  .
//SORTCNTL DD  *
    INCLUDE COND=(110,5,CH,EQ,C'ENGL')
/*
```

Note: When using VS COBOL II, you need to understand the use of the SORT-CONTROL special register. For full information, see *VS COBOL II Application Programming Guide.*

Calling from a COBOL Program

To call DFSORT from a COBOL program, use the COBOL statements SORT and MERGE. The following sections show sample programs that use the COBOL SORT and MERGE statements. These examples assume that the COBOL environment is available. For complete information, see *IBM OS/VS COBOL Compiler and Library Programming Guide, VS COBOL II Application Programming Guide, VS COBOL II Application Programming: Language Reference,* and *IBM VS COBOL for OS/VS.*

Sorting Records

The sample COBOL program on the following pages calls DFSORT to sort the bookstore master file (MASTER-FILE) by title in ascending order. The sorted master file is written to SORTED-MASTER-FILE.

Notice that the control field and order of the sort are specified in the COBOL program itself rather than with a SORT control statement.

SORT-RETURN is the COBOL special register for the DFSORT return code.

Calling DFSORT from a Program

Below is the JCL for the program:

```
//EXAMP    JOB  A492,PROGRAMMER
//BOOKS    EXEC PGM=COBOLPGM
//STEPLIB  DD   DSN=A492.SM,DISP=SHR
//         DD   DSN=USER.PGMLIB,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//MASTIN   DD   DSN=BOOKS.INPUT,DISP=OLD
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//MASTOUT  DD   DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(1,1)),UNIT=SYSDA
//PRINTFL  DD   SYSOUT=A
/*
```

In contrast to the JCL for executing DFSORT with the JCL EXEC statement (see Chapter 2), the above JCL has these differences:

- The program name on the EXEC statement is that of the COBOL program.
- The STEPLIB DD statement defines the library containing the DFSORT program, as well as the library containing the COBOL program.
- The name of the DD statement for the input file need not be SORTIN.
- The name of the DD statement for the output file need not be SORTOUT.
- The SYSIN DD statement is not used for passing control statements (instead, as we'll explain later, control statements are passed by the SORTCNTL DD statement).

Calling DFSORT from a Program

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
    COBOLPGM.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT SD-FILE ASSIGN TO
    DUMMYNM.
    SELECT MASTER-FILE ASSIGN TO
    MASTIN.
    SELECT SORTED-MASTER-FILE ASSIGN TO
    MASTOUT.
    SELECT PRINT-FILE ASSIGN TO
    PRINTFL.
DATA DIVISION.
FILE SECTION.
SD  SD-FILE
    DATA RECORD IS SD-RECORD.
01  SD-RECORD.
 05 TITLE-IN    PICTURE X(75).
 05 AUTH-LN-IN    PICTURE X(15).
 05 AUTH-FN-IN    PICTURE X(15).
 05 PUB-IN     PICTURE X(4).
 05 COUR-DEPT-IN  PICTURE X(5).
 05 COUR-NO-IN    PICTURE X(5).
 05 COUR-NAM-IN    PICTURE X(25).
 05 INST-LN-IN    PICTURE X(15).
 05 INST-INIT-IN    PICTURE X(2).
 05 NO-STOCK-IN    PICTURE 9(8) COMP.
 05 NO-SOLD-IN    PICTURE 9(8) COMP.
 05 PRICE-IN    PICTURE 9(8) COMP.

FD  MASTER-FILE
    DATA RECORD IS MASTER-RECORD.
01  MASTER-RECORD.
 05 FILLER     PICTURE X(173).

FD  SORTED-MASTER-FILE
    DATA RECORD IS SORTED-MASTER-RECORD.
01  SORTED-MASTER-RECORD.
 05 FILLER     PICTURE X(173).

FD  PRINT-FILE
    DATA RECORD IS OUTPUT-REPORT-RECORD.
01  OUTPUT-REPORT-RECORD.
 05 REPORT-OUT    PICTURE X(120).
 .
 .
 .

PROCEDURE DIVISION.
 .
 .
 .
```

```
SORT-ROUTINE SECTION.
    SORT SD-FILE
    ASCENDING KEY TITLE-IN
    USING MASTER-FILE
    GIVING SORTED-MASTER-FILE.
    IF SORT-RETURN > 0
    DISPLAY "SORT FAILED".
    .
    .
    .
SORT-REPORT SECTION.
    print a report on PRINT-FILE using SORTED-MASTER-FILE.
    .
    .
    .
    STOP RUN.
```

Merging Records

The sample COBOL program on the following page calls DFSORT to merge the presorted bookstore master file (MASTER-FILE) with another presorted file (NEW-BOOKS-FILE) to create a new master file (MERGED-FILE).

Below is the JCL for the program:

```
//EXAMP    JOB  A492,PROGRAMMER
//BOOKS    EXEC PGM=COBOLP
//STEPLIB  DD   DSN=A492.SM,DISP=SHR
//         DD   DSN=USER.PGMLIB,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//MASTERFL DD   DSN=BOOKS.INPUTA,DISP=OLD
//NEWBOOKS DD   DSN=BOOKS.INPUTB,DISP=OLD
//MERGEDFL DD   DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(1,1)),UNIT=SYSDA
//PRINTFL  DD   SYSOUT=A
/*
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
    COBOLP.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT SD-FILE ASSIGN TO
    DUMMYNM.
    SELECT MASTER-FILE ASSIGN TO
    MASTERFL.
    SELECT NEW-BOOKS-FILE ASSIGN TO
    NEWBOOKS.
    SELECT MERGED-FILE ASSIGN TO
    MERGEDFL.
    SELECT PRINT-FILE ASSIGN TO
    PRINTFL.
DATA DIVISION.
FILE SECTION.
SD  SD-FILE
    DATA RECORD IS SD-RECORD.
01  SD-RECORD.
 05 TITLE-KEY  PICTURE X(75).
 05 FILLER     PICTURE X(98).

FD  MASTER-FILE
    DATA RECORD IS MASTER-RECORD.
01  MASTER-RECORD.
 05 FILLER     PICTURE X(173).

FD  NEW-BOOKS-FILE
    DATA RECORD IS NEW-BOOKS-RECORD.
01  NEW-BOOKS-RECORD.
 05 FILLER    PICTURE X(173).

FD  MERGED-FILE
    DATA RECORD IS MERGED-RECORD.
01  MERGED-RECORD.
 05 FILLER    PICTURE X(173).

FD  PRINT-FILE
    DATA RECORD IS OUTPUT-RECORD.
01  OUTPUT-RECORD.
 05 FILLER    PICTURE X(120).
 .
 .
 .

PROCEDURE DIVISION.
 .
 .
 .

 MERGE-ROUTINE SECTION.
     MERGE SD-FILE
     ASCENDING KEY TITLE-KEY
     USING MASTER-FILE NEW-BOOKS-FILE
     GIVING MERGED-FILE.
     IF SORT-RETURN > 0
     DISPLAY "MERGE FAILED".
     STOP RUN.
```

8 Calling DFSORT from a Program

Using VS COBOL II FASTSRT

If you compile the previous COBOL program for sorting records with VS COBOL II, the input (from MASTER-FILE) and the output (to SORTED-MASTER-FILE) would qualify for the VS COBOL II FASTSRT option. When using this compile-time FASTSRT option, your sort runs considerably faster, because DFSORT rather than COBOL does the input and output processing. For full information on FASTSRT, refer to *VS COBOL II Application Programming Guide.*

Note: COBOL evaluates sort input and output independently to see if it qualifies for FASTSRT. If either the input or the output of your sort does not qualify because of the presence of an input or output procedure, you may be able to replace such a procedure, using DFSORT control statements to accomplish the same thing. For example, you can use a control statement (OUTREC) to indicate how records will be reformatted before being written to the output data set.

8 Calling DFSORT from a Program

Calling from a PL/I Program

When calling DFSORT from a PL/I program, the information that must be passed to DFSORT includes a SORT or MERGE control statement, a RECORD control statement, and the amount of main storage to be allocated. By now, you are familiar with the SORT and MERGE control statements. On the RECORD control statement, you specify the record type and length. Following the RECORD control statement, you specify the amount of main storage in bytes. (The more main storage you allocate, on the order of about one megabyte, the better the performance will be.)

You can also pass control statements by using the SORTCNTL DD statement.

The sample PL/I program on the following page calls DFSORT to sort the bookstore file by title. It allocates 900000 bytes of main storage.

The following is the JCL for the program. The SORTCNTL DD statement is used to pass an INCLUDE control statement that selects only the English department books.

```
//EXAMP    JOB  A492,PROGRAMMER
//BOOKS    EXEC PGM=PLIPGM
//STEPLIB  DD   DSN=A492.SM,DISP=SHR
//         DD   DSN=USER.PGMLIB,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//SORTIN   DD   DSN=BOOKS.INPUT,DISP=OLD
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTOUT  DD   DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(1,1)),UNIT=SYSDA
//SORTCNTL DD *
   INCLUDE COND=(110,5,CH,EQ,C'ENGL')
//SYSPRINT DD   SYSOUT=A
/*
```

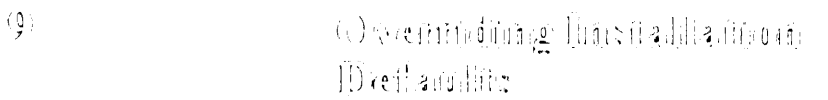
8 Calling DFSORT from a Program

```
PLIPGM:  PROC OPTIONS(MAIN);

           DCL 1 MASTER_RECORD,
               5 TITLE_IN     CHAR(75),
               5 AUTH_LN_IN   CHAR(20),
               .
               .
               .
               5 PRICE_IN     BIN FIXED(31);

           DCL RETURN_CODE FIXED BIN(31,0);
           .
           .
           .
           CALL PLISRTA (' SORT FIELDS=(1,75,CH,A) ',
                        ' RECORD TYPE=F,LENGTH=(173) ',
                        900000,
                        RETURN_CODE);
           IF RETURN_CODE ¬= 0 THEN DO;
                PUT SKIP EDIT ('SORT FAILED')(A);
                CALL PLIRETC(RETURN_CODE);


           END;
           .
           .
           .
           CALL OUTPUT;
           .
           .
           .
           OUTPUT: PROCEDURE;
           .
           .
           .
           . Print a report from the sorted master file (SORTOUT)
           .
           .
           .
           END;
END PLIPGM;
```


9
Overriding Installation
Default

9 Overriding Installation Defaults

When DFSORT is first installed, it has IBM-established defaults. During installation, your system programmer has the option of changing these defaults.

For example, one IBM default is to list the DFSORT statements in the output data set for messages; however, at your site, the default may be to *not* list the statements.

Furthermore, separate defaults may be established for jobs executed with JCL and those called from a program. So, when you execute DFSORT with JCL, the default may be to list the statements, and when you call DFSORT from a program, the default may be to *not* list them.

Although it usually isn't necessary, you can temporarily override some of the installation defaults either by specifying parameters on the JCL EXEC statement or by writing an OPTION control statement. If calling DFSORT from an assembler program, you can also override defaults by means of a parameter list.

In this chapter, we'll discuss how to override a couple of the many available defaults. We'll also limit our discussion to the JCL EXEC statement and the OPTION statement. For a list of all the possible defaults, and information on how to code the assembler parameter list, see *DFSORT Application Programming: Guide.*

Using the JCL EXEC Statement

If executing DFSORT with the JCL EXEC statement, you can use the PARM parameter to override certain defaults. For example, if the default at your installation is to list DFSORT statements and you don't want them listed, you can specify NOLIST in the PARM field, as follows:

```
//SORT  EXEC  PGM=SORT,PARM='NOLIST'
```

On the other hand, if the default is not to list the statements and you want them listed, you can specify LIST in the PARM field:

```
//SORT  EXEC  PGM=SORT,PARM='LIST'
```

Using the OPTION Statement

Whether you execute DFSORT with the JCL EXEC statement or call it from a program, you can use the OPTION statement to override certain defaults. To do this, you place the OPTION statement among the other DFSORT control statements that follow the SYSIN or SORTCNTL DD statement.

9 Overriding Installation Defaults

A particular default that can be overridden with the OPTION statement is one that specifies whether equally collating records are to be written in their original order.

The IBM default is that DFSORT may write equally collating records in random order. If your installation has kept this default and you want to temporarily override it (so that equally collating records are written in their *original* order), you can specify EQUALS on the OPTION statement, as follows:

```
OPTION  EQUALS
```

Or, if your installation has established EQUALS as the default and you want to temporarily override it (so that equally collating records may be written in *random* order), you can specify NOEQUALS on the OPTION statement:

```
OPTION  NOEQUALS
```

Note that some defaults can be overridden by the OPTION statement, but not the JCL EXEC statement, and that other defaults can be overridden by the JCL EXEC statement, but not the OPTION statement.

For a table showing all defaults and exactly how each can be overridden, see *DFSORT Application Programming: Guide.*

10 Copying Files

DFSORT lets you copy data directly without performing a sorting or merging operation.

With the exception of SUM, you can use any of the control statements discussed so far when copying. In other words, the program gives you the ability to select and reformat the specific data you want to copy.

You can specify COPY on the SORT control statement, on the MERGE statement, or on the OPTION statement.

Specifying COPY on the SORT, MERGE, and OPTION Statements

The SORT statement changes very little when you specify COPY. Just replace the parenthetical information with the word COPY as shown below:

```
SORT FIELDS=COPY
```

The MERGE statement also changes very little when you specify COPY. Again, just replace the parenthetical information with the word COPY as shown below:

```
MERGE FIELDS=COPY
```

You can specify COPY on the OPTION statement as follows:

```
OPTION COPY
```

Using COPY with INCLUDE and INREC

Suppose you would like to select and reformat specific data to be copied from the file shown in Figure 26:

	Book Title	Course Dept.	Price
BYTE	1 75	110 114	170 173
	LIVING WELL ON A SMALL BUDGET		9900
	PICK'S POCKET DICTIONARY		295
	INTRODUCTION TO BIOLOGY	BIOL	2350
	STRATEGIC MARKETING	BUSIN	2350
	SUPPLYING THE DEMAND	BUSIN	1925
	COMPUTER LANGUAGES	COMP	2600
	COMPUTERS: AN INTRODUCTION	COMP	1899
	NUMBERING SYSTEMS	COMP	360
	SYSTEM PROGRAMMING	COMP	3195
	VIDEO GAME DESIGN	COMP	2199
	EDITING SOFTWARE MANUALS	ENGL	1450
	INKLINGS: AN ANTHOLOGY OF YOUNG POETS	ENGL	595
	MODERN ANTHOLOGY OF WOMEN POETS	ENGL	450
	SHORT STORIES AND TALL TALES	ENGL	1520
	THE COMPLETE PROOFREADER	ENGL	625
	CRISES OF THE MIDDLE AGES	HIST	1200
	EIGHTEENTH CENTURY EUROPE	HIST	1790
	THE INDUSTRIAL REVOLUTION	HIST	795
	ADVANCED TOPICS IN PSYCHOANALYSIS	PSYCH	2600
	INTRODUCTION TO PSYCHOLOGY	PSYCH	2200

Figure 26. Bookstore File as a Source for a Copy Application

You have a complete bookstore file that is sorted by department and title, and you want a copy of just the reading list (without the prices) for the computer department. Figure 27 on page 66 shows the copy of the file.

```
                                                          Book        Course
                                                          Title       Dept.

BYTE                                                      1      75   110 114

COMPUTER LANGUAGES                                                    COMP
COMPUTERS: AN INTRODUCTION                                            COMP
NUMBERING SYSTEMS                                                     COMP
SYSTEM PROGRAMMING                                                    COMP
VIDEO GAME DESIGN                                                     COMP
```

Figure 27. List of Computer Texts Copied from Bookstore File

In this example, we used the INCLUDE statement to select only departments equal to "COMP", added the INREC statement to eliminate the Price field, and used the OPTION statement to specify the copy function. The statements looked like this:

```
INCLUDE COND=(110,5,CH,EQ,C'COMP')
INREC FIELDS=(1,114)
OPTION COPY
```

Writing the JCL

The JCL DD statements for a copy application are the same as those for a sort, with one exception:

- You do not use the SORTWKnn DD statement.

Below is some sample JCL that will execute the previous copy example:

```
//EXAMP    JOB  A492,PROGRAMMER
//SORT     EXEC PGM=SORT
//STEPLIB  DD   DSN=A492.SM,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//SORTIN   DD   DSN=BOOKS.INPUT,DISP=OLD
//SORTOUT  DD   DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(1,1)),UNIT=SYSDA
//          DCB=(LRECL=114,BLKSIZE=114,RECFM=F)
//SYSIN    DD   *
           INCLUDE COND=(110,5,CH,EQ,C'COMP')
           INREC FIELDS=(1,114)
           OPTION COPY
/*
```

Note: You can use SORT FIELDS=COPY or MERGE FIELDS=COPY instead of OPTION COPY to produce the same results.

III

Using DFSORT
Efficiently

11 Using DFSORT Efficiently

You will get the best performance from DFSORT if you follow these guidelines:

- Be generous with main storage.
- Use high speed disks for sort work data sets.
- Use INREC.
- Use INCLUDE or OMIT.
- Execute DFSORT with the JCL EXEC statement.
- Use FASTSRT with VS COBOL II.

Be Generous with Main Storage

The amount of storage allocated to DFSORT is a default that you can override using either the OPTION statement or the PARM field on the JCL EXEC statement.

In general, the more main storage available to DFSORT (on the order of about one megabyte), the better the performance.

If the default at your installation is small and if enough real storage is available, you will probably want to override it when you sort or merge large files.

You can allocate more storage using the OPTION statement by specifying MAINSIZE=nK, where nK is the total amount of main storage to be allocated to DFSORT. For example:

```
OPTION  MAINSIZE=1200K
```

You can allocate more storage using the PARM field on the JCL EXEC statement by specifying SIZE=nK. For example:

```
//SORT  EXEC  PGM=SORT,PARM='SIZE=1200K'
```

Use High Speed Disks

Using disks for sort work data sets (SORTWKnn) is much more efficient than using tapes. You should avoid using tapes for sort work data sets whenever possible.

High speed disks, such as the IBM 3380 Direct Access Storage, offer the best performance.

11

Using DFSORT Efficiently

Use INREC

INREC can help improve performance while shortening records. The shorter the records, the faster the processing. Therefore, you should use INREC whenever possible, to eliminate unnecessary fields.

Remember that INREC reformats records *before* they are processed, and OUTREC reformats them *after* they are processed. So, you should use INREC to shorten records, and OUTREC to lengthen records.

Use INCLUDE or OMIT

Naturally, the size of the input file(s) also affects the amount of time processing will take. The fewer the records, the faster the DFSORT application. Include or omit can help improve performance when records are excluded, so you should use INCLUDE or OMIT whenever possible to select only the records pertaining to your application.

Execute DFSORT with JCL

As a rule, DFSORT is more efficient when executed with the JCL EXEC statement than when called from a program.

Although calling DFSORT from a program may be convenient if the program modifies the data before or after DFSORT (for example, if DFSORT sums numbers and the program calculates their average), you should be aware of the possible trade-off in performance.

Use FASTSRT with VS COBOL II

With VS COBOL II, using the FASTSRT compile-time option enhances DFSORT performance. With FASTSRT, DFSORT rather than COBOL does the input and output processing. For more information on this option, see your *COBOL II Application Programming Guide.*

Appendix A.
Sample File

Book Title	Author's Last Name	Author's First Name	Publisher	Course Department	Course Number	Course Name	Instructor's Last Name	Instructor's Initials	Number In Stock	Number Sold Year-to-Date	Price
BYTE 1 75	76 90	91 105	106 109	110 114	115 119	120 144	145 159	160 161	162 165	166 169	170 173
COMPUTER LANGUAGES	MURRAY	ROBERT	FERN	COMP	00032	INTRO TO COMPUTERS	CHATTERJEE	CL	5	29	2600
LIVING WELL ON A SMALL BUDGET	DEWAN	FRANK	COR						14	1	9900
SUPPLYING THE DEMAND	MILLER	TOM	COR	BUSIN	70251	MARKETING	MAXWELL	RF	0	32	1925
VIDEO GAME DESIGN	RASMUSSEN	LORI	VALD	COMP	00205	VIDEO GAMES	NEUMANN	LB	10	10	2199
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	WILDE	KAREN	COR	ENGL	10856	MODERN POETRY	FRIEDMAN	KR	2	32	595
COMPUTERS: AN INTRODUCTION	DINSHAW	JOKHI	WETH	COMP	00032	INTRO TO COMPUTERS	CHATTERJEE	CL	20	26	1899
PICK'S POCKET DICTIONARY	GUSTLIN	CAROL	COR						46	38	295
EDITING SOFTWARE MANUALS	OJALVO	VICTOR	VALD	ENGL	10347	TECHNICAL EDITING	MENDOZA	VR	13	32	1450
NUMBERING SYSTEMS	BAYLESS	WILLIAM	FERN	COMP	00032	INTRO TO COMPUTERS	CHATTERJEE	AN	6	27	360
STRATEGIC MARKETING	YAEGER	MARK	VALD	BUSIN	70124	ADVANCED MARKETING	LORCH	MH	3	35	2350
THE INDUSTRIAL REVOLUTION	GROSS	DON	WETH	HIST	50420	WORLD HISTORY	GOODGOLD	ST	15	9	795
MODERN ANTHOLOGY OF WOMEN POETS	COWARD	PETER	COR	ENGL	10856	MODERN POETRY	FRIEDMAN	KR	1	26	450
INTRODUCTION TO PSYCHOLOGY	DUZET	LINDA	COR	PSYCH	30016	PSYCHOLOGY I	ZABOSKI	RL	26	15	2200
THE COMPLETE PROOFREADER	GREEN	ANN	FERN	ENGL	10347	TECHNICAL EDITING	MENDOZA	VR	7	19	625
SYSTEM PROGRAMMING	CAUDILLO	RAUL	WETH	COMP	00103	DATA MANAGEMENT	SMITH	DC	4	23	3195
SHORT STORIES AND TALL TALES	AVRIL	LILIANA	VALD	ENGL	10054	FICTION WRITING	BUCK	GR	10	9	1520
INTRODUCTION TO BIOLOGY	WU	CHIEN	VALD	BIOL	80521	BIOLOGY I	GREENBERG	HC	6	11	2350
ADVANCED TOPICS IN PSYCHOANALYSIS	OSTOICH	DIANNE	FERN	PSYCH	30975	PSYCHOANALYSIS	NAKATSU	FL	1	12	2600
EIGHTEENTH CENTURY EUROPE	MUNGER	ALICE	WETH	HIST	50632	EUROPEAN HISTORY	BISCARDI	HR	23	21	1790
CRISES OF THE MIDDLE AGES	BENDER	GREG	COR	HIST	50521	WORLD HISTORY	WILLERTON	DW	14	17	1200

This sample file is for use with Chapters 2 through 7.

Assume that the file is used at a college bookstore to keep information about the books it sells. Each horizontal line represents a record, and each column a record field. For the sake of illustration, the file has only twenty records, each 173 bytes long.

The first nine fields of each record contain character data and the last three fields contain binary data (binary data is shown in its character representation). Note that because binary data cannot contain decimal points, the prices are shown in cents rather than dollars.

Blanks in the fields pertaining to courses indicate that the book is not required for any class.

For your quick reference, the table to the right shows the length and data format of each field.

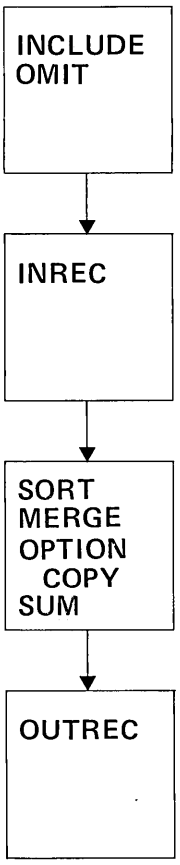
Field	Length	Data Format
Title	75	CH
Author's Last Name	15	CH
Author's First Name	15	CH
Publisher	4	CH
Course Department	5	CH
Course Number	5	CH
Course Name	25	CH
Instructor's Last Name	15	CH
Instructor's Initials	2	CH
Number In Stock	4	BI
Number Sold Y-to-D	4	BI
Price	4	BI

Appendix B.
Processing Order of Control Statements

Appendix B.
Processing Order of Control Statements

The flowchart below shows the order in which control statements are processed. (SUM is processed at the same time as SORT or MERGE; it is not used with copy.)

Although you may write the statements in any order, they will always be processed in the order shown below.



Index

Index

Z

Getting Started With DFSORT
SC26-4109-2

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

SC26-4109-2

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape | Please do not staple | Fold and tape

IBM®

Getting Started
with DFSORT

SC26-4109-2

File No. S370-33

Printed in U.S.A.