

Disk System

Concepts and Programming

Programmer's Guide

Disk System

Concepts and Programming

Programmer's Guide

Preface

Before your System/3 Disk System arrives, much time will be devoted to planning. Some activities involved in this planning are:

- 1. Selecting and educating personnel.
- 2. Planning the physical site for the system.
- 3. Studying and documenting current applications.
- 4. Designing computer application.
- 5. Writing and testing programs.
- 6. Converting data files.
- 7. Documenting operating procedures.

As a programmer you may be involved in most or all of these activities. This book explains designing and programming computer applications.

This book assumes, then, that you understand the applications in your company and that you have read the IBM System/3 Disk System Introduction, Form C21-7510.

From the *Introduction* you should know the functions of five IBM System/3 Disk System devices: Multi-Function Card Unit, Processing Unit, Printer, Disk Storage Drive, and the Printer-Keyboard. You should also know three characteristics of the System/3 disk: cylinders, tracks, and sectors. You must know these functions and characteristics to efficiently design your computer applications.

Chapters 1 and 2 of this manual explain in detail the disk concepts you must know to design applications for System/3. These chapters specifically explain how to:

- Choose the organization of a disk file
- Design disk records, calculate the size of a disk file, and document this information

Chapters 3, 4, and 5 explain how to use disk files with:

- The RPG II language
- Operation Control language
- The Disk Sort program.

None of the topics in this book are completely described. Three manuals are available for further reference.

IBM System/3 Disk System Reference Manual, Form C21-7512

IBM System/3 Disk System Utility and Sort Programs Reference Manual, Form C21-7522

IBM System/3 Disk System RPG II Reference Manual, Form C21-7504

First Edition

Changes are continually made to the specifications herein; any such change will be reported in subsequent revisions.

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, Programming Publications, Department 425, Rochester, Minnesota 55901.

Contents

CHAPTER 1: CHARACTERISTICS OF DISK FILES .	1	CHAPTER 3: RPG II DISK PROGRAMMING	21
Organizing a Disk File	1	Creating a Disk File	
Sequential Disk Files		Creating a Sequential File	22
Indexed Disk Files	1	Creating an Indexed File	22
Ordered and Unordered Records	2	Maintaining a Disk File	
Processing Indexed Files		Adding Records to a Disk File	23
Sequential Processing	2	Updating Records in a Disk File	31
Random Processing		Using External Indicators With Disk Files	47
Review-File Organization and Processing	3	Review: RPG II Entries For Disk Files	51
Creating And Maintaining Disk Files	3	Three Sample Processing Jobs	52
Creating A Sequential File	4	•	
Creating An Indexed File	5		
Adding Records to a Sequential File			61
Adding Records to An Indexed File		OCL Cards Required to Compile An RPG II Program	61
Deleting Records		OCL Cards Required to Load and Run Some IBM	
Updating Records	6	Programs	61
Reorganizing a File	6	OCL Cards Required to Load and Run An Object Program	
Review-File Maintenance	6	That Uses Card Files	62
Choosing a File Organization Method		OCL Cards Required to Load and Run A Program That	
How is the File Used?		Uses One Disk File	62
Review-File Use		OCL Cards Required to Create a Disk File	64
How Volatile is the File?		OCL Cards Required to Load and Run A Program That	
Review-Volatility		Uses More Than One Disk File	65
How Active is the File?	9	OCL Cards Required to Run A Job That Uses A Disk File	
What is the Size of the File?	9	And External Indicators	66
		OCL Cards Required to Run Several Jobs, One After	
Review-Choosing a File Organization Method		The Other	67
Examples		//Date Statement	
Example 1		Summary - OCL Statements	68
Example 2	11		
CHAPTER 2: PLANNING DISK FILES	13	CHAPTER 5: SORT	71
		Tag-Along Sort	71
Designing a Record		Describing Output Records That Retain Control	
Determining Field Size		Fields	71
Providing for a Delete Code		Describing Output Records That Drop Control	
Providing Extra Space	13	Fields	75
Naming Fields	13	Review-Tag-Along Sort	76
Documenting Record Layout	14	Header Line	76
Record Length		Field Lines	76
Block Length		ADDROUT Sort	
Review-Designing Records		OCL Required With Disk Sort Program	
Determining Size and Location of a Disk File		OCL Required With Card Sort/Collate Program	
Determining Number of Records in a File			
Calculating Record Space	17	Review-Sort	19
Calculating Index Space	17	INDEX	_
Deciding Where to Locate the File	18	INDEX	81
Review-Calculating File Space	20		
Calculation 1: Record Space	20		
Calculation 2: Index Space	20		
Calculation 3: Total Space for an Indexed File			
Chantar Davioyy	20		

This chapter is composed of four parts that describe various characteristics of disk files. The first part explains two ways to organize disk files: sequential and indexed. The second part describes the several ways indexed files can be processed. The third part explains several functions involving disk files:

- 1. Creating a file
- 2. Adding records to a file
- 3. Deleting records from a file
- 4. Updating records in a file
- 5. Reorganizing a file

The fourth part presents factors to consider when choosing one of the organization methods.

A thorough knowledge of these topics is needed to understand Chapter 3 which explains how to use RPG II for disk files.

ORGANIZING A DISK FILE

An important part of any data processing job is *file* organization. File organization is the arrangement of records in a file. The purpose of this section is to explain how disk files are organized. Since some disk files are organized like card files which you are familiar with, let's briefly review their organization.

Cards are normally arranged in a file in a particular sequence. For instance, a personnel file might be in alphabetic sequence by employee name, or it might be in numeric sequence by employee number. Data such as employee name and employee number are control fields. Recall that control fields determine the order of the cards in a file.

When card files are processed, cards are read one after another in order. To process only certain cards, all cards must be read. Thus, it takes longer to process a few cards in a large file than to process a file containing only the desired cards. Card sorters and programs, such as the Card Sort/Collate program, can be used to decrease card processing time by selecting only the desired cards and then processing them.

Also, if a card file is needed in one sequence for one job and another sequence for another job, the file must be sorted for each job. For example, salesmen receive a monthly report of their commissions. To produce this report the monthly transaction cards must be sorted in salesman number sequence. Then when the monthly sales report (the amount of sales for each item) is produced, the same transaction cards must be sorted by item number. The same file is used for both reports, but it is arranged in two different orders according to two different control fields: one by salesman number and one by item number.

Sequential Disk Files

A disk file can be organized and processed like a card file. Such a disk file is called a *sequential* file.

When a sequential disk file is processed like a card file, records are processed one after another in the order they occur. An example of a sequential disk file is an employee master file. This file contains information needed for various reports concerning each employee, such as payroll checks. Since checks are usually produced in order by employee number, records are processed in order. The lowest employee number is processed first and so on until the last record, the highest employee number, is processed.

Why, then, should a sequential file be placed on disk instead of cards? The major advantages are storage and speed. A disk can hold the same amount of data as 25,600, 96-column cards. Not only can large amounts of data be stored on a disk, but one disk pack is more convenient to handle than many cards. The disk storage drive also provides faster processing of records than the MFCU.

Sequential disk files, such as the employee master file, are processed consecutively. Consecutive processing means records are processed one after another in the order they occur. To process only certain records all records must be processed, or at least read. In this case, as with cards, consecutive processing can be time consuming.

Indexed Disk Files

It would be helpful, then, if disk records were available like books in a library where you go to an index, find the location where the book is stored, go to the right shelf, and get the book you want. No one would read all books in the shelves before reading the desired book. Likewise, it would be desirable to skip the records not needed in a job and process only the desired ones. Disk files can be organized to overcome the limitation of consecutive processing; they can be *indexed*.

An indexed disk file is organized somewhat like the books in a library; it has two parts, an *index* and the records. The index contains two facts about each record in a file. First the contents of the record's *key field* appear in the index. A key field contains data that uniquely identifies a record. For example, customer number may be the key field for a customer master record. Then, the *disk address* follows the key field. The disk address represents the location on the disk where the record is stored. For each record, then, there is an *entry* in the index that contains

KEY FIELD DISK ADDRESS

An index contains the same number of entries as there are records in the file. A file with 2000 records has 2000 entries.

Ordered and Unordered Records

The records in an indexed file or a sequential file can be ordered or unordered. An ordered file means the records are arranged in order according to some major control field or by frequency of use. An unordered file means the records are not in any particular order.

For example, a wholesale distributor organizes his file of inventory items as an indexed file. The records are loaded on the file by frequency of use (unordered). Thus, the most active items are at the beginning of file. However, the index is created in ascending order by item number. When the file is used to write customer orders, most of the records needed are located in a small area of the file. The total time to process the orders is less than if these records were scattered throughout the *entire* file.

PROCESSING INDEXED FILES

As stated previously, indexed files overcome the limitation of consecutive processing which is often used with a sequential file. Indexed files can be processed several ways because the index provides several ways to find records.

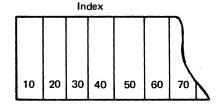
Sequential Processing

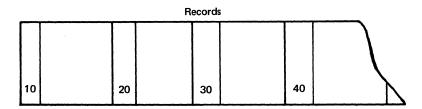
When an indexed file is processed sequentially, the keys are processed one after the other in ascending order. If the records are not in order on the disk, they can be processed in order by using the index. There are two ways to sequentially process an indexed file.

Sequential by Key

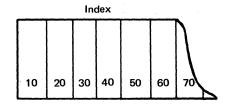
One way to sequentially process an indexed file is sequentially by key. Sequential by key means processing records in the order of the keys. This method is used to process all records in a file regardless of their order. To illustrate this processing method, note the differences and similarities between File A and File B.

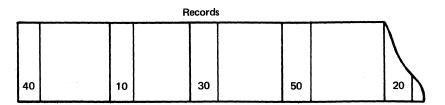
FILE A





FILE B

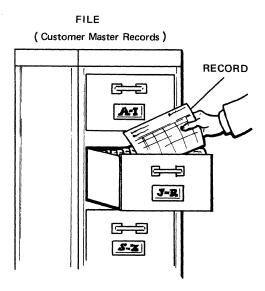




Both files contain the same records. Both indexes are in order according to key field. The only difference between the two files is the order of the records. The records in File A are in order according to key field. The records in File B are unordered. All records in both files can be processed in order by specifying sequential by key. (You specify processing method in your RPG II programs.)

Sequential Within Limits

The second way to sequentially process an indexed file is sequential within limits. Sequential within limits is like processing all the records one after another in order, in a filing cabinet drawer that is marked to identify that group of records.



Sequential within limits processing is best suited to jobs where records are processed in groups at specified times. For example, a wholesale company prepares monthly statements of each customer's charges. Statements are written on the tenth, twentieth, and last day of each month. On the tenth, statements are written for all customers whose names are in the first part of the alphabet (A-I). Statements for the other two parts, J-R and S-Z, are written on the 20th and the last day, respectively. This type of statement writing is called *cyclic*.

Sequential within limits can also be used to process several groups of records at one time. Sequential within limits is not illustrated in Chapter 3, RPG II Disk Programming. If you must use this capability, refer to IBM System/3 Disk System RPG II Reference Manual, Form C21-7504.

Random Processing

Indexed files can also be processed randomly. This type of processing also uses the index and is called random by key. This method permits processing of one particular record without regard to its relation to other records. You must specify the key of the record (through RPG II). The key is then found in the index; the disk address (adjacent to key) is used to locate the record; and the record is transferred to storage for processing. For example, records in a customer master file are to be updated to reflect current information. The transaction cards are not in order. The disk record is found by matching the customer number in the card with the key (customer number) in the index. The disk address, adjacent to this key, is then used to find the record.

Often an indexed file is used in several different jobs each of which requires a different processing method. For example, during statement writing, a customer file may be processed sequentially within limits to allow cyclic statement writing. During a billing job, the same file may be processed randomly by key to allow updating master records with unsorted transaction cards. Then during an aged trial balance job (each customer's outstanding balance is printed), the file may be processed sequentially by key.

REVIEW - FILE ORGANIZATION AND PROCESSING

So far you have studied two ways to organize a disk file: sequential and indexed. You also learned that a sequential file can be processed consecutively, one record after another in order, and that an indexed file can be processed sequentially and randomly. Sequential processing can be by key or within limits. Random processing is by key.

CREATING AND MAINTAINING DISK FILES

Before a disk file can be processed, it must be written on a disk. The first time a file is written on the disk, the process is called *loading* or *creating*. A card file can be used to create a disk file. You record in cards the data you want on the disk. Then you write an RPG II program that transfers the data from the cards to the disk. Chapter 3 explains how to write such an RPG II program.

Once a file is created, file maintenance is often necessary. File maintenance means performing those functions that keep a file current for daily processing needs. Some file maintenance functions common to both sequential and indexed files are: adding, deleting, and updating records. Adding means putting a record in a file after the file is created. Deleting means identifying a record so it won't be processed with the other records. Updating means adding or changing some data in a record.

Creating a Sequential File

A sequentially organized disk file is similar to a sequentially organized card file. The records are either ordered, in ascending or descending order by some control field, or unordered. However, regardless of order when a sequential file is created, records are placed consecutively, one after another on the disk. Both tracks in one cylinder are filled, then both tracks in the next cylinder, until the whole file is placed on the disk. Figure 1 shows this process.

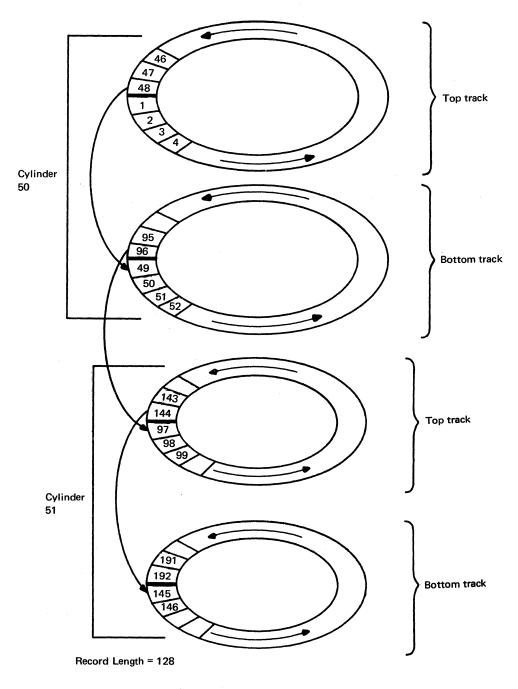


Figure 1. Writing Records on a Disk

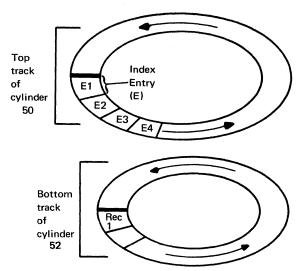
The numbers on the tracks in the figure correspond to the number and position of each record. Record 1 is written at the beginning of the top track of cylinder 50. Records 2 through 48 are written, in order, on the top track of cylinder 50. Then, records 49 through 96 are written, in order, on the bottom track of cylinder 50. When cylinder 50 is filled, cylinder 51 and subsequent cylinders are filled in the same manner.

In this example, each record is 128 positions long. Recall that a sector is 256 positions long, and that a track contains 24 sectors. So, in this example, two records are written in each sector; 48 records are written on each track; and 96 records are written on a cylinder.

The 97th record is written at the beginning of the top track of cylinder 51. The next 95 records are written on cylinder 51. Ninety six more records are written on cylinder 52, cylinder 53, and so on until all records are written on the disk.

Creating an Indexed File

When an indexed file is created, the index is created as the records are written on the disk. The index area precedes the area where records are placed. In an indexed file, the records are either ordered or unordered, but the index contains key fields which are *always* ordered.



The index for a certain file takes five tracks. So index entries are written on cylinders 50 and 51 and the top track of cylinder 52.

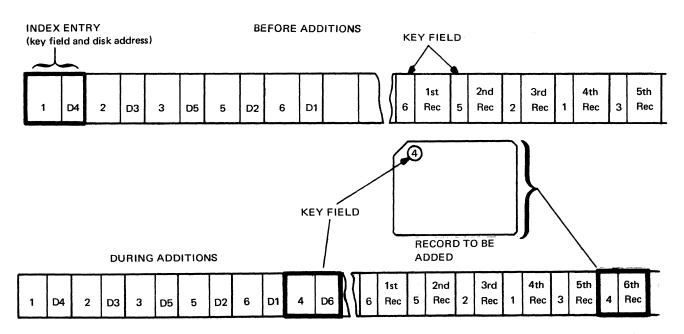
Records are written beginning in the first sector of the bottom track of cylinder 52. Both the index area and the record area must start at the beginning of a track.

Adding Records to a Sequential File

When records are added to a sequential file, they are written at the end of the records already in that file. Records are so added whether the records already present are ordered or unordered.

Adding Records to an Indexed File

When a record is added to an indexed file it is also written at the end of the records already in that file. The index entry (key field and disk address) for that record is written at the end of the current entries in the index area.



After all records are added, the index is automatically sorted so the keys of the new records are in ascending sequence.

AFTER ADDITIONS

-		-										T	n۲	·		_		_			r			 	
													$\ \ $		1st		2nd		4th		5th		6th		
	. 1	D4	2	D3	3	D5	4	D6	5	D2	6	D2	W	6	Rec	5	Rec	7	Rec	3	Rec	4	Rec		

Deleting Records

When a record becomes inactive, you will no longer want to process it with the other records. A record is not physically removed from a file during regular processing; thus, it is necessary to identify or tag the record so it can be bypassed. One way of tagging such a record is to put a code in a particular location in the record. This code is referred to as a delete code. When the file is processed, your program must check for the delete code; if the code is present, the program bypasses that record. The delete procedure is the same for both sequential and indexed files.

Updating Records

The update function can also be the same for both indexed and sequential files. The record to be updated is read into storage. The data to be changed is changed in storage, and then the record is written on the disk in its original location.

Reorganizing a File

Another file maintenance function performed on indexed files is *reorganization*. Reorganizing means re-creating an indexed file. Reorganization of an indexed file may be necessary for two reasons: to increase processing efficiency and to free disk space.

For example, an indexed file was created with the records in ascending key field order. Since that time several records were added to the file. Recall that these records are added at the end of the file, but the keys are in order in the index. When this file is processed sequentially by key, the disk access arm must move back and forth between the sequenced records (those originally created) and the added records. Often processing time for a particular job increases in this situation. During reorganization the added records can be placed in sequence in the re-created file.

Another reason to reorganize a file is that often the space reserved for a file becomes filled as additions are made. (Planning file space is discussed in Chapter 2). Reorganizing

is a means of freeing space since inactive records, those with a delete code, can be physically removed from a file during reorganization.

Reorganization, then, is a means of physically placing added records in sequence with the records originally created. It is also a means of physically removing records that are tagged for deletion. The IBM Disk Copy/Dump program can be used to reorganize an indexed file. The IBM System/3 Disk System Reference Manual, Form C21-7512, explains how to use the program to reorganize indexed files.

REVIEW - FILE MAINTENANCE

This section explained file creation and maintenance. The file maintenance functions explained were: adding records, deleting records, updating records, and reorganizing a file. Creating and reorganizing a file are usually performed independently of other processing. Adding, deleting, and updating can occur separately, but most usually occur during processing of a specific job. Examples explaining how these functions are coded in RPG II are contained in Chapter 3.

CHOOSING A FILE ORGANIZATION METHOD

The first two sections described several characteristics of the System/3 Disk System and explained the flexibility they provide to perform a variety of jobs. Due to this flexibility and variety, it is important for you to analyze each of your jobs and choose the file organization method that gives you the best possible performance.

In many cases, the type or organization for a file is immediately evident. Some applications, however, may require additional study because of their complexity, because a file is used in several jobs, or because unusual processing is required. Studying existing applications is an important aspect of planning for a data processing system. Decisions in this area must be made *before* programming begins, since efficiency of your data processing installation may be affected. This section described factors to consider when making these decisions.

There are no absolute rules for choosing a file organization method. However, several characteristics of the file to consider are:

- 1. Use of the file
- 2. Volatility of the file
- 3. Activity of the file
- 4. Size of the file

Each of these four characteristics is discussed next. Following these discussions are two examples. One example explains why a sequential file is chosen; the other describes why an indexed file is chosen.

How is the File Used?

The answer to this question takes priority over all other considerations. To answer this question you must ask several other questions.

First, is the file a master file? Recall that a master file is fairly permanent, is generally used in several jobs, and is often used with several other files. For example, a customer file contains a record for each customer. Each record may contain such data as customer name and address, shipping information, credit status, accounts receivable, and sales information. Although certain data in a record such as accounts receivable, may change, the record remains in the file as long as the customer does business with the company. (These changes are made with a transaction file.) Since this file contains so much information about each customer, it may be used in several jobs to produce various reports. Likewise, the file may be used with several other files, master or transaction.

Recall that a transaction file contains records of a less permanent nature than a master file, and transaction files contain data that is used to update a master file.

Thus, when choosing a file organization method for a master file, the major question to ask is: What are the processing requirements of the file? To answer this question, you must study the applications in which the file is used and ask:

- Is the file used with other files or in several jobs?
 - 1. If so, what is the organization of the other files?
 - 2. If used with transaction files, are the transaction records ordered or unordered?

- Must the file be sorted for any jobs?
- Must the file provide for inquiry?

Using a Master File with Several Files or in Several Jobs

If a master file is used with several files (a transaction file, another master file, or both), the master file can be either sequential or indexed. The determining factors are the processing requirements of the various runs that will be using the file and the organization of the other files.

If transaction files are ordered (sorted in the same sequence as the master file), then the master file may be either sequential or indexed. However, to process unordered transaction files against a master file, the master file must be indexed and processed randomly by key.

If the master file is used in several jobs, and records must be processed both in order and randomly, then indexed is the better organization. Records can be processed in order (sequential by key) and randomly (random by key).

Sorting a Master File

If the master file must be sorted for some jobs, a sequential file is easier to sort than an indexed file. The Disk Sort program cannot produce a sorted indexed file. That is, an indexed file can be sorted; but the sorted file will be a sequential file. Instead of keeping the sorted file as the master file, the original file must be kept. Chapter 5 explains the Disk Sort program in more detail.

Inquiring Against a Master File

Most businesses need to get information from a file on an *inquiry* basis. An inquiry is a request for information from some type of storage.

Some jobs that emphasize the importance of immediate inquiry and response are:

Demand Deposit Accounting

What is the balance of account number

133420?

Inventory Control

How many of part number 55632 are on

order?

Manufacturing

How many sub-assemblies of part number 16414

are on hand?

Payroll

What are the year-todate earnings for employee number 13862?

The System/3 Disk System provides for inquiry. The ability to obtain the desired information and the speed with which it can be obtained depend upon the organization of the file.

Where inquiry is required, a critical question in choosing the best file organization method is: How fast must the inquiry be answered? The less critical the response time, the greater the choice of organization and processing methods.

To decide how fast the inquiry must be, ask a series of questions. Can the answer to the inquiry wait until the next updating of the specific master file? If it can, then these inquiries can be treated as additional transaction records and so processed. File organization, in this case, could be either sequential or indexed depending on other processing needs. If the inquiry cannot wait, additional questions must be asked.

Can the answer wait until the end of the present computer run? If so, the disk pack containing the specific master file is mounted at the completion of the current job; the inquiry program is loaded; and the file is processed to produce the required answers. Obviously, response time varies considerably depending on (1) the job that is in progress when the inquiry arrives and (2) the organization of the file inquired against.

An indexed file processed randomly by key will usually provide the best response time. (If the desired record were the first record in a sequential file, the response is also fast, but this type of inquiry is rare.)

Review-File Use

You must consider several factors to answer the question: How is the file used? Specific answers are impossible without knowing the specific file and the applications that use it. However, if you consider the processing requirements of the file, you have some guidelines for arriving at specific answers.

How Volatile is the File?

The number and frequency of records added to or deleted from a file affect the type of file organization chosen. *Volatility* refers to number of additions and deletions. High volatility means many records are added and deleted; low volatility means few records are added or deleted.

Recall that for both sequential and indexed files records are added at the end of the current records. If a file is sequential and the control fields of the added records are higher than the last record on the file, additions cause no problem. However, if they are not higher, and processing of the file depends on the records being processed in control field order, additions do cause a problem. In this situation, records added at the end of the file would be out of sequence.

To avoid this problem, the disk file can be re-created when such additions are made. To re-create a sequential file you write an RPG II program. Job 4 in Chapter 3 illustrates this process.

However, if additions are made to an indexed file, there is no need to rewrite the file. Records are also added at the end of the file, but the keys are in ascending order in the index. Thus, if the records must be procedded in order, they can be processed sequentially by key. Thus, one of the advantages of an indexed file is that additions and deletions can be handled without rewriting the file.

However, as the number of additions increases, the efficiency of sequentially processing an indexed file decreases. Sequentially processing the added records by key requires more time than processing the records from continuous positions on the disk. This increase occurs because additional access arm movement is required to read records at the end of the file. The arm must move back and forth between the index and the records. Even if the original records are in sequence, the added records are not. The arm must make one additional move for each added record that is processed.

Thus, for a highly volatile file where records must be processed in order, a sequential file with consecutive processing is best. However, if a highly volatile file does not require processing records in order, the file can be indexed and processed randomly by key.

If a highly volatile file requires both sequential and random processing, an indexed file is best. In this case, to overcome the decrease in time due to arm movement, the file can be reorganized.

Recall that reorganization means building a new indexed file from the old one by physically merging added records in key order and excluding all records tagged for deletion. Processing with the reorganized file is highly efficient. As additions and deletions occur, this efficiency gradually diminishes, until you reach a point where reorganization again becomes advisable.

Review - Volatility

The critical questions concerning volatility are:

- 1. What are the processing requirements for the file?
- 2. Is volatility high or low?

Possible answers are:

- 1. If all records must be processed in order and volatility is high, a sequential file is suggested.
- If only certain records are processed and volatility is high, an indexed file (random by key) with planned reorganization is suggested. If this file is used in other jobs where records are needed in order, it can be processed sequentially by key.
- 3. If volatility is low, an indexed file is better. There is no need to reorganize this file as often as a highly volatile indexed file. To process all records, the file can be processed sequentially by key. Also, an indexed file allows particular records to be processed (sequential within limits or random by key).

As you can see several factors influence the answers to the two questions. It is impossible to provide specific answers. Only with a well-defined file and well-defined applications can specific answers be given.

How Active is the File?

The next important consideration, after volatility, is the activity of the file. Activity refers to the number of records in a file for which there are transactions. Activity is usually expressed as a percentage: for example, 10 percent of the records in a file. Disregarding other processing needs, the higher the activity the better the file is suited to sequential organization. (A high percentage indicates high activity.)

For example, activity on 100 cards out of 1000 in an inventory file means that, at any one time, there are transactions to be posted to 10 percent of the records in the file. As activity increases, consecutive processing becomes more efficient. This implies a sequential file with consecutive processing or an indexed file processed sequentially by key. An indexed file may be necessary for a high activity file, if other processing needs so justify. Low activity may justify an indexed file processed randomly by key.

For an high activity file, you should consider *batch* processing. This means the application does not require transaction records to be processed the moment they occur;

some time lag is all right. Transactions can be accumulated, or batched, and processed at certain times. The time lag may be hours, weeks, or even months, depending on the application.

Generally, you may assume that low activity justifies an indexed file processed randomly by key, and high activity justifies a sequential file processed consecutively.

What is the Size of the File?

You must consider on-line capability. Recall that on-line means data is available on the disk for processing. The size of a file is affected by two important file-organization characteristics:

- 1. A sequential disk file can be written on any number of disks, which can then be mounted and processed consecutively.
- 2. An indexed file must be entirely on-line.

The fact that indexed files must be entirely on-line imposes physical restrictions on maximum file size. If you must periodically reorganize an indexed file, you must have enough on-line disk space to contain the old file and the new file. If a disk file exceeds the size limitations imposed by indexed organization, you can organize it as a sequential file. However, if other factors require the file to be indexed, you can punch the new file on cards and then create it on disk from the cards, instead of using the Disk Copy/Dump program to reorganize.

Sorting a File

If a file will be sorted by the System/3 Disk Sort program, the size of the file also affects the choice of a file organization method.

The System/3 Disk Sort program uses disk work areas. A work area is space on the disk that the program uses to arrange records in the specified order. The size of these work areas must be considered when planning files that need sorting.

When a sequential disk file is sorted, the maximum size of the input file is a little less than half the total on-line disk storage capacity. On a 1-spindle system, the input file can be overlaid by the output file. In this case, the input file is not preserved. On a 2-spindle system, half the total on-line capacity is one pack. The pack that contains the input file can be removed before the sort program starts writing the output file. Another pack can be mounted, and in this manner the input file can be preserved.

If the size of the file to be sorted appears too large for a particular system, another way to sort the file is available.

Instead of sorting the records, an *ADDROUT* sort may be performed. An ADDROUT sort produces a file of relative record numbers. The relative record number can be used by an RPG II program to specify the location of a record in the disk file. The record numbers for a file are sorted into the sequence specified by the control fields. These numbers are written on the disk. They can be used as input to an RPG II program that processes the records in the desired sequence.

The ADDROUT sort offers three advantages:

- 1. The original file is preserved.
- 2. The work and output areas must only be large enough to provide space for the record numbers, not for the records.
- 3. Random processing of a sequential file is possible through RPG II.

REVIEW – CHOOSING A FILE ORGANIZATION METHOD

To choose a file organization method, many factors must be considered. Four major questions must be asked.

- 1. How is the file used?
- 2. How active is the file?
- 3. How volatile is the file?
- 4. What is the size of the file?

Each of these questions pose additional questions. The four major questions and subsequent questions are not independent. Many times all four factors are important, and often they are equally important. The choice is yours. Only you know the requirements of your installation.

EXAMPLES

The following two examples show how the factors just discussed apply to specific file characteristics and which file organization method is best under these assumed conditions.

Example 1

File: Customer Master File

Characteristics:

1. Applications that use the file:

Billing—prepared as items are shipped

Statement Writing—prepared monthly in customer number order

Aged Trial Balance Report—prepared monthly in customer number order

- 2. Volatility—30 new records a week; 25 records tagged as deleted.
- 3. Activity—20 percent highly active records; 10 percent inactive records; 70 percent slightly active records
- 4. File Size—2000 records at creation time

Since orders occur randomly, the billing job requires the ability to process the file randomly by key. Statements and

the Aged Trial Balance Report are printed in customer number order, so sequential processing is required. Since an indexed file can be processed both sequentially by key and randomly by key, this file lends itself to the index file organization method.

Since activity is low, only 20 percent of the records are processed during billing, an indexed file is better than a sequential file. Recall that records in a sequential file must be processed in order one after the other.

Since 30 new customers are added each week, at the end of three months the file will contain about 2400 records.

 $30 \times 4 = 120 \text{ records a month}$

 $120 \times 3 = 360$ records in three months

2000 records at beginning

360 added records in three months

2360 records at end of three months

At the end of the fourth month, 2480 records would be in the file (2360 + 120 = 2480). If at creation time, file size is defined as 2400 records, the fourth month poses a problem: what to do with the 80 extra records? Since 300 inactive records can be deleted during reorganization, the volatility of this file requires quarterly reorganization.

File size for 2400 records is approximately 56 tracks; so size is not a problem if the file is an indexed file and it is reorganized every three months. (Chapter 2 explains the calculations needed to determine exact file size.)

Recommended File Organization: Indexed; quarterly reorganization.

Example 2

File: Employee Payroll (Hourly and Salary)

Characteristics:

1. Applications that use the file:

Payroll Checks—printed in ascending order by employee number

Payroll Register—printed in ascending order by employee number

- 2. Volatility—five new employee records a month
- 3. Activity—98 percent active records; 2 percent inactive records
- 4. File Size—240 records

Payroll checks and the Payroll Register are printed in ascending order. Employee numbers are assigned consecutively; as a new employee joins the company, he is assigned the next available employee number. Thus all additions are placed at the end of the file, but they are in order.

Since the reports are printed in ascending order, and additions occur in order, the file can be processed in order. High activity and low volatility suggest a sequential file.

Recommended File Organization: Sequential with consecutive processing.

After deciding which file organization method to use, you should design the record and determine file size and location. This chapter explains how to design records and determine file size and location.

DESIGNING A RECORD

The applications that use a certain file determine what data is needed in a record. You should study these applications and then decide the *layout* of the record. Layout means the arrangement of fields in a record. When you design a record, you must consider processing requirements of the record and then determine field length, location, and name.

To illustrate these design considerations, a name and address file is used in this chapter. Each record in the file contains the following data:

Field	Size (number of positions)
Customer Number	6
Name	20
Street Address	20
City and State	20
Record code	2
Delete code	1
(Other fields)	47 (total)
	116 TOTAL

Determining Field Size

Field size depends on the nature of the data in the field. First, the length of the data may vary. In the example, name is 20 positions. The length of each customer's name varies, but 20 positions should be sufficient for all names. Secondly, all data in a field may be the same length. For example, customer number is six positions, and all six positions are used in each record.

There are no firm rules for determining field size. The major problem involves fields with variable length data. For example, if name is planned as 15 positions, and a new customer has 19 characters in his name, a problem arises

when adding his record to the file. To avoid this problem, try to estimate the largest length of the data that will be contained in a field. Use this length to determine field size.

Providing for a Delete Code

Recall that records are not automatically deleted. You must place a delete code on a record with an RPG II program. Then when the file is processed, you must check for this code with an RPG II program. In the example, if a customer becomes inactive, we don't want to process his record. Thus, a 1-position field is included to provide for a delete code.

Providing Extra Space

At this stage in planning, it is often wise to allow for data to be added to a record. For example, suppose this name and address file were created with the fields described, and at a later time each customer's zip code is needed. If all positions in the record are used, there is no place to add zip code. Since record length is not yet established, we can allow for such additions to this record. Although it is often difficult at the planning stage to imagine what data might be added, it is wise to reserve extra space.

Naming Fields

At the same time you are determining field size and location, you can also decide on names for each field. Since you must specify field names in your RPG II programs, it is a good practice to choose names that follow the RPG II rules for forming field names. If these rules are considered at this planning stage, your RPG II programs are easier to write.

An RPG II field name can be from one to six characters long. The first character must be an alphabetic character, but the remaining characters can be any combination of alphabetic or numeric characters. Blanks and special characters are not allowed. The field names in Figure 2 follow these rules.

One other consideration is important when choosing field names: the name should be meaningful. Since field names are restricted to six characters, abbreviations are often

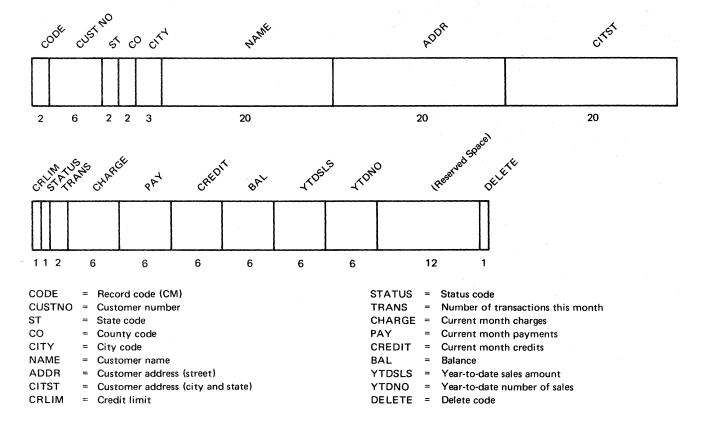


Figure 2. Layout of Customer Master Record

necessary. For example, the word *address* has seven letters; it is shortened to ADDR in Figure 2. Meaningful field names contribute to better documentation, and often avoid misinterpretation or confusion while writing RPG II programs.

DOCUMENTING RECORD LAYOUT

When record layouts are documented, your RPG II programs are easier to write. Figure 2 shows the layout of a customer master record. A record layout should include the order of the fields in the record, the length of each field, and the name of each field.

Record Length

A record may contain all pre-defined fields, or space may be reserved for data to be added to the record. In either case, all records in a particular file must be the same length. In your RPG II programs you must specify *record length*. Record length is the sum of the field lengths (including reserved space).

In our example, the sum of the fields is 116 positions. However, record length is established at 128, thus reserving 12 positions for data that might be needed at a later time.

Block Length

In your RPG II programs, you must also specify information about blocks. A block is the number of records that is transferred between a disk file and the processing unit (input) or between the processing unit and a disk file (output). Although only one record at a time is available for processing by your program, one or several records may be transferred at one time. Transferring blocks of records can decrease the time required to perform a job. When records are transferred one at a time, access time is required for the disk access arm to locate each record. When several records are transferred at a time, access time is usually less.

When more than one record is transferred, the records are blocked. Transferring blocked records can result in more rapid processing. When only one record is transferred at a time, the records are unblocked.

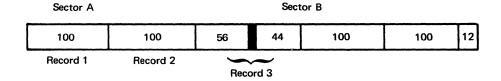
You may want to use unblocked records when an RPG II program takes a large amount of storage. In this case, there is a possible increase in total time to do the job, but your program will fit in storage.

Block length is a *multiple* of record length. For example, if your record length is 64, block length could be 256 (64 X 4 = 256). Block length is four times as large as record length. The multiple, 4 in this case, indicates the number of records you want transferred at one time.

The design of the System/3 Disk System influences block length. Recall that the smallest division of a disk is a sector, and it can contain up to 256 characters. The system transfers data in sectors, that is, multiples of 256 characters. If your record length is 128, you might have a block length of 256, indicating that you want two records transferred (128 X 2 = 256). Or you might have a block length of 512, indicating that four records are transferred (128 X 4 = 512).

For efficient blocking, you should choose a record length that is a multiple of 256 (256 X 2 = 512) or sub-multiple of 256. A sub-multiple is a number that divides into 256 a whole number of times. For example, 64 is a sub-multiple of $256 (256 \div 64 = 4)$.

You can specify a record length that is not a multiple or sub-multiple of 256. The system allows you complete flexibility in choosing a record length to fit your application and your disk storage capacity. When you use a record length which is not a multiple or sub-multiple of 256, no disk storage is wasted; some records will simply reside in more than one sector.

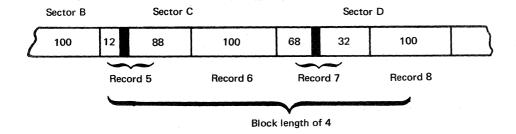


However, when you specify 100-character records as shown in the example, the computer requires more *main storage* to process these records.

You recall that the system always transfers data from disk to the computer in increments of sectors. To process record 3 therefore, two sectors must be in main storage, sector A and sector B. Fifty-six characters of record 3 reside in sector A; the remaining 44 reside in sector B. Thus, to process 100-character records with a block length of 1 requires that 512 characters (two sectors) be available in main storage.

As another example, suppose you specified 100-character records with a block length of 4. Four 100-character records *can* span three sectors.

To process your records in this case requires 768 characters (three sectors) in main storage.



REVIEW - DESIGNING RECORDS

When you design a record, consider the following questions:

- 1. Does the record contain all fields needed for all applications that use the file?
- 2. Should the record provide for a delete code?
- 3. Is extra space reserved for fields to be added?
- 4. Are the records blocked? If so, does the block length take into consideration that data is transferred in multiples of 256 characters?
- 5. Is the record layout well-documented?
 - a. Are field location and size clearly marked?
 - b. Do field names follow the RPG II rules for forming field names?

DETERMINING SIZE AND LOCATION OF A DISK FILE

Another aspect of the planning stage is (1) determining how much disk space a file requires and (2) deciding where to locate the file on the disk. These two factors must be considered together since they directly affect each other. For example, two files are already written on a disk on cylinders 8-155. A third file is to be created; it will occupy 55 cylinders. Since a disk contains 200 cylinders, this file is too large to be contained on this disk (155 + 55 = 210). The file must be written on another disk.

Determining Number of Records in a File

Let's first consider the disk space required for a file. To determine this space, you must plan how many records will be in the file at a specified time.

To determine the number of records in a file, you must consider several factors. First, you must know how many records will be in the file when it is created. If the file already exists, perhaps as a card file, use the number of records in this file as a base.

You must also know if records will be added or deleted. If additions are expected, how many records are expected, and how often will they occur? If records will be tagged for deletion, consider periodically removing them from the file. By removing records that you no longer need, you free disk space and allow more records to be added.

Only after considering these factors and the applications that use the file can you determine the number of records in the file. For example, the customer name and address file will contain 6000 records at creation time. It is estimated that each month 200 records will be added and 80 records will be deleted. It is also planned that the deletion records will be removed once a month. At the end of six months the file will contain 6720 records. (1200 records are added; 480 records are deleted).

6000	Records at creation
+1200	Records added in six months
7200	
- 480	Records deleted in six months
6720	Records in file after six months

This example points out another factor to consider. When determining the number of records in a file, consider expansion for a reasonable time into the future (at least six months). Of course, most files have deletions, and thus growth is usually slow. In a file where the number of additions and deletions are about the same, deletion records need be removed only when the disk space allowed for the file is filled.

When you create a disk file, you must give System/3 information about the size of your file. You give this information to the system through Operation Control Language (OCL) cards. (OCL is discussed in Chapter 4.) However, during the planning stage, you must calculate how much space is needed so you can decide if a file will fit on a certain disk.

Calculating Record Space

The amount of space required for a file depends upon your file organization method: sequential or indexed. If an indexed file and a sequential file contain the same number of records, the amount of space required for the records in both files is the same. However, additional space is required for the index of an indexed file.

Since the same amount of space is required for the records of a sequential file and an indexed file of the same size, record space is calculated in the same way for both files. To determine record space, you must know the number of characters in the file.

To calculate the number of characters in a file, multiply the number of records (allowing for expansion) by the length of each record. For the customer name and address file, there will be 6,720 records in the file at the end of six months. Each record contains 128 characters. Thus, the number of characters in the file is calculated as:

6720 Number of records in the file

x128 Number of characters in each record

53760

13440

6720

860,160 Total characters in the file

You must know how many tracks are needed. Since a track contains 24 sectors, and a sector contains 256 characters, each track can contain 6,144 characters (24 X 256 = 6144). To calculate the number of tracks the file requires, divide the number of characters in the file by 6144. In our example, this calculation is:

	140	Tracks
6144 🦣	860160	
	6144	
	24576	!
	24576	
	0	
	0	
1		

The calculation results in a quotient of 140 and no remainder. So 140 tracks are needed for the name and address file.

When your calculation has a remainder, always add one more track to the quotient. Otherwise, space is not reserved for the remaining characters.

Calculating Index Space

If the file is indexed, you must also determine the amount of space for the index. To find the space needed for the index, you must know the size of the index entry. Recall that an index entry is composed of a key and a disk address. Key lengths vary, depending upon the application, but disk addresses are always three characters long. Thus, the size of an entry is the key field length plus 3.

For the name and address file, the key field is customer number (CUSTNO), and it is six characters long. In this case, the index entry length is 9(6 + 3 = 9).

Another factor affecting index space is sector length. Recall that a sector is the smallest division of a disk and can contain up to 256 characters. For System/3 an index entry must be completely contained within a sector: an entry cannot start in one sector and end in a different sector.

To determine the number of entries that can be written in a sector, divide 256 by the entry length. For the name and address example (entry length is 9), this calculation is:

	28	Entries in a sector
Entry Length 9) 256	
	18	
	76	
•	<u>72</u>	
	4	Remainder
1		

Notice that the division results in a remainder of 4. Thus, 28 entries can be written in one sector. The last four positions of the sector are not used since a complete entry must be written in a sector. The twenty-ninth entry is written in the first nine positions of the next sector.

Remember, when calculating the number of index entries in a sector, drop the remainder.

Since index space, like record space, is specified in number of tracks, you must convert the sector space to track space. To do this, you must perform two calculations.

First divide the number of index entries in a sector into the number of records. In our example, this calculation is:

The result of this calculation (240 in this example) gives how many sectors are needed for the index. This result must then be converted to tracks.

Since there are 24 sectors in a track, to find the number of tracks required, divide the number of sectors by 24.

In this example, there isn't any remainder. However, if your calculation results in a remainder, add one track to your answer. Otherwise, not enough space will be reserved for the index.

Finally, for an indexed file, add the number of tracks required for the index to the number of tracks required for the records of the file. In our example, the sum is 150 tracks.

Deciding Where to Locate the File

After you determine the amount of space the file requires, you can decide where to locate the file on the disk. Since a disk can contain several files, depending upon their size, it is good practice to document what files are on which disk.

The Disk File Layout Chart (Figure 3) is available for this purpose. The Disk Layout Chart shows space available on the fixed and removable disks. There are 406 positions (0 - 405), represented on the chart. Each position corresponds to a track. In Figure 3, notice that tracks 0 through 7 have a line through them. These tracks are reserved for system use only and are not available for data files.

As you create more files, you can refer to the chart of a particular disk to determine the amount of available space on that disk. It is helpful then to indicate the required space for each file on a Disk Layout Chart. It is also helpful to indicate the name of the file on the chart. Figure 4 shows the space and location of the name and address file using the indexed method. The calculations performed to determine the amount of disk space can be documented on the back of the chart.

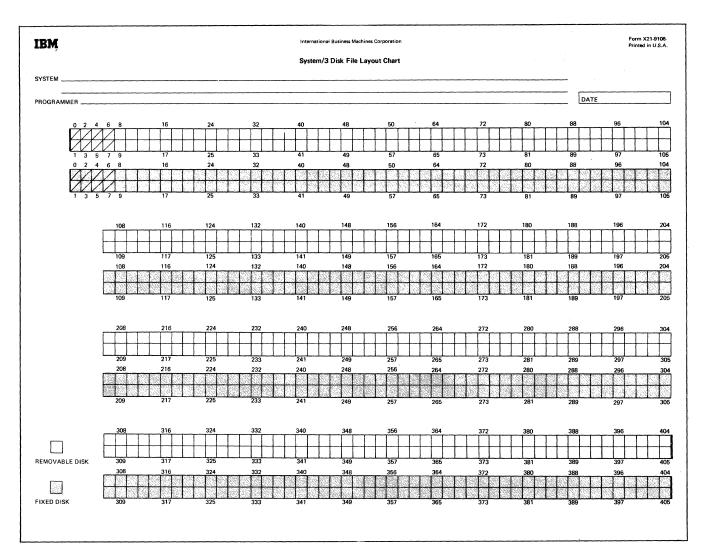


Figure 3. Disk File Layout Chart

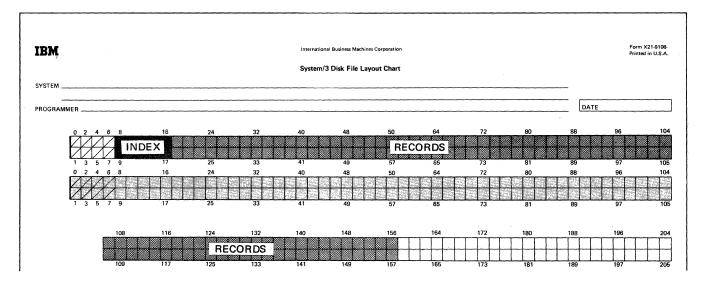


Figure 4. Disk Layout for an Indexed File

REVIEW - CALCULATING FILE SPACE

Calculation 1: Record Space

To calculate the number of tracks required for the records of a file, the following steps are necessary:

- 1. Multiply the number of records by record length to get total number of characters.
- 2. Divide total number of characters by 6,144 (the number of characters in a track) to get number of tracks.
- 3. Round to nearest whole number, if a remainder results.

Calculation 2: Index Space

To calculate the number of tracks required for an index the steps are:

- Add 3 to the key field to get the length of index entries.
- 2. Divide 256 by the entry length (a sector contains 256 characters) to get the number of entries per sector.
- 3. Drop the remainder from step 2. (Complete entries must be contained in a sector.)
- 4. Divide the number of records by the result of step 3 to get number of sectors.
- 5. Divide this result by 24 (the number of sectors in a track).
- 6. Round to the nearest whole number, if a remainder results.

Calculation 3: Total Space for an Indexed File

Add the results of Calculation 1 and Calculation 2 to get the total number of tracks needed for an indexed file.

CHAPTER REVIEW

During the planning stage, but after the file organization method is chosen, record and disk layout are designed.

When designing your record, you should determine:

- Field size and location
- Field names
- Record length
- File name
- Block length

Remember, if you assign field names at this time that follow the RPG II rules, your RPG II program will be easier to write.

Use the Disk File Layout Chart to document:

- File size and location
- File name
- Size calculations (optional)

This chapter explains how to use the RPG II language for indexed and sequential disk files. To understand the material in this chapter, you should know how to organize, maintain, and process these two types of files (Chapter 1). The first and major part of this chapter explains how RPG II is used to:

- Create a file
- Maintain a file

Seven sample jobs illustrate these functions. The last part of the chapter contains three sample processing jobs.

To understand the sample jobs, a basic knowledge of RPG II is also necessary. Specifically you should know how to:

- Describe files
- Describe records
- Use matching records
- Use control levels
- Use mathematical operations (ADD, SUB, Z-ADD)
- Use indicators on Input, Output, and Calculation sheets (MR indicator, control level indicator, record identifying indicator, and resulting indicators).

In the sample jobs these RPG II topics are not explained, except where they relate to disk topics. If you do not fully recall and understand some of these topics, you should refer to IBM System/3 Disk System RPG II Reference Manual, Form C21-7504.

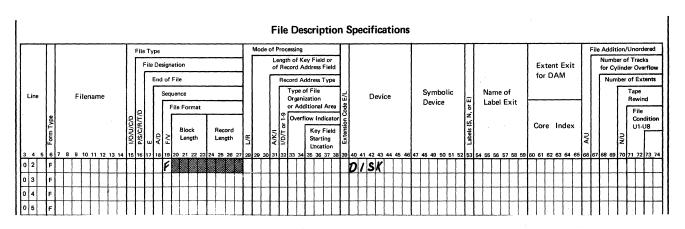
CREATING A DISK FILE

The RPG II language used with the System/3 Disk System can create sequential and indexed files. Records in the desired sequence are read from card (or disk) input files, and the disk records are written. Since several required RPG II specifications are common to both sequential and indexed files, these specifications are explained first, followed by two sample programs. One program explains how to create a sequential file; the other explains how to create an indexed file. The uncommon RPG II specifications, those required for a sequential file or for an indexed file, are explained before each program.

To create either type of file, you must write certain entries on the File Description sheet. Four entries are required to describe various characteristics of the disk file. These entries are coded in the following columns:

File Format	column 19
Block Length	columns 20 - 23
Record Length	columns 24 - 27
Device	columns 40 - 46

All records in a file must be the same length. Thus, an F is required in column 19; it specifies that records are the same length (fixed). Columns 20 - 27 are shaded to indicate that block length and record length vary depending on the records in your files. However, block length is always a multiple of record length. If records are unblocked (one record transferred at a time), you make the same entry in the *Block Length* and *Record Length* columns. For example, if the length of a record is 128, 128 is entered in columns 21 - 23 and in columns 25 - 27. Whenever a disk file is described, DISK is required in the *Device* columns.



Creating a Sequential File

To create a sequential disk file no additional disk entries are required. The four entries explained, in addition to other basic RPG II entries, are sufficient to create a sequential file. Job 1 (Figure 5) explains the coding needed to create a master name and address file from a card file.

Note: Job 1 includes the layout of the input cards, the layout of the disk records, and the coded RPG II specification sheets needed to create the name and address file. The disk entries explained previously are emphasized on each sheet. Those RPG II entries, such as defining fields, that are assumed prerequisites are not re-explained. However, some of these entries, those crucial to correctly performing the job, are discussed. This method is used in all sample jobs in this chapter.

Creating an Indexed File

To create an indexed file, four more File Description entries are required. Two entries tell the RPG II Compiler that an indexed file is to be created: an A in column 31 and an I in column 32. The I specifies an indexed file; the A specifies that a key field exists.

The other two entries describe the length and location of the key. Key length must be entered in columns 29 - 30. Recall that key length is the same length for all records in a file. The maximum key length is 29 characters. Columns 35 - 38 are used to specify the location of the first character of the key. For example, if a key field is located in columns 73 - 78 of a disk record, you would enter a 6 in column 30 and a 73 in columns 37 - 38.

Job 2 (Figure 6) shows the disk entries, and other RPG II entries, needed to create an indexed inventory file.

IBM																										-	nte	rnat	iona	l Bu	usin	ess N	lach	ines	Cor	pora	tior					_							_															1-90	
22														R	P	3	(c	N.	TF	36)L	. ()Α	R	D	<i>,</i>	۸N	ID	F	-11	LE	C	ÞΕ	S	CF	RIF	T	IC	N	S	PI	EC	:IF	10	CA	T	0	NS															n U.S	
Date																							Pu	ınd	ning		1	Gra	ohio			7		T			T	7		Τ]						,	Page	Γ	1	2				ogra		ion	75	7	6 7	7	78	79	80
Program																							In	stru	icti	on	F	un	ch		I			I			I			I										L					IUI	511(1)	iica	ion	_					_	
Programme	er_																																																																
																									C	OI	nt	ro	1 (Са	rc	S	pe	ci	fic	cat	tio	ns	;																										
Line F	Form Type		e t	o iife	Object	Listing Option	Exe	e to ecute	Debug	1	1	InputPence	Output-Shillings ui	Output-Pence	Inverted Prin	% 360/20 2501 Buffer	Of Pos	Prin	t la Coronada	Alternate Collating Sequ	7 2	18 2	9 3	80 :	31 3	32	33	34	35	36	37	38																			tual			62	63	64	65	66 6	7 6	8 6	9 7	0 7	1 72	2 73	74
0 1	н					T		1	T	T	T	Γ	T	T	Γ		1	T	T	T	T	T	T	T	T	1	1		1	1		T	T		T	T	T	Γ	Γ	Γ	T		Γ						T	Ι	Ι	Γ						I	Ι	Ι	I	Ι			
										Fi	le T	e D	esig		e							M	lode	e of	Pro Len	oce igth Rec	ord		y F Idre	ield ss F	or Fiel	d	S	ре			,		ns				nbo	lic				Van	ne (nf			Ex					Fi	N fe	um or C	ber ylin	of 1 der	Ovi	ks erflo	w
Line	Φ Form Type	7	8		10 °			3 14	- 1	10/0/0/0/0 = 10/0/0/0/0		1	200	File	Blo Lei	gth		L	ecor engt	th	7 2	7	9 3	- 1	A/K/I	I/D/T or 1-9	Ov	aniz Add erfl	ow Ke St	Ind ey f arti	licar Fiel ing tion	tor d	& Extension Code E/	40		De\ 12 4			5 4	6 47	C	Dev	ice			없 Labels (S, N, or E)		Lab	el 8	Σxi			Cor					A/U	7 6	8 6	9 7		U1-	diti U8	1
	F					T	T	T	T	T	T	T	F					ı			T				A	1									1				T	T	T	T	T			П		T	T	T	T	T	Γ					T	T	T	T	T	T	T	П
0 3	F					J		I	T	I	T	I	Ī	Ī							1	ľ		Ī	1								j				I	I	I	I	Ī	I	I	I					I	I	Ι	I									I	I	I		
0 4	F					I	I	I	I	I	I		I	I							I		I													I	I	I	I	I									I	I	I							I	I	I	I	I	Ι		
0 5	F						I											\rfloor							1								\rfloor	\rfloor	\rfloor	\perp							Ī	L																	1		L		
0 6	F		L					1	1	1				1							1	1		1									1			1	1	1		L				L	L				1	1				L					1	1				L	
0 7	F																																												L																				Ш
	F					1	7	+	ľ	Ŧ	Ŧ	Ŧ	T	F			1	1	7	Ŧ	Ŧ	T	$\frac{1}{1}$	1	1								\exists	-	-	Ŧ	Ŧ	Ŧ	I	Ŧ	T	Ŧ	Ŧ					7	+	Ŧ	T	Ŧ	F	-				-	Ŧ	Ŧ	Ŧ	F	F	F	H

MAINTAINING A DISK FILE

In Chapter 2, four file maintenance functions were explained. These four functions apply to both sequential and indexed files. RPG II provides the capability to perform these functions. This section explains the RPG II coding required to do three of these functions: add a record, update a record, and reorganize a file. In order to explain the coding needed for these three functions, the sample jobs show the functions performed separately. However, you may write a program that performs more than one of these maintenance functions. Since the delete function is usually performed during normal processing, and it does not require new entries, this function is not explained in this section.

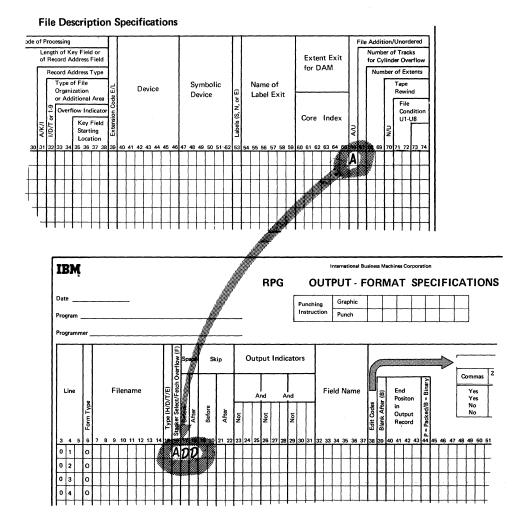
Adding Records to a Disk File

After a file is created, it is often necessary to add records to the file. Recall that adding records to a disk file means writing the records at the end of the records already in the file. To add records to an indexed or a sequential file, two more disk entries are needed, one on the File Description sheet and one on the Output-Format sheet.

The A in column 66 on the File Description sheet tells the system that records will be added to the file described on this line. ADD tells the system that the fields defined on the following lines constitute the record to be added to the file specified in columns 7 - 14.

Adding Records to an Indexed File

Job 3 (Figure 7) shows the RPG II coding required to add records to the master inventory file (indexed) created in Job 2 (Figure 6). Note that the card layout and the disk record layout are the same for both jobs.



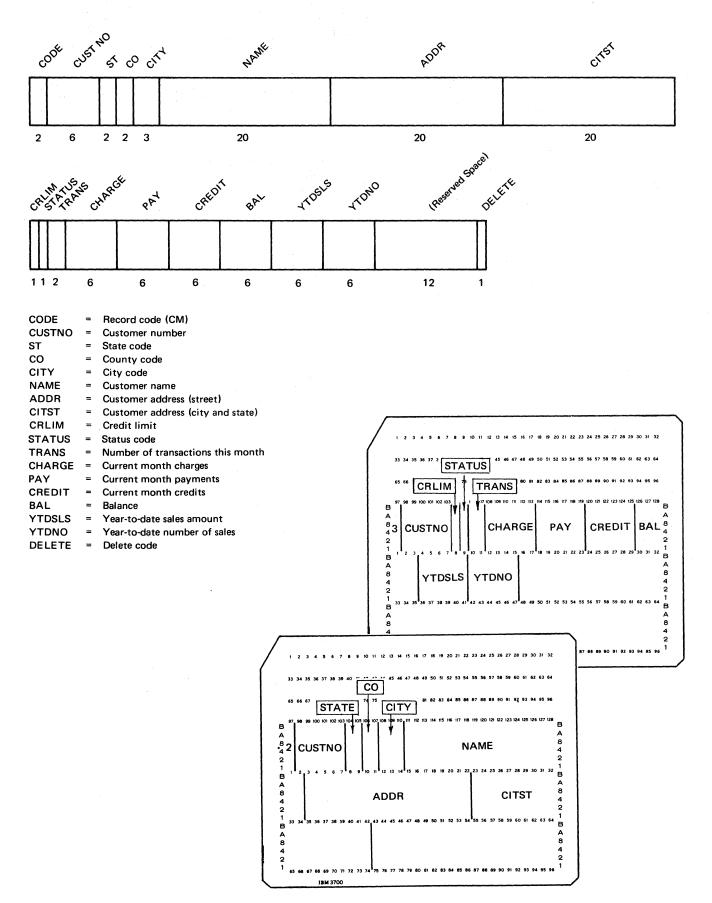
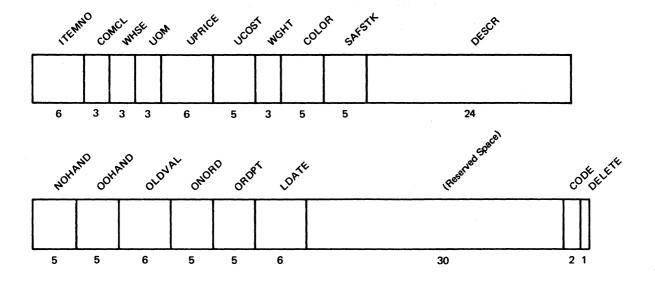


Figure 5. Job 1: Creating a Sequential Customer Master File (Part 1 of 2)

File Description Specifications

3 0 0	5	т т т в Form Type	CA SE	Å	DI	1 12 N		14	Q/2/n/o/1 15 1 1 1 0	6/1/N/O/S/1	Enc 7 18	i of Sequ	B L	e	h 2 23	Le	core	27	M/J 28 2	9 30	Lei	Rec	ord A ord A Type Organ	Add Add of niza dditi		Area licato	S Extension Code E/L	M		42 4 C I	11	4 45	46 4	D	evi			∑ Labels (S, N, or E)	L	.abe	e of I Ex	it		Core	DAI	Exit M		Г	Num for (Cylindry and the control of the cylindry and the cylindry	Tap Rev	e wind ile condit	ow its
0	_	F				+			+	1	<u> </u>		4	-		+	-	 	-	-		-	+	+		+	+	\vdash	H		+		+	+			+	-	H	+	$\ $	+	+	H	+	$\ $	+	+	H	+	\mathbb{H}	+	Н
		F				-			$\left\{ \right.$	1	ľ) = 		cke (6)	u . L				-	Ŧ			T	F			-						-	F		Ŧ	H		H	П	Ŧ	\blacksquare	Ŧ	Ŧ		Ŧ	П	7	$\exists I$
IB	M	_		==			_								_				=			=	Inte	rnat	ional	Busin	iess N	fachi	ines (Corpo	oratio	on .			_					_				=				==				X21-6	
Date Prog Prog		ner _																		Pur	RPC nchir	 1g	Ţ		PU'	T 5	SPI	EC	iF	IC	AT	ΓIC	N	S]						Page	- [rogra dentif		ion	75	76 7	" <i>"</i>	78 79	80
Li		Form Type			ena				Sequence	N	Option (0)	Record Identifying Indicator			sition	No + ON	C/2/D	Character	Po	sitio	2 n	Not (N)	C/Z/D	Character	Posi	tion	4) 4-14	ı	Character	Stacker Select	- 1	Fr	om	Lo	-	Го	d	Decimal Positions			ame		Control Level (L1-L9)	Matching Fields or	Chalning	Field Record Relation	Plu	M a	licat	Zero or Blan	*	Sterli Sign Positi	ion
3 0 0 0 0 0 0 0 0 0 0	O to	I I I I I see ear	ine:	R s 0 enc	1 a	and cho	d to	4, k ti yp	Ø :	2 du in 2 r	mr	3 ns t c st	Ø 15 care be	rds.	17 Ir	1 cor	D Inta	3	inf inf b	fori	ma an car	nti d d	on 16 typ	u: , Coe ha	sed 01 3		38 3	9 40	0 41	42	43 4	44 48		2 8 2 8		7	7 (4	000	:U	S RI	۲N	0	59 60	61	62 6	3 64		56 6.	7 68	69 7			3 74
IE Date Prog	_	ner_												.,,,,			F	PG	ì		O	ing	TP	U	T -								IFI	CA	Т	10	NS	•			Page	1	2			Progr Ident		tion	75		rinted	X21-94 d in U. 78 79	S.A.
	ne 4 5	 Form Type 	7 8			ime	2 13		Type (H/D/T/E)	Safora	1	Dofesso	. Skip	After	Not		And		And	 					ame		Blank After (B)	Didnik Aiter (D	in Out	iton tput cord	P = Packed/B	r = racked/b =	46		Ye Ye Ne	es es o		Y N Y N	es o es o Cor	es	No Si	ign Ed	CR A B C D	J K L N	ч П	Y = Z =	Piel Zero Sup	s Sign e Id Ed o oppress	n lit s	69 70		Sterlin Sign Positio	on
0 0 0 0 0	1 2 3 4 5	o o o o o o o o o o o o o o o o o o o	inc dis ntil 0 s	e k	D ec ec erd	h cores	ard, v	d to	yr de is th	pres	ar 't v	e i	need nt sed	ede to i. i ord	ed w nd	to v	wr e it too	ite				C	US AF	51	-N						2 8 5		CICO	ν, Λi	s a is	dd a i	ed	to	th	e c	lisk e th	c re	eco ca	ord.	. 1	This use	s ed 1	to					

Figure 5. Job 1: Creating a Sequential Customer Master File (Part 2 of 2)



Although 95 positions are presently used, the disk record is defined as 128 positions to provide space for adding data later. Field names are the same as the input card field names, with two exceptions: CODE = MI (Master Inventory) and DELETE (position reserved for a delete code).

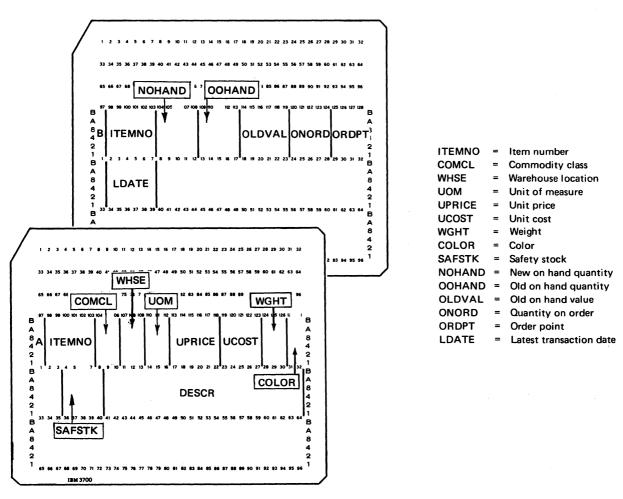
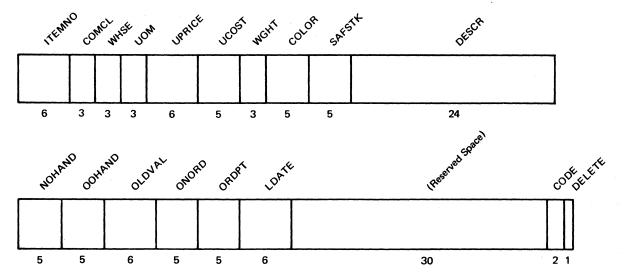


Figure 6. Job 2: Creating an Indexed File (Part 1 of 2)

IBW	International Business Machines Corporation	anti automosti cominina e e e e e e e e e e e e e e e e e e e	Form X21-9094 Printed in U.S.A.
Date	RPG INPUT SPECIFICATIONS	1 2 Page 0 2 Program	75 76 77 78 79 80
Program	Punching Graphic Instruction Punch	rage V Z Identifica	tion
Programmer			
D Record	I Identification Codes Field Loca	ition	Field
l light	2 3		Indicators
Form Type Sequence Number (1:N) Option (0) Record Identifying Indicator Oction (0) Ci2/D Ci2/D Character Character	alect d/B = Bi	Tield Name 5 5 2	Sterling Sign Zero Position
Sequence Number (1: Option (0) Record Ider CIZID CIZID CHARACTER C	Not (N) Not	Control Le Control Le Control Le Control Le Chelning Field Recc	lus Minus or Blank
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 2	3 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48	49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	66 67 68 69 70 71 72 73 74
		64 DATA1	++++++++++++++++++++++++++++++++++++
0 3 1	2	70 ITEMNO	
0 4 T Ø21 2 Ø 1 C B	 	7 OF TEALO	++++++++++++++++++++++++++++++++++++
0 6 1	8	39 DATA2	
0 7 I			
			ĺ
	File Description Specifications		
File Type	Mode of Processing Length of Key Field or		File Addition/Unordered Number of Tracks
File Designation	of Record Address Field Record Address Type	Extent Exit for DAM	for Cylinder Overflow Number of Extents
Line Filename Sequence	Type of File Syr	mbolic Name of	Tape
File Format	or Additional Area	vice Label Exit	Rewind
Block Record Length Length Length L	Overflow Indicator	Core Index	Condition U1-U8
	Starting Location	[-]	A/L
0 2 F CARDIN IP F 96 96	29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 43 46 47 48 4	9 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	66 67 68 69 70 71 72 73 74
03 F I NV MSTR 0 F 256 128	06AI 1 DISK		
0 5 F			
0 6 F	Indexed File Key field		
0 7 F			
IBM	International Business Machines Corporation		Form X21-9090
RPG	OUTPUT - FORMAT SPECIFICAT		Printed in U.S.A.
Date	Punching Graphic	Page 1 2 Page 1 3 Program Identific	
Program	Instruction Punch	ldentific	ation
Programmer			
Space Skip Output Indi	cators	Edit Codes	
Overfil		mmas Zero Balances No Sign CR - X = Re	
Line Filename (1) 12 And A	nd Positon	Yes Yes 1 A J Y = Da	is sign Cian
orm Type Ser Select. Before Before After Not	After a sed/8	No Yes 3 C L Z ≈ Zei	ro ppress
L Sell	Record B	Constant or Edit Word	
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 28 28 28 27 28 0 1 0 1 0	29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48	49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	66 67 68 69 70 71 72 73 74
0 2 0	DATA1 63		
0 3 0 Indicator 20 specifies that a disk record	DATA2 95		
is to be written after every card type B.			
Since both card types are needed to write a disk record, we don't want to		eld names are usedach card, data from	
0 7 0 write the disk record until card type B	both cards is ava	ilable at the time	
0 8 0 is processed.	the disk record is	s written.	
0 9 0			

Figure 6. Job 2: Creating an Indexed File (Part 2 of 2)



Although 95 positions are presently used, the disk record is defined as 128 positions to provide space for adding data later. Field names are the same as the input card field names, with two exceptions: CODE = MI (Master Inventory) and DELETE (position reserved for a delete code).

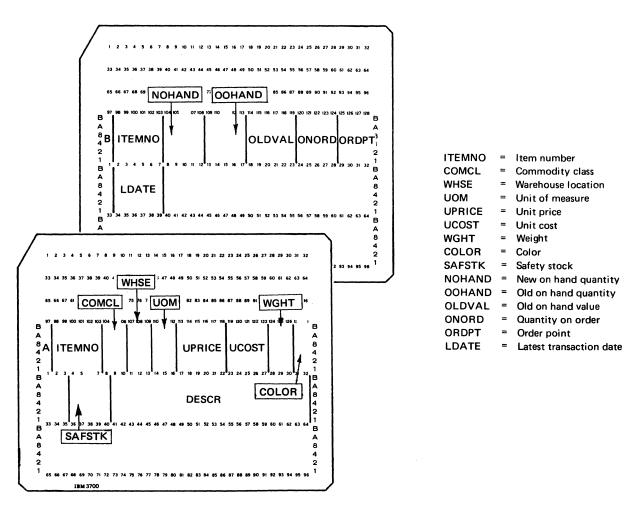


Figure 7. Job 3: Adding Records to an Indexed File (Part 1 of 2)

	File Description Specifications	
File Type File Designation End of File Sequence File Format Block Length Length Length Length 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 2 0 2 F CARDIN IP 0 3 F J NVM5 TR 0 6 F 0 7 F 1 F	Mode of Processing Length of Key Field or of Record Address Field Record Address Type Type of File Organization Or Additional Area Overflow Indicator of Starting Starting Location Location 12 2 2 3 2 3 3 1 3 2 3 3 3 4 5 38 3 7 3 3 39 40 41 42 43 44 45 46 47 48 49 50 51	Extent Exit for DAM Name of Label Exit Core Index 2 2 2 2 3 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
IBM Date Program Programmer	International Business Machines Corporation RPG INPUT SPECIFICATIONS Punching Graphic Instruction Punch	Form X21-8094 Printed in U.S.A. 75 76 77 78 79 80 Page
Frim Type Sequence Sequence Option (D) Record Identifying Indicate (1.N) Phot (N) Not	Cord Identification Codes 2	Field Name
3	2 6	4 DATA1
IBM, Date Program Programmer	Punching Graphic Punch P	Form X21-9090 Printed in U.S.A. 1 2 75 76 77 78 79 80 Page 75 76 77 78 79 80 Identification
Form Type Type (H/D/T/E) Backer Select/Fatch Overlt After After Not Not	Field Name Field	Edit Codes
3 4 5 6 7 8 9 10 11 12 13 14 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	DATAI 63 DATA2 95	1 52 53 54 55 56 57 58 59 60 61 52 63 64 65 66 67 68 69 70 71 72 73 74
0 8 0		

Figure 7. Job 3: Adding Records to an Indexed File (Part 2 of 2)

Adding Records to a Sequential File (Added at End of Records in the File)

Records can be added to a sequential file in the same way they are added to an indexed file: records are added at the end of the file. This method of adding records to a sequential file is good for files such as the employee payroll file explained in Chapter 2. Recall that this file was organized in ascending order according to employee number. New employees are assigned the highest available number. Thus, when records are added for each new employee, the records will be in order on the disk.

To add records to this type of sequential disk file, the four basic disk entries and the two add entries (A,ADD) are required.

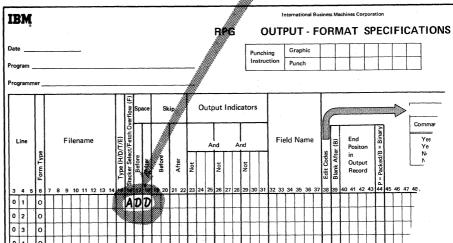
Adding Records to a Sequential File (Merging Records Between Records in the File)

Often records must be added *between* existing records in a sequential disk file. In Chapter 2 this situation was mentioned, and you were told these additions could be handled in an RPG II program. When records must be added between existing records, a new file must be created. This file will contain the added records merged in correct order with the records from the original file.

More disk space is required for this method of adding records. Space must be available for the old file and the newly created file.

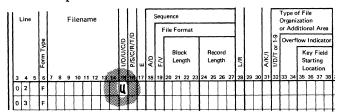
Job 4 (Figure 8) illustrates how to add records between existing records of a sequential file.

File Description Specifications File Addition/Unordered Mode of Processing File Type Length of Key Field or of Record Address Field Number of Tracks File Designation Extent Exit for DAM End of File Number of Extents Record Address Type Type of File Tape Rewind Symbolic Name of Device Device Label Exit File Format flow Indicat Condition Core Index Key Field Starting DISK



Updating Records in a Disk File

Many jobs not only require the ability to add records, but to change certain data in a record. Recall that this function is called updating. RPG II provides several ways to update both sequential and indexed files. The next four jobs illustrate four ways to update disk files. Regardless of which way it is used, one new disk entry is required on the File Description sheet.



The U in column 15 indicates that the file is an update file. Since updating means getting a record from a disk file, changing some data, and then putting the record back in its original location, an update file is like a combination input/output file. For this reason, the update file must be described on both the Input and Output sheets.

Field locations must agree between the two sheets. The *To* location, columns 48 - 51 on the Input sheet, must be the same as the *End Position*, columns 40 - 43 on the Output sheet. Field names may vary depending on the type of updating being done.

IBW	International Business Machines Corporation RPG INPUT SPECIFICATIONS	Form X21-9094 Printed in U.S.A.
Date	Punching Graphic Instruction Punch	1 2 75 76 77 78 79 80 dentification
Programmer		
1 Indicator	Record Identification Codes 2 3	ndicators
	Position (N) 190 Positi	Control Matchi
0 1 1 0 2 1		16 YTDSLS
0 3 1 0 4 1		10000
• • • • • • • • • • • • • • • • • • • •	11111111111	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	Must agree	May vary
IBM,	International Business Machines Corporation RPG OUTPUT - FORMAT SPECIFICATIO	Form X21-9090 Printed in U.S.A.
Date	Punching Graphic Instruction Aunch	1 2 75 76 77 78 79 80 Page Program Identification Page Program Page Program Page Program Program
Programmer	E Prich	
Form Type Page Pa	No N	Edit Codes Sero Balances No Sign CR - X = Remove Plus Sign Yes 1 A J Y = Date Field Edit Yes 3 C L Z Zero Zero No 4 D M Zero Zero Suppress
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	YTDSLS 46	

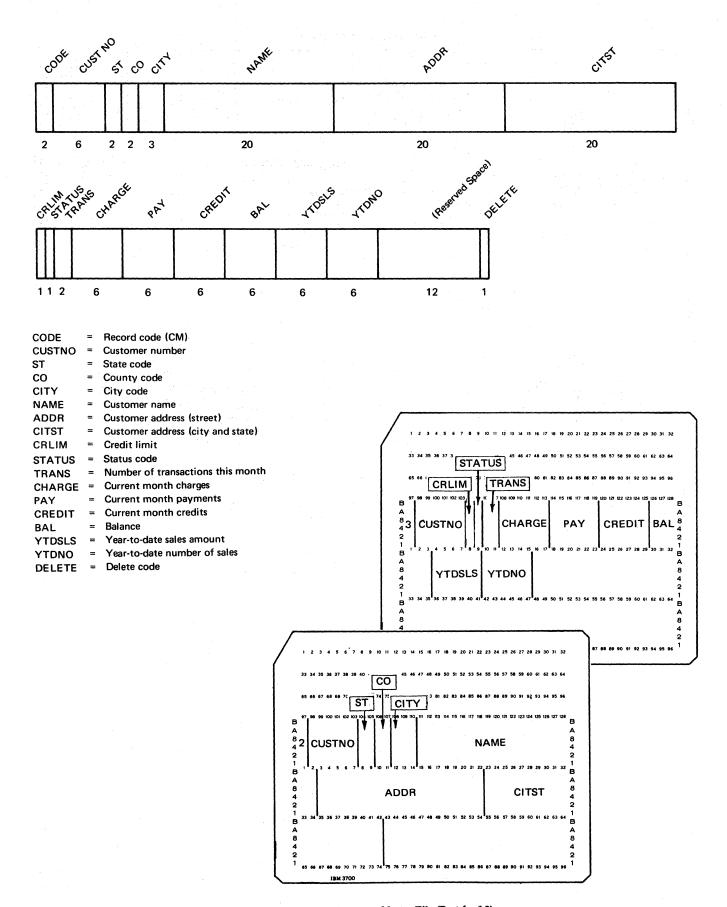


Figure 8. Job 4: Adding Records to (Re-creating) a Sequential Customer Master File (Part 1 of 2)

1		File Description Specifications
Line 3 4 5	File 1 Fi	e Designation Concept Concept
0 2 0 3 0 4 0 5	FCARDIN IP FSEQDISK IS FNEWSEQ O	F 128 128 D15K D15K
Date		RPG INPUT SPECIFICATIONS
Program		Punching Instruction Program Instruction
Programm	ner	
Line	Form Type Egeuence Mumbe (14M)	Record Identification Codes 1 2 3 Field Location Position (N)
3 4 5	I CARDIN AA	7 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 74 74 74 74 74 74
0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9	To add the	### 2 #### 2 #### 2 ######
	columns 61	
		RPG OUTPUT - FORMAT SPECIFICATIONS
Program Programn	ner	Punching Graphic Page 03 Program Identification
Line	Form Type Fillename Fillename Fillename Fillename Fillename 19 89 10 11 12 13 144 7. 8 9 10 11 12 13 144	Commas Zero Balances No Sign CR - X = Remove Plus Sign Sign Sterling Sign Position Yes No 2 B K 7 - 7 - 7 - 7 - 7 - 7 - 7 - 7 - 7 - 7
0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 0 1 1 1	NEWSEQ D	When indicator 02 is on, the existing disk record is written on the disk. When indicator 03 is on, the existing disk record is written on the disk. Since we want to add records (from cards) between existing disk records and M1 ensures that the input records are processed in sequence, the output records will be in sequence according to customer number

Figure 8. Job 4: Adding Records to (Re-creating) a Sequential Customer Master File (Part 2 of 2)

Updating All Records in a File

All or part of the records in a file may be updated. Let's discuss updating all records first.

All records may need updating for a variety of reasons. For example, you might add a new field to all records. If the record were originally designed with expansion in mind, the field can be added with the update function. For example, a customer file was created without zip code; then zip code was required. It can be added as an update field. In this case an input file (probably cards) is processed against the master disk file which is specified as an update file.

In another case you might want to change some data in all the records as a result of calculations, not as the result of an input file. For example, payroll records contain year-to-date information. After all reports and W-2 forms are completed at the end of the year, the year-to-date fields must be set to zeros, so accumulations can begin for the new year. In this type of job, you have only one file, the update file. Job 5 illustrates this type of updating.

Updating All Records in a Sequential File: We discussed consecutive processing of sequential files in Chapter 2. Recall that with consecutive processing all records are read in order, one after the other. To update each record in a sequential file, the four basic disk entries plus U in column 15 are needed on the File Description sheet.

Job 5 (Figure 9) shows the RPG II coding required to update all records of a sequential customer credit file. Two fields in each customer's record contain year-to-date sales data. These two fields indicate the activity of each customer. Throughout the year this data is accumulated. At the end of the year the two fields are set to zero, so the accumulation can begin for the next year. Only one file, the update file, is used in this job since updating occurs as a result of calculations.

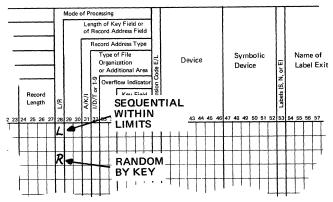
Updating All Records in an Indexed File: All records in an indexed file can also be updated. Recall that to process all records in an indexed file, the file is processed sequentially by key. Thus, to update all records in an indexed file, the entries required to describe an indexed file plus U in column 15 are the disk entries needed on the File Description sheet.

IBM					International Business Ma	chines Corporation				Form X21-9092
IDM		F	PG CONTE	OL CARI	AND FILE	DESCRIPTIO	N SPECIFI			Printed in U.S.A.
Date	·			Punching Instruction	Graphic		T	Page	Program Identifica	75 76 77 78 79 80 tion
Program					Punch					
Programme	r			Co	ontrol Card Sp	ecifications				
Line F	1 111 1	MFCM Stacking Sequence Input-Shillings Courput-Pence Output-Pence	Inverted Print 360/20 2501 Buffer 360/20 2501 Buffer Sc. 2 360/20 Sequence	7 28 29 30 31 32	233 34 35 36 37 38 37			ibrary manual for actual		66 67 68 69 70 71 72 73 74
0 1	4									
		File Type File Designat		Mode of Prod Lengt of Re	cessing th of Key Field or ecord Address Field	Specifications			Extent Exit for DAM	File Addition/Unordered Number of Tracks for Cylinder Overflow
Line	Filename	End of Fi		11 17	Type of File Organization	Device	Symbolic Device	Name of		Number of Extents Tape Rewind
, i	- i 3	1/O/U/C/D P/S/C/R/T/D E A/D F/V	Block Record Length Length	L/R A/K/I I/D/T or 1-9	Overflow Indicator Key Field Starting Location			Labels (S, N, or	Core Index	File Condition U1-U8
0 2	6 7 8 9 10 11 12 13 14	17 18 19 2	21 22 23 24 25 26 2	28 29 30 31 32 A I	33 34 35 36 37 38 39	D/SK	6 47 48 49 50 51 52	2 53 54 55 56 57 58 5	9 60 61 62 63 64 65	66 67 68 69 70 71 72 73 74
11111						0/01	 	 	 	++++++++++++++++++++++++++++++++++++
0 4	:	11111		11111			<u> </u>	111111	11111	
0 5 1	F									
0 6	F	11111						111111		
0 7		44444		11111	\Box					
		+111	+++++	++++	+++++	\blacksquare	+++++	++++++	+++++	

Updating Some Records in an Indexed File

An indexed file can also be updated sequentially within limits or randomly by key. In addition to those disk entries required for an update indexed file, one other entry is needed on the File Description sheet. This entry specifies how a file is processed. For indexed files, a file can be processed sequentially by key, sequentially within limits, or randomly by key. The entry is made in column 28.

File Description Specifications



For indexed files, if column 28 is left blank, the file is processed sequentially by key. If any of your files must be processed sequentially within limits, refer to the *Disk System RPG II Reference Manual*.

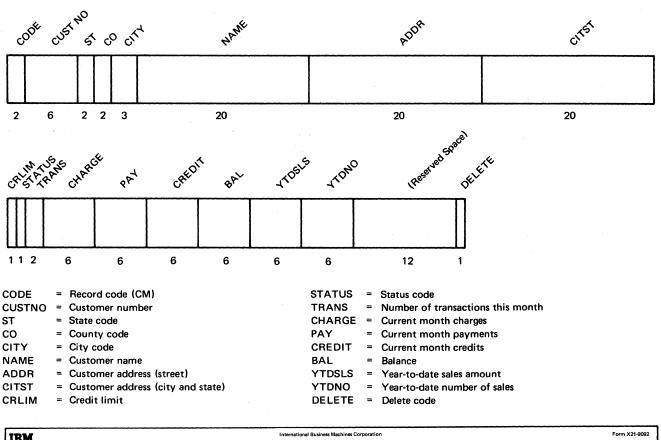
Updating Records Randomly by Key:

When only some records in an indexed file must be updated by input file (transaction), the indexed file can be updated randomly by key. Indexed files are usually updated randomly by key when the input file is unordered or activity is low. Recall that low activity means only a few records of a master file are affected. Regardless of whether the input records are in order or not, random processing by key provides the ability to update certain, not all, records.

When an indexed file is updated randomly by key, input records are *chained* to update records. Chaining means matching the control field from the input record with the key in the index.

When records are updated randomly by key, the CHAIN operation code must be used. To use CHAIN, you must make five entries on the Calculation sheet.

IBM																_					_					In	tern	atio	nal	Bus	ines	s Ma	chi	nes (orp	orat	ion	_																				_					X21		
																				1	RF	G	i	C	;Δ	۱L	С	U	L.A	۲,	ΓI	or	v :	SP	E	CI	FI	C.	A.	TI	O	N	S																				d in		
Date																						Г			_	7	-		hic	_	_	Т		Т	7		Т	7	_	_		7								_	T	2		F	rogi	ıram		1	75	76	<u>77</u>	/ 7º	8 7	79 8	0,
Program_																								ruc		٠		nc		-		\dagger		H	1		t	+		+		1							Pag	e L	L				den			n [Ĺ	L	L	L	L	_	الــ
																_						_						_					_									J																							
Programm	ier_		_					_	_	_	_			_	_	_				_					_																_	_		_	_						_														_
					in	dica	ator	rs																	-																					L			ato		╝														
		_	一		T			Γ	_											١					1																				S	F	lus	Mi	nus	Zer	0														
Line	e e	Control Level (LO-L9, LR, SR)			And		Α	l ind						Fa	act	or '	ı				O	pe	rat	ion	.				F	act	tor	2				F	Resu	ılt	Fi	eld		Fi	ield engt	h	Positio	1	ligh	Con	par	Equ	ıal						Cor	mn	nen	ts					
	Form Type	trol (L	1 1	4	_	_	L	_	\dashv										1					1																				ima	1	>2	1<	2	1 =	2														
	For	8 3	ş		Ž			Š												١																								1	ة اة	-	able	Loc e (F	kup	r 2)	is														
3 4 5		١, ,		10		, ,	14	16	16	.,	10	10	20 -	21 ·	າາ	33 .	24 -	ır.	ne '	,,	, o	, o	on .	21 '	,,	22	24	25	26	27	20	20	40	41	42	42		AE.	AC.					- 1	- (- ; •	ngn	10	JW	Equ	iai į			2 6	2 6.	A C:		67	- 60	60	70	71	70	72	,
	С	Ť	Ť	1	Τ	1	T	۳	10	Ï	٦		7	Ϊ	٦	٦	Ť	٦	1	Ť	٦		1	Ī	٦	٦	34	33	30	Ť	1	35	Ī	T	12		Ϊ	1	40	ΪÍ	40	100	Π̈́	Ť	32 3	3 3	T	150	5,	30	7	T	T	Ť	Ť	Ť	T	ř	n	Ñ	ñ	ĤΤ	٦	٦	٦
0 2	С	П	T		T	T	T	T	Γ												C	Н	A	1	N											П	П	٦				Г	П	1	1						1	7	T	T	T	T	T	Г	П	П	П	П	Π	7	٦
0 3	С	П	T	П	T	T	T	Γ	Γ					Ī	1	T	T	1	1	٦													Ī	Γ			П						П	1	T	T	Ī	T			1	T	T	T	T	T	T	Γ	П	П	П	П	П		7
0 4	С	П	Т	П	T	T	T	Γ	T					Т	7	T	T	1			1	1								Γ	Γ	Г		Γ				1						1	T	T	T	T					T	T	T	T				П	П	П			7
0 5	С					Ι																																								I								I		\perp	L	L							
0 6	С					L																								L	L	L		L								L				l							1	\perp	\perp	\perp	L	L		Ш	Ц	Ц			
0 7	С	Ш	L		\perp		L	L									\perp													L	L		L	L	L							L			1	l	L	L					1	\perp	L			L		П					
	С	Ц	L	Ш	\perp	1	L	L	L	Ц						_	\perp							_					L	L	L	L	L	L	L	Ц	Ц					L	Ш	1	1	1		L						1	\perp	1	L	L		Ш	Ц	Ц	Ц		
0 9	С	Ц	L	Ш	\perp			L	L		_			\perp		1	1	1		1					1			L	L	L	L	L	L	L								L	Ц	1	1	1	L	L				_	1	1	\perp	\perp	_	L		Ц	Ш	Ц	Ц		
1 0	С	Ш	L	Ц	1	1	L	L	L	Ц			1	1	_	1	4	1	1	1	4	1	1	1	1				L	L	L	L	L	L		Ц					Ц	L		1	1	1	1	L			_	4	1	1	1	1	1	L	L	Ш	Ш	Ц	Ц	Ц	4
1 1	С	Щ	\perp	Н	1	1	L	L	L	Ц	4		4	4	4	1	4	1	4	4	4	4	1	1	1	4		_	L	L	L	L	L	L		Ц	Ц	4		Ц	Ц	L	Ш	4	1	1	L	Ļ		1	4	4	1	4	1	1	1	L	L	Ц	Ш	Ц	Ц	4	4
1 2	С	Н	-	Н	4	1	1	L	L	Н		_	_	4	_	4	4	4	1	4	-	4	4	4	4	_			_	L	L		L	L	L	Ц	Ц	4				_	Ш	4	4	1	1	L			4	1	1	+	+	1	1	-	L	Ц	Ш	Н	Н	4	4
1 3	С	H	1	\sqcup	4	+	1	L	L	Н	H		4	4	4	4	4	1	4	4	4	4	4	4	4	_	_	_	L	L	-	L	L	L	L	Ц	Ц	4		Н		L	\sqcup	4	4	+	\downarrow	Ļ		1	4	4	+	4	\downarrow	+	+	Ļ	1	Н	Ш	Н	Н	\dashv	4
	С	4	-	Н	+	+	4	1	L	Н	4	_	4	4	4	4	4	-	4	4	4	4	4	4	4	4	_	L	-	L	L	-	L	L	L	Ш	Ц	4		Н	\sqcup	L	Н	4	4	1	+	Ļ		4	4	4	4	+	+	+	+	Ļ	\vdash	Н	Ш	Н	\vdash		4
1 5	С	+	╄	Н	+	+	+	Ļ	L	H	4	_	4	+	4	\dashv	4	4	4	4	4	+	-	4	4	4	_	_	L	L	H	H	┝	L	L	Н	4	4	_		4	H	Н	4	+	+	+	+	_	4	4	4	+	+	+	+	+	_	H	Н	Н	H	Н	\dashv	4
++	C C	\vdash	+	H	+	+	+	-	H	H	-	-	\dashv	+	+	+	+	+	-	+	-	+	+	+	+	-	_		-	F	+	+	L	+	H	H	H	+	_	H	-	\vdash	H	+	+	+	+	+	H	\vdash	-	+	+	+	+	+	+-	+	+	Н	Н	H	H	H	\dashv
H +	С	\vdash	+	H	+	+	+	\vdash	+	H	+	\dashv	+	+	+	+	+	+	+	+	-	+	+	+	+	-	_	<u>_</u>	-	H	1	H	H	+	H	Н	H	+	-	Н	Н	-	H	+	+	+	+	+	H	+	+	+	+	+	+	+	+	+	+	Н	Н	Н	H	H	\dashv
$\parallel + \parallel + \parallel$	С	+	+	H	+	+	+	\vdash	+	H	+	\dashv	+	+	+	+	+	+	+	+	+	+	+	+	+	-		_	-	\vdash	-	+	+	+	-	H	\dashv	+	-	H	\dashv	\vdash	H	+	+	+	+	+	H	H	\dashv	+	+	+	+	+	+	\vdash	+	H	Н	H	H	\dashv	\dashv
$\parallel + \parallel +$	c	\vdash	+	H	+	+	+	\vdash	\vdash	H	1		+	+	+	+	+	+	+	+	+	+	+	+	+	\dashv		-	-	\vdash	+	+	-	H	-	H	\forall	\dashv	-	H	\forall	\vdash	H	+	+	+	+	+	Н	Н	+	+	+	+	+	+	+	+	+	H	H	H	H	Н	\dashv
Ш	Ľ	Щ	丄	Ш	ᆚ	⊥_	\perp	L	L	Ш	┙			_	_	ᆚ		_	_	_		_			_	_		L	L	L	L	L	L	L	L	Ш	Ш			Ш	Ш	L	Ш	J	\perp	L	L	L	L	Ш	_	\perp	┙	丄	丄	丄	_	L	L	Ш	Ш	Ш		\sqcup	



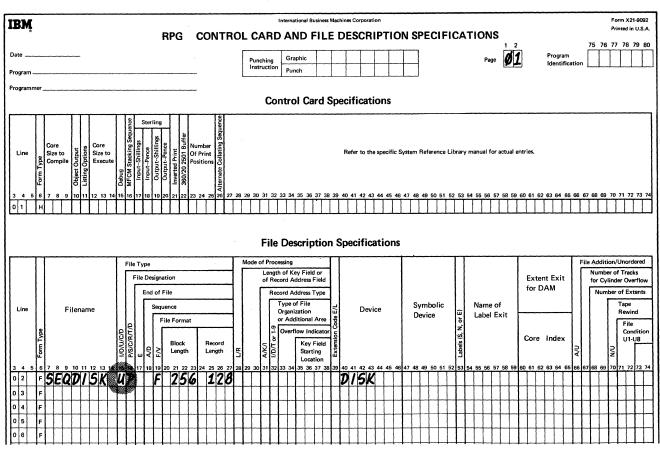


Figure 9. Job 5: Updating Records in a Sequential File (Part 1 of 2)

IBM	International Business Machines Corporation	Form X21-9094 Printed in U.S.A.
Date	RPG INPUT SPECIFICATIONS	1 2 75 76 77 78 79 80
Program	Punching Graphic Page Instruction Punch	ge Ø2 Program Identification
Programmer		
Reco	ord Identification Codes Field Location	Field Indicators
From Type Sequence Problem (b) Record Identifying Indicator Option (b) CiZIO CiZ	2 3 2 8 8	161 151 1
Line Filename	Position Position Position Field Nam	Re Sign Zero Position
Sequence Number (1:) Option (0) Not (N) COZIO	Position No Property	Ontro
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27	28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 5	
0 1 1 SEQDISK AA Ø1	1Ø3 1Ø8 YTDSL	
0 3 1	103 106 YTDSL 109 114 YTDSL	
0 4 1		
IBM	International Business Machines Corporation	Form X21-9093-1 Printed in U.S.A.
	RPG CALCULATION SPECIFICATIONS	1 2 75 76 77 78 79 80
Date	Punching Instruction Punch Page	ge Ø3 Program Identification
Program————————————————————————————————————	1000	
Indicators	Resultii Indicate	
	Arithme	tic
Line e e e e e e e e e e e e e e e e e e	Operation Factor 2 Result Field Field Field Field Field Length 2 Field Length 2 Field Fiel	Equal Comments
Form 1y Not		
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 28	Table (Facts High Low 38 38 38 38 38 38 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 58 55 56 57	Equal
01 C Ø1	Z-ADDØ	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 2 C 0 1	Z-ADDØ YTDSNO	
	set to zero by using Z-ADD. Recall that Z-ADD sets the	
field defined under Result F	ield to zero and then adds the value specified in Factor 2.	·
0 + 0 = 0, in this example)		,
TDM	International Business Machines Corporation	Form X21-9090
IBM,		Printed in U.S.A.
Date	Punching Graphic Page	1 2 75 76 77 78 79 80 Program Identification
Program	Instruction Punch	
Programmer		
Tive (H/D/T/E) State Salest/Fetch Overflow (F) After Part After Part North N	Luit Luit	Codes
Line Filename (3)	Field Name	Plus Sign Sign Sign Y = Date Parision
Type HIDT/TE Select/Fer Party Manual Type HIDT/TE Party Manual Type HI	Position I II Yes No 1 2	2 B K Field Edit
Type (H/) Stocker Sele Before After Before After Not	Record & Constant o	or Edit Word
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27	28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
	YTD5LS 103 YTD5NO 114	
0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Y//UDNU	╎╎╎╎╎╎╎╎╎╎╎ ┼┼┼┪
0 5 0	Update Fields	
0 6 0	1	
0 8 0		

Figure 9. Job 5: Updating Records in a Sequential File (Part 2 of 2)

Under Factor 1 you must enter the name of the control field in the input file. CHAIN is entered in the Operation columns. In Factor 2 you must enter the name of the update file. These three entries are needed for the system to locate the update record. The control field from the input file is used to find the index entry that contains the same field (key). Since this entry contains the disk address of the update record, the desired record can be located and read into storage. The record can then be updated.

The other two entries are needed in case the desired record is *not* located. In the first two Resulting Indicator columns (54 - 57), enter the same indicator. The third Resulting Indicator columns (58 - 59) must be blank. When a record is not found in the update file, the specified indicator turns on.

A record may be updated at detail time or total time. That is, updating can be done after each transaction record is read or after a number of transaction records are read. Thus, columns 9 - 17 are used to specify conditioning indicators.

When an update file is *chained*, another entry is required on the File Description sheet.

The C in column 16 identifies a chained file. (An R in column 28 is needed since the file will be processed randomly by key.)

Job 6 (Figure 10) illustrates how the CHAIN operation code is used to update a master inventory indexed file, and at the same time print a back order report. The back order report lists every transaction item (by customer) where quantity shipped is less than quantity ordered.

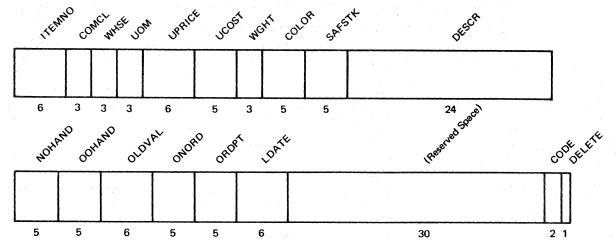
IBM	International Business Mad	hines Corporation		Form X21-9092
RPG CONTE	OL CARD AND FILE	DESCRIPTION SPECIFIC	ATIONS	Printed in U.S.A.
Date			1 2 Program	75 76 77 78 79 80
Program	Punching Graphic Instruction Punch	+++++	Page Identific	
				!
Programmer	Control Card Sp	ecifications		
Size to Compile Comp	7 70 30 30 31 32 32 34 35 36 37 39 30	Refer to the specific System Reference Libra		s ee er eo eo 70 71 77 73 70
0 1 H	7 20 20 30 31 32 33 34 35 35 35		337 33 33 33 33 33 34 34 32 33 34 34 33	
<u></u>	File Description S	Specifications		
File Type File Designation	Mode of Processing Length of Key Field or of Record Address Field		Extent Exit	File Addition/Unordered Number of Tracks for Cylinder Overflow
End of File Sequence	Record Address Type Type of File	Symbolic	for DAM Name of	Number of Extents Tape
File Format	Organization	Device Device	Label Exit	Rewind
Ype Ype	Overflow Indicator	.	Core Index	Condition U1-U8
Block Record Length Length	Key Field Starting Location		Labora	D/A N/U
3 4 5 6 7 8 9 10 11 12 13 14 18 11 11 11 11 11 11 11 11 11 11 11 11	7 28 29 30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49 50 51 52 5	3 54 55 56 57 58 59 60 61 62 63 64 65	66 67 68 69 70 71 72 73 74
0 2 F C	R			
0 4 F	++++++++++++++++++++++++++++++++++++		++++++++++++++++++++++++++++++++++++	+++++++
0 5 F			++++++++++++++++++++++++++++++++++++	
0 6 F	++++++			
0 7 F	╂╂╎╂ ╂┼┼┼┼┼┼		 	
	 			
F				

Updating Records Using the IBM 5471 Printer-Keyboard

The IBM 5471 Printer-Keyboard is an input/output device. It prints data from storage, accepts data into storage, or both prints and accepts data. The device can be used to print a field. For example, in Job 6, we could have printed all item numbers for which no disk records were found.

When fields are printed by the Printer-Keyboard, we say they are *displayed*. To use RPG II to display fields, the fields must be called on a *display file* and described on the File Description sheet. Two new entries are needed on the sheet.

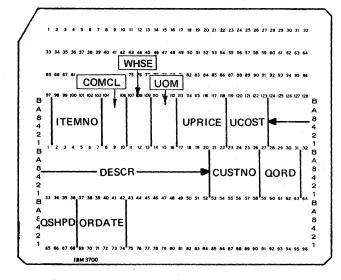
IBM	International Business Machines Corporation	Form X21-9092 Printed in U.S.A.
" RPG CONT	ROL CARD AND FILE DESCRIPTION SPECIFICATIONS	
Date	1 2	75 76 77 78 79 80 Program
	Punching Graphic Page Punching Punch	Identification
Program	Tulial	
Programmer	Control Card Specifications	
Core	Refer to the specific System Reference Library manual for actual entries. 7 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61	62 63 64 65 66 67 68 69 70 71 72 73 74
0 1 H		
File Type	File Description Specifications	File Addition/Unordered
	Length of Key Field or	Number of Tracks
File Designation	for	tent Exit for Cylinder Overflow
End of File	Record Address Type	Number of Extents Tape
Line Filename Sequence	Organization Device	Rewind
File Format	or Additional Area	File Condition
Block Record Length Length		e Index U1-U8
VOIN UND VO	Key Field Starting X S	N/N N/O
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26		62 63 64 65 66 67 68 69 70 71 72 73 74
0 2 F		
0 3 F		
0 4 F		
0 5 F D F	CONSOLE	
0 6 F		
0 7 F		
F		
il		



Although 95 positions are presently used, the disk record is defined as 128 positions to provide space for adding data later. Field names are the same as the input card field names, with two exceptions: CODE = MI (Master Inventory) and DELETE

(position reserved for a delete code).

ITEMNO = Item number COMCL = Commodity class WHSE Warehouse location UOM Unit of measure UPRICE = Unit price = Unit cost UCOST DESCR = Description CUSTNO = Customer number QORD = Quantity ordered **OSHPD** = Quantity shipped ORDATE = Order date



BACK ORDER LISTING

CUSTOMER	ITEM	SHIPPED	ORDERED	BACK ORD.	DATE
292165	A22446	100	300	200	3/17/69
430879	A40911	400	500	100	3/18/69
144367	B70128	60	125	65	3/17/69
590552	C80111	350	150	100	3/19/69

	/	1	2	3	3	4	5	1	5	7	8	9	1	0	1	2		3	\$	5	6	7	8	9	1)	1	2	13	3	4	5	5	6	7	1	3	9	1)	1	1	2	3	4		5	6		7	8	9	1)	1	2	3	4	4	5	6	7	1	В	9	0)	1	2	3	1	1	5	6	1	7	8	9	1	0		1	1	
H	1	\mathcal{C}	U	19)	Ī	C		1	E	R		I			1	7			V				T	T			S	1	1	1	F	7	P	E	1)		Ī	1		1)	R	Z	1	Ē	K	2/		$\overline{\mathcal{D}}$		E	3/	4	\mathcal{C}	K	1		0	R	Z).				I				T	1	D	1			E	I			I		_	
	2			I	I			I																I	Ι				I			Ι	I						Ι	1		Ī			Ι	I			T			Γ	Γ	I				I				I	I				T																	
\mathcal{D}	3	X	X			X	X			_			I	I		X	\rangle			X	X	X			Ι						X			X	X				I			I		X			X	X		X			I	I		X	X			X	X		I				I						/	X		X	1	1		X			_	
	4 5		(C	U	S	TI	N] 0 1)						(-	LE TE	I N	11	10))] 2:	SI	HI L	Ţ PI ↓	D)							I		(Q)F	31	T (c) ⊥)					-(В	C	K	0	R	D	1	1				1		_		1	וכ	R	D	A	T	E	 - -			1		-	

Figure 10. Job 6: Updating a Master Index File Using CHAIN (Part 1 of 3)

IBM [®]	International Business Machines Corporation Form X21-90 Printed in U.S Printed in U.S
Date	RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS
Date	Punching Instruction Punch Punch Punch Punch
Programmer	
	Control Card Specifications
Steel	Second S
	File Description Specifications
End	Mode of Processing Length of Key Field or of Record Address Field Record Address Type Type of File Orsanization File Format Block Length of Key Field or of Record Address Type Type of File Orsanization Overflow Indicator Name of Label Exit Core Index Record Starting Location Addition/Unorderec Number of Tracks for Cylinder Overflow Number of Tracks for Cylinder Overflow Name of Label Exit File Addition/Unorderec Number of Tracks for Cylinder Overflow Number of Extent Tape Rewind File Condition File Condition Un-UB Starting Location Add 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 68 69 7
0 2 F //NVM5TR UC 0 3 F CAROIN 0 4 F BOL / 5T 0 6 Chained File Chained File	F 256 128R 6AI
IBM	International Business Machines Corporation Form X21-9 Printed in U.
·	RPG INPUT SPECIFICATIONS
Program Programmer	Punching Instruction Graphic Page
	Record Identification Codes Field Location Field
Line Filename (N) 1 baguny (O) 1 baguny (O) 2 baguny (O)	To To To To To To To To
0 2 1	Ø1
0 3 1 CARDIN BB	64 6800SHPD
0 5 1	
0 6 1 0 7 1	53 580CUSTINO 59 6300000
08 1	59 6300000 2 70 (TEMNOLI)
0 9 1	
10 1	ITEMNO is the key field in the transaction cards.
1 1 I I I I I I I I I I I I I I I I I I	L1 is assigned to ITEMNO so all transactions for
1 3 I	an item can be accumulated.

Figure 10. Job 6: Updating a Master Index File Using CHAIN (Part 2 of 3)

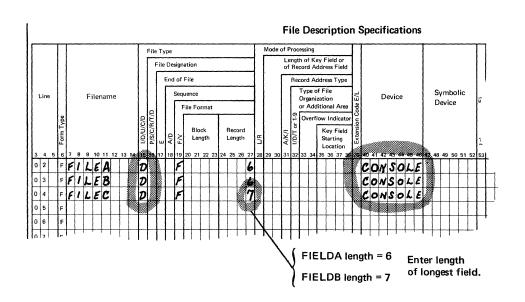
									1	RPG	CA	LCUI	LAT	TION	SPE	CIFI	CAT	IOI	IS			,	1 2				75	76 7	7 78	79 80
Date											nching	Graph		П		T						Page	Ø3		Progr ident	am ificatio		\Box	Ï	
Program										In	struction	Punci	h																	
Programmer	r																				Re	sulting								
Line i	Control Level (L0-L9, LR, SR)	Aı	ndicato	And			Fact	tor 1		Opera	ition		Fact	tor 2		Resi	ult Field		Field Length	Decimal Positions Half Adjust (H)	Arit Plus M Co High I 1 > 2 1 Lc Table (thmetic linus i mpare	qual = 2 2) is			Co	mmen	its		
3 4 5 0	6 7 8 C	9 10 11	12 13 1	15 16	1			23 24 2		40.0		33 34 35 76 U			40 41 42	43 44 R C	45 46 47	748			50	56 57 5	8 59 60	61 62	63 64	65 66	67 68	69 70	71 72	73 74
11+++	c	01	$\forall \dagger$	Н,	6	<i>lOK</i>				SUE		05)H	S	D		70		H	6	0	שכ		+	$\parallel \parallel$	\vdash	H	H	$\dag \uparrow$	$\dagger \dagger$	Н
	c L1			П	1	TE	MN	0		CHA	IN	INY	MS	TR							99	99				\prod	П	П	\prod	Ш
0 4 0	c L 1 1	V99	\mathbb{H}	H	1	IO.	AN	V		5 U.C.	ກກ	101	ЭH			NU TO	HAN TGL	K					+		\vdash	+	H	\vdash	+	Н
0 -		++	\check{H}	++	++	++	++-		++-	2 4	PP	ntity		 	 	// IV	i Pla	201	++ ion 0	 ord	The	ron	+ 	or is	cto	rad	H- in	+		Ш
lo Lir	ne 01		a 5-p	ntity ositi	snı on	pped field	ıs s I call	led B	ctea CKO	RD.	iqua Ifqu	ntity antity	oru y sh	ippe	ed is l	ess ti	han q	uar	ntity	orde	red,	indi	cato	r 50	tur	ns c	n.	Ţ	Ш	Ш
مند ا	Ω2							ACCOUNT.								200000000	600.ma										.	+	+	Н
Lir	ne 0 3										es), t	he tra	ınsa	ctio	n carc	l's ke	ey fie	ld d	hain	s to 1	the u	ıpda	te fil	le. I	f a	like			\parallel	
			key i	s fo	und	, inc	licat	or 99	stay	s off.	Ind	licato	r 99) tur	ns on	only	if no	o k	ey is	foun	d in	the	upda	ate f	ile.			1	4	Ш
Lir	ne 04		Whe	1 T	EW	NO (chan	g es a	nđ a	like k	ey it	foun	e (I	L 1 is	on a	nd 9	9 is e	44)	the	on h	and	qua	ntity	15 G	pda	ted.		+	++	Н
Hi Lin	ne 05		Afte	r the	on	han	ıd qu	uantit	ty is u	updat	ed, T	готѕ	HP	is se	t to z	ero,	so ac	cun	nulat	ions	for	the i	next	item	ı wi	ll be)	1	\parallel	Ш
												e sam															ver,	1	1	Ш
Ш.			101	SHE		es m	Ot al	pear						1 1				1 1		1 1 1		-1-1	1			++	11	++	11	Ш
Date Program Programmer	r								RPG	Pu	inching struction	Grapi Punci	hic							•		Page	Ø 4		Prog Iden	ram tificati	Г	76 7	7 78	79 80
)w (F)	Space	SI	cip .	Out	put Inc	dicators	T											Edit C	odes							
				Overfi	\vdash	├	-		Τ		\dashv			M		ান	ı	Co	mmas	Zero Bal to Pri	ances	No Sig		Τ-	X =	Reme		ı	Ster	ling
Line	_ F	ilenam	е	D/T/E) ct/Fetch					And	And	_] 「	ield Na	me	s er (B)	End Posito	o 3 = Binar		1	Yes Yes	Yes No		1 2	A B	J K	Y =	Plus : Date Field Zero	1	ı	Sigr	
	E A			/pe (H/D/	Before After	Before	After	Not	Not	Not				it Code	in Outpu Recor				No No	Yes No		3 4	C D	М		Supp	ress			1
3 4 5	6 7 8 9	10 11	12 13 14	Star 1	17 18	19 20	21 22	23 24 2	5 26 27	28 29 30	31 32	33 34 35	36 37	38 39	40 41 42	-	45 46 47	7 48 4	19 50 51				Edit W 8 59 60		63 64	65 66	67 68	69 70	71 72	73 74
0 1 0	B01	.15	T ,	H	1		Ш	17		\prod	П			\Box		П	\coprod	П	\prod		П	\prod	\prod	П	П	П	П	П	\prod	Ш
0 2 0		+		K	H	\vdash	\mathbb{H}	O F	H	$+\!\!+\!\!\!-$	+H	++	\vdash	H	\mathbb{H}	8	CU	16	TON	E D	,	+	+	-	+	\dashv	\vdash	+	\vdash	Н
0 4 0		H	+	$\dagger \dagger$	$\dag \uparrow$	$\dag \dag$	\Box	H	H	+	Ш	+		ff		5	17			LN	$\forall t$	#	$\dagger \dagger$	\parallel	\vdash	\parallel	$\dagger \dagger$	$\dagger \dagger$	ff	Ш
0 5		Ш	\prod	П		П		Ш	Ш		Ш				2	8	'5H		PPE			\prod	\prod			П	П	П		Ш
0 6 0		111	$+\!\!+$	${\mathbb H}$	\Vdash	#	Ш	Ш	\mathbb{H}	+	\mathbb{H}		-	\mathbb{H}		8	OR	D	ERE	\mathcal{D}'		$+\!\!+\!\!\!+$	#	-	\vdash	#	-	\vdash	\vdash	Ш
0 8		+	+	+	${\mathbb H}$	+	H	++	Н	$+\!\!+$	\mathbb{H}	+	+	H	4	8	'DA	4	K , 0	Kυ.	+	+	++	-	+	+	+	+	\vdash	Н
0 9				D	1			50	8						П							\perp	止							Ш
1 0 0		\coprod	4	11	$\!$	11		Ш	$\perp \perp \mid$	\coprod	C	UST.	NO		Ш.	6 7	Ш	\sqcup	4		$\perp \downarrow$	11	1			\coprod	11	\coprod	1	Ш
1 1 0		+	+	H	H	++	Н	\mathbb{H}	++	+	6	TEM SHP	NU D	1	1	7	\mathbb{H}	+	$+\!\!+$	H	++	+	$+\!\!+$	₩	-	+	\vdash	\vdash	╁┼	H
1 3		+	$\dagger \dagger$	$\dagger \dagger$	H	$\dagger \dagger$	H	H	$\dagger \dagger \dagger$		ŏ	ORD	<i>\\</i>	$\dagger \dagger$	3	7	\mathbb{H}^{+}	$\dagger\dagger$	+		+	+	+	\vdash	$\dag \uparrow$	$\dagger \dagger$	$\dag \uparrow$	$\dagger \dagger$	$\dag \uparrow$	Н
	0	Ш		П		\prod					\mathcal{B}	CKO	RD		4	6	Ш	П	\perp		\perp	\Box	\perp			\prod	П	П		Ш
1 5		IME	TD	#	H	H	H		III.	4	10	RDA	TE	#	116	Ø	1	++	$+\!\!+$	++	4	#	+	4	4	\dashv	#	#	\dashv	Ш
	o / N \	(IVI)	IK	#	+	+	H		LN9	7	N	ОНА	ND	+	16	8	++	+	++	H	+	+	++	+	$\dag \uparrow$	++	+	+	+	HH
	9 - 15	- 									rinte	d. Ti	his I	,,			ose it	tem	s wh	ere t	he q	uan	tity s	hipp	bed	is le	ss th	nan		団
	c 17			•		-		-				order			-1 L-				داما		- اسم	ا- مه				ادم		4i		Ш
	6 - 17		Un	nan	u qı	uant	ity is	s writ	iten (on the	e aisi	k in it	s or	rgina	ai loc	atıor	ı. Ih	e ti	ieid v	vas u	pda	red a	ıs a r	esui	τ οτ	cal	cula	ผดท	s.	111

Figure 10. Job 6: Updating a Master Index File Using CHAIN (Part 3 of 3)

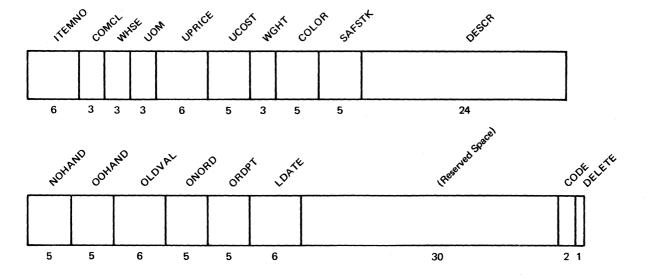
D identifies the file as a display file; CONSOLE is the required device name for the Printer-Keyboard.

Since a file is defined, F is required in column 19, and record length is required in columns 24 - 27.

When fields are displayed, the DSPLY operation code must be used. Fields can be displayed three ways using DSPLY. These three methods and their corresponding File Description entries are: Job 7 (Figure 11) shows how a display file can be used during an update job to print the item numbers for which no master record is found. Note that this job is the same as Job 6 (Figure 10) with the addition of the Printer-Keyboard.

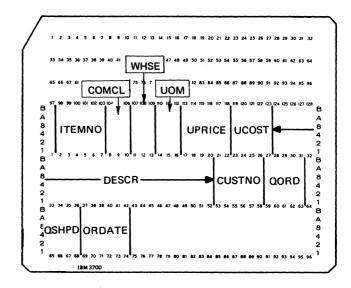


IBM	[_		_										In	tern	ati	onal	Bus	ine	ss M	achi	nes	Cor	ore	tio																												orm			093 .S.A.	
	,																					R	PG	ì	(CA	۱L	С	U	L	A7	П	01	N	SF	E	C	IF	IC	Α;	١T	10	ŊΩ	ıs									. :	2								76							80	
Date																							Γ	Pur	nchi	ing		G	ira	ohic		Γ	Т		T		_	T		Τ		Γ	٦								Page	_	T	٦				ogra				Ĺ	Ť	٦	Ë	Ť	٦			٦
Program.																		_					L	Ins	truc	tio	n	P	un	ch			1		T			1		T		T	1									L					Ide	enti	tica	itio	n	L		ᆚ		1	_1		L	J
Program	mer							_			_																																																											
l —	Т	Т	7			_		-		_		-	Г	_		_						Γ				_	Г		_		_	_	_	_	_	_	Т						Т	-	_	Т	Т	Т			ıltin		7	_	_	_				_	_		_	_	_	_	_	_		7
			Į			lr	ndi	cat	ors	5																											l									l		-	_		met		4																	1
	l		æ																								ŀ																١.	iel		1		1	lus	Mi	nus	Ze	ro																	1
Line	1	ol Level	LR, S			An	d		Ar	nd						Fa	ctc	or 1				(Эре	rat	ioi	n				F	ac	tor	2					Re	su l	t F	ie	ld			iu igtl	mal Position	1	H	igh	L	pare ow < 2	Equ	ual						(Co	mn	ne	nt	š						
	ľ	Control	6.9	-	Т	+	-	7	\dashv	÷.	_	Т																																		li s	Holf Ag	ľ			kur		2																	1
	13	18	리	ž	1	1	Not		I	Not			1																																	1	, ,	I	able	(F	acto	r 2)																		1
3 4 5	5 6	7	8	9	10	11	12	13	14	15	16									×	×				×		8							40	4	F	ie	ılc	A	1	is	: 10	ı ri	nt	ec	ı L	ı	11	ligh <u>55</u>		ow 57			60	61	62	63	64	65	66	67	7 6	8 6	9 :	70	71	72	73	3 74	1
0 1	(F	1	E	L	D.	4				0	S	P	L	Y	F	1	1	E	A	L		L	1	F	ie	lc	1	١	is	'n	01	c	h	an			_	L									L		L	L	1	1		L	L		L	
0 2	10		Ц		_	1					*																					P	r	1		F	rc	g	aı	m	d	lo	es	no	ot	ha	alt	i.,	, .	L									L		L	1	1	1		L	L	L	L	1
0 3	0	: []											L					\perp		L	۰	L							L	١.	4														- 1	Εi	ele	ďι	Δi	S	pri	int	te	d.	•				'	,		L	1	1		L			L	1
0 4	4		Ц		1	1					L		L	Ц	Ц	$ \bot $	1	\perp	1		Œ	D	S	P	L	Y	F	1	1	Ł	E		T	Γ	ľ	Ι	F	1	E	1	_ 2	D		•							alt						_					L	1	1		L	L	L	L	1
0 5	1		Ц		1	1			1		L		L	Ш	Ц		1	\perp	1	L		P		***							₩												•	7	_	Da +	ata +	3 C	an +	t	e e	en ⊢⊣	te	re	d	ın	 	16	IC	<i> </i>	٩.	1	1	1		L	L	L	ļ	1
0 6	ľ	: [Ц		1	1			\perp		L		L		Ц		1	\perp	1	L	L	L						L	L	L	L	L	1	L	L		L	L	L	L	1	1	1	1	1		L		L	L							Ц		L	L	L	1	1	1		L	L	L	L	1
0 7	ŀ				1	1					L	L	L				1	\perp			L	L					L		L	L	L	L	L	L	L	L	L	L	L	L	1	1	1	1	1	1		1		١.	1		١		. !					ł	1	l	1	1		L	L	L	L	1
0 8	ŀ		Ц		1	1					L						#													b							b								1						oe oe					aı	กก	٥					1	\perp		L	L	L	L	1
0 9	C	Ш	Ц		1	1							F	1	E	L	D	4		L		D	S	P	L	Y	F	1	2	E	C		L	L			F	1	E	1	. 2	D I	3		░						p						9	٠.					1	1		L	L	L	L	1
1 0	19	Ш	Ц		1	1					*					◩								▓					L	L	L	L		L		L				Ľ			ı		1						hal				_			.			_		1	1		L	L	L	L	1
1 1	d	Ш	Ц		1	1			1		L	Ц	L	Ц	Ц		1	\perp	\perp	L	L	L	Ц					L	Ľ	ľ	Γ	ľ	ľ	ľ	ľ	Γ		Γ	Ĺ	ľ	Ι	I	1	1	1	L +	λa. +	τa +	ca ⊢	n	be	e	nt	er	ec	ונ	n ⊢⊣	-	e	a	H	• +-	1	1		L	L	L	1	1
1 2	1		Ц	1	1	1					L	Ц	L	Ц	Ц		1	\perp	1	L		L	Ц					L	L		L	1	1	L	L		L	L	L	L	1		1	1	1	1	1	1	L	L				Ц					L	L	L	1	1	1		L	L	L	1	1
1 3	1	Ш	Ц	_														s th																			е	_	L	L	1	1	1	1	1	1	1	L	L	L	Ц	Ц			Ц		L		L	L	L	1	1	1		L	L	L	L	1
1 4	1	Ш	Ц															Γhi sh			ne	3 S	an	ne	n	aı	ne	9 1	m	at	15	a	SS	ıgı	ne	d		_	L	L	1	1	1	1	1	1	L	1	L	L					Ц		L		_	L	L	1	1	1		L	L	L	L	1
1 5	1		Ц												Ц				1		1_		لــــا			L		L	1	L	1	1	1	1	1	L	Ł	1_		L	l		L		1	\perp	1		L												L	1	1	1						١



Although 95 positions are presently used, the disk record is defined as 128 positions to provide space for adding data later. Field names are the same as the input card field names, with two exceptions: CODE = MI (Master Inventory) and DELETE (position reserved for a delete code).

ITEMNO = Item number COMCL = Commodity class WHSE = Warehouse location UOM = Unit of measure UPRICE = Unit price **UCOST** = Unit cost = Description DESCR CUSTNO = Customer number QORD = Quantity ordered **QSHPD** = Quantity shipped ORDATE = Order date



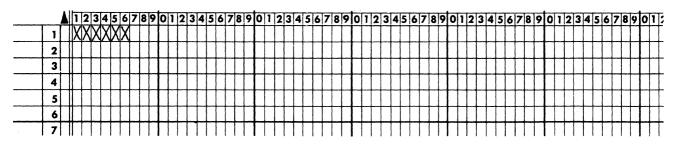


Figure 11. Job 7: Updating an Indexed File and Using the Printer-Keyboard (Part 1 of 3)

IBM,	International Business Machines Corporation	Form X21-9092 Printed in U.S.A.
RPG CC	ONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS Punching Graphic Program Page Program Program	75 76 77 78 79 80
Program	Instruction Punch Identifica	ition []]]]
Programmer	Control Card Specifications	
Core Core Size to Core Size to Compile Core Size to Size t	int Refer to the specific System Reference Library manual for actual entries.	66 67 68 69 70 71 72 73 74
	File Description Specifications	
Since the Printer-Keyboard is being u CONSOLE is the device name for the	Mode of Processing Length of Key Field or of Record Address Field Record Address Field Type of File Overflow Indicator Key Field or or Additional Area Overflow Indicator Key Field or or Address Field Name of Label Exit Core Index Core Index Core Index CONSOLE Used to display a field, the field must be defined as a display file. Printer-Keyboard.	File Addition/Unordered Number of Tracks for Cylinder Overflow Number of Extents Tape Rewind File Condition U1-U8 2 4 66 67 68 69 70 71 72 73 74
IBM	International Business Machines Corporation	Form X21-9094 Printed in U.S.A.
*	RPG INPUT SPECIFICATIONS	75 76 77 78 79 80
Program	Punching Instruction Punch Page 22 Program Identificat	ion _
Programmer		
	2 3 Aliana Field Name Fie	Field Indicators Zero or Blank 66 67 68 69 70 71 72 73 74
0 1 I / NVM5 TR AA Ø3		
O 3 I CARDIN BB Ø1	64 68@MOHAND	
0 4 1 0 5 1	64 68 ØQSHPD 69 74 Ø ØRDATE 53 58 ØCUSTNO 59 63 ØQORD 2 7 Ø I TEMNOLI	
0 6 1	69 7400RDATE 53 580CUSTNO 59 630QORD 2 701TEMNOL1	
0 8 1	2 TOTTEMNOLI	
1 0 1		
1 1 1		
	╒╏┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋┋	╫╫╫╫╫╢

Figure 11. Job 7: Updating an Indexed File and Using the Printer-Keyboard (Part 2 of 3)

											International Bus											Form X21-90 Printed in U.	S.A.
								1	RPG	C/	ALCULAT	FION:	SPECIF	ICATIO	NS 1		1	2			75 76	77 78 79	80
Date										unching	Graphic Punch	-					Page 0	3	Progra Identif		Ш	Ш	Ш
Program_									L		, , , , ,	L			j								
Programn	T T	T		_							T .				ПТ	Re	esulting	Τ					$\neg $
		Indica	tors													-	dicators thmetic	-					
	SR)		T	7											Field su	~ h-	Minus Zero ompare						
Line	8 3 E	1 7	And			Facto	or 1		Oper	ation	Fac	tor 2	Re	sult Field	Field Piering Period	₩ High	Low Equal 1< 2 1 = 2			Comn	nents		
	Form Ty Control (L0-L9,	Not	Not	П											Decim	T L	ookup						
		1111.													52.5	High	(Factor 2) is Low Equal						
0 1	6 7 8 C	9 10 11 12 13	14 15 16		9 20 5H		23 24 25	5 26 27	C 01		QORD	38 39 40	41 42 43 4	4 45 46 47 48	49 50 51 52		56 57 58 59 50	60 61	62 63 64	65 66 67	7 68 69 7	0 71 72 73	74
0 2	С	01		Õ	5 <i>H</i>	PD			AD	2	TOTSH	P	TO	TSHP	60								\square
0 3	c L1				TE	MN(0		CH_{ℓ}	4/1	INVMS	TR				99	99			44		111	$\perp \parallel \parallel$
0 4	c L 1	N99	+++	N	UH.	AN)	И		<u> 24</u> (101	идирь		NC	HAND T5HP	H + H				HH	++	+++	+++	$+ \parallel$
0 6	c 1 1	99		7	TF	MN	0		057	01 V	CON5L	0117	1/0	IJAP	H + H	1++	+++			+	+++	+++	+11
0 7	c			1		7				- 1	UUIUU	-0 G		TTTT			111			11		111	\Box
0 8	c I	ndicator 9	9 turi	is or	wh	en a	disl	k reco	ord i	s not	located (Line (03). Th	e DSPL	Y line (0	6) is c	onditio	oned	by _				
0 9	++ -	_1 and 99 displayed.	Sov This	hen ise d	no of D	reco ISPI	ord is LY d	four	nd ar	id all alt ti	transacti ne prograf	ons fo	r that I hange I	TEMNO	are pro	cessed VO is t	l, ITEM orinted	NO onl	is v.		144	+++	
1 0	c c	1			+	+	11	111		11	1		† † † †	1111		111	111			++	+++	+++	+1
1 2	c	\Box	HH	Н	+	\forall	+	+	+	++		+++	1111			+	$\pm \pm \pm$			\forall	$\dagger \dagger \dagger$		$\Box I$
1 3	С																						
IBM						ACCES CONTRACTOR	TOTAL PROPERTY.		CHECK COMP		International B	Susiness Mac	hines Corporat	ion	The second secon			f-to-4006		en viscos é		Form X21-9	
adıv.								RPO	G	O	JTPUT - F	FORM	AT SP	ECIFICA	ATIONS							Printed in U	
Date									_		Ta	Т	ТТ		7		Page Ø		Prog	ram	75 76	77 78 7	9 80
Program										Punchin Instruct	9		+		-		Page W	4	lden	tification			Ш
Programn	ner										a ti saveta Advissa deli saveta			and the contract of the contra									
Γ	TT		<u>E</u>			П	_			Т		T										T	\neg
			1 13	Snacel				tput In															- 1
	1 1		l left	Space	Sk	IP	Ou	tpat m	uicato	rs		6			Zerr	o Balances	Edit Cod						- 1
1		Eilonomo	tch Overflo	Т	- 5K	"P		T	T	rs	Field Neves	F	End	Nary L	tommas 1	o Balances to Print	No Sign	CR		Remov Plus Sig		Sterli Sign	
Line	ed.	Filename	T/E) /Fetch Overfi			IP		And	And	rs	Field Name	les (B)	End Positon	(B = Binary)	Yes Yes		T 1	CR A B C	J Y =	Plus Sig Date Field E Zero	gn idit		
Line	эгт Туре	Filename	ype (H/D/T/E) ker Select/Fetch Overflo	Before After	Before			T	T	rs	Field Name	ank After (B)		Packed/B = Binary	Yes	Yes No Yes No Yes	No Sign 1 2 3 4	CR A B C D	J K Ł Z ≈	Plus Sig Date Field E	gn idit	Sign	
Line 3 4 5	9 Form Type 8	Filename	Type (H/D/T/E) Stacker Select/Fetch Overfl	Before After	Before	After	Not	Not	And		Field Name	Edit Codes Blank After (Positon in Output Record	P = Packed/B = Binary	Yes Yes Yes No No	Yes No Yes No Yes No	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5	o B C	9 10 11 12 13	Type (H/D/T/E) Stacker Select/Fetch Overfl	Before After	Before	After	23 24 2	And 52 25 26 27	And			Edit Codes Blank After (Positon in Output Record	P = Packed/B = Binary	Yes Yes No	Yes No Yes No Yes No	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2	0 B C	9 10 11 12 13	5 Type (H/D/T/E) 5 Stacker Select/Fetch Overfl	Before After	Before	After	23 24 2	And 52 25 26 27	And			Edit Codes Blank After (Position in Output Record	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3	0 B C	9 10 11 12 13	T Type (H/D/T/E)	Before After	Before	After	23 24 2	And 52 25 26 27	And			Edit Codes Blank After (Position in Output Record	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3	0 B C	9 10 11 12 13	T Type (H/D/T/E)	Before After	Before	After	23 24 2	And 52 25 26 27	And			Edit Codes Blank After (Position in Output Record	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5	0 BC 0 0	9 10 11 12 13	T Type (H/D/T/E)	Before After	Before	After	23 24 2	And 52 25 26 27	And			Edit Codes Blank After (Position in Output Record	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5	0 B C	9 10 11 12 13	A Type (H/D/T/E)	2 Before	Before	After	23 24 2 11	And 55 26 27 D	And			Edit Codes Blank After (Position in Output Record	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7	0 B 0	9 10 11 12 13	A Type (H/D/T/E)	Before After	Before	After	23 24 2	And 55 26 27 D	And	330 31 3	2 33 34 35 36 3	2 Edit Codes 8 Edit Codes 8 Blank After (Positon in Output Record 0 41 42 43 -	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8	0 80 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	9 10 11 12 13	A Type (H/D/T/E)	2 Before	Before	After	23 24 2 11	And 55 26 27 D	And	31 3	2 33 34 35 36 3	88 Edit Codes 68 Blank After 1	Positon in Output Record 0 41 42 43 -	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1	0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	9 10 11 12 13	A Type (H/D/T/E)	2 Before	Before	After	23 24 2 11	And 55 26 27 D	And	31 3	2 33 34 35 36 3	88 Edit Codes 68 Blank After 1	Position in Output Record 0 41 42 43 43 43 43 43 43 43 43 43 43 43 43 43	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1	0 80 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	9 10 11 12 13	A Type (H/D/T/E)	2 Before	Before	After	23 24 2 11	And 55 26 27 D	And	31 3	2 33 34 35 36 3	88 Edit Codes 68 Blank After 1	Position in Output Record 0 41 42 43 43 43 43 43 43 43 43 43 43 43 43 43	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1 1 1 1 2 1 3	0 80	9 10 11 12 13 DL I S T	OK Stacker Select Feeth Overfile	2 Before	Before	After	52 23 24 2 117 01	And	And 150/2 28 29	31 3	2 33 34 35 36 3	88 Edit Codes 68 Blank After 1	Position in Output Record 0 41 42 43 43 43 43 43 43 43 43 43 43 43 43 43	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1 1 1 1 2 1 3 3	0 80 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	9 10 11 12 13	OK Stacker Select Feeth Overfile	2 Before	Before	After	52 23 24 2 117 01	And 55 26 27 D	And 150/2 28 29	330 31 3	2 33 34 35 36 3 CUSTNO TEMNO 25HPD 20RD DRDATE	Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z	Position in Output Record 88 155 28 38 48 48 48 5 7 7 5 7 7 5 7 7	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1 1	0 80 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	9 10 11 12 13 DL I S T	OK Stacker Select Feeth Overfile	2 Before	Before	After	52 23 24 2 117 01	And	And 150/2 28 29	330 31 3	2 33 34 35 36 3	Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z	Position in Output Record 0 41 42 43 43 43 43 43 43 43 43 43 43 43 43 43	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1 1 1 1 2 1 3	0 80 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	9 10 11 12 13 DL I S T	OK Stacker Select Feeth Overfile	2 Before	Before	After	52 23 24 2 117 01	And	And 150/2 28 29	330 31 3	2 33 34 35 36 3 CUSTNO TEMNO 25HPD 20RD DRDATE	Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z	Position in Output Record 88 155 28 38 48 48 48 5 7 7 5 7 7 5 7 7	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1 1 1 1 2 1 3	0 80 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	9 10 11 12 13 DL I S T	OK Stacker Select Feeth Overfile	2 Before	Before	After	52 23 24 2 117 01	And	And 150/2 28 29	330 31 3	2 33 34 35 36 3 CUSTNO TEMNO 25HPD 20RD DRDATE	Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z	Position in Output Record 88 155 28 38 48 48 48 5 7 7 5 7 7 5 7 7	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion
3 4 5 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1 1 1 1 2 1 3 3	0 80 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	9 10 11 12 13 DL I S T	OK Stacker Select Feeth Overfile	2 Before	Before	After	52 23 24 2 117 01	And	And 150/2 28 29	330 31 3	2 33 34 35 36 3 CUSTNO TEMNO 25HPD 20RD DRDATE	Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z	Position in Output Record 88 155 28 38 48 48 48 5 7 7 5 7 7 5 7 7	Dacked/B = Biocked/B = Biocked	Yes Yes No No	Yes No Yes No Const	No Sign 1 2 3 4 tant or Ed	CR A B C D	J K L Z ≈	Plus Sig Date Field E Zero Suppre	gn Edit ess	Sign Positi	ion

Figure 11. Job 7: Updating an Indexed File and Using the Printer-Keyboard (Part 3 of 3)

USING EXTERNAL INDICATORS WITH DISK FILES

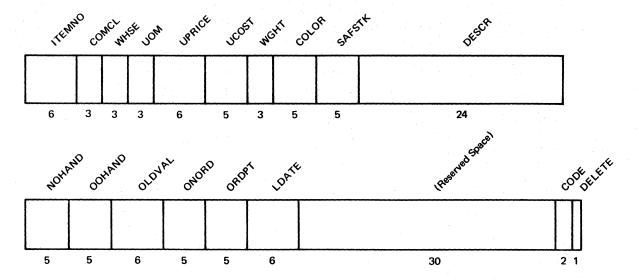
You can condition a file outside an RPG II program by using an *external indicator*. The indicator is called external because it is set on or off prior to running a job. It is set by an OCL card and *cannot* be changed during the job. The program has no control over it. Chapter 4, Operation Control Language (OCL), explains how to set external indicators.

RPG II provides eight external indicators, specified as U1-U8. External indicators are entered in columns 71-72 on the File Description sheet.

External indicators are used to condition when a file is used. For instance, a job may require two output files one time and one file another time. Instead of writing two different programs (one program having one file, the other two), you can condition a file by an external indicator. When the indicator (U1-U8) is on, the file is used; when it is off, the file is not used.

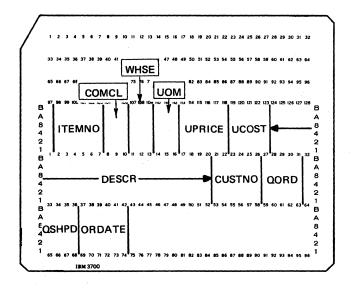
In addition to using these indicators to condition a file, you may use them to condition calculation and output operations. Job 8 (Figure 12) shows the use of an external indicator that conditions a Printer-Keyboard file, conditions calculations, and conditions output.

International Business Machines Corporation Form X21-9092 Printed in U.S.A.														
* RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS														
ate Punching Graphic Page Program														
Program	Instruction													
grammer														
rrugrammer	Control Card Specifications													
Line Core Size to Size to Core														
0 1 H														
File Type File Designation End of File Sequence	File Description Specifications Mode of Processing Length of Key Field or of Record Address Field Record Address Type Type of File Organization or Additional Area 8	Symbolic Name of Device W Label Exit	Extent Exit for DAM Extent Exit File Addition/Unordered Number of Tracks for Cylinder Overflow Number of Extents Tape Rewind											
File Format Compared Compare	Overflow Indicator of Section 1972 Section 1972	5 N S S S S S S S S S S S S S S S S S S	Core Index											
0 2 F														
0 3 F														
0 4 F														
0 5 F														
0 6 F														
0 7 F														
F														



Although 95 positions are presently used, the disk record is defined as 128 positions to provide space for adding data later. Field names are the same as the input card field names, with two exceptions: CODE = MI (Master Inventory) and DELETE (position reserved for a delete code).

ITEMNO = Item number COMCL Commodity class WHSE Warehouse location UOM Unit of measure **UPRICE** = Unit price UCOST = Unit cost DESCR = Description CUSTNO = Customer number QORD Quantity ordered **QSHPD** = Quantity shipped ORDATE = Order date



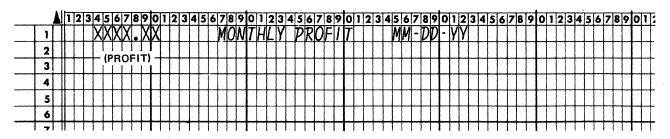


Figure 12. Job 8: Using the Printer-Keyboard as an Output File (Part 1 of 3)

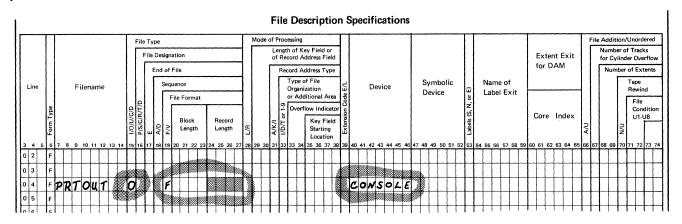
IBM	International Business Machines Corporation Form X21 Printed in	
RPG CONT	ROL CARD AND FILE DESCRIPTION SPECIFICATIONS	79 80
Program	Punching Instruction Punch Page 1 Program Identification	Ш
Programmer		
	Control Card Specifications	
Steeling	Refer to the specific System Reference Library manual for actual entries. 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72	2 73 74
	File Description Specifications	
Since the Printer-Keyboard is	used as an output file, it must be defined as an output file.	red s
Since the summary report is p (through an OCL statement) p	printed monthly. U1 is used to condition when the file is used. For this job U1 is set on	ents
Seq (7)		dition
Block Record Control	Key Field & Core Index U1.U	
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26	27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72	73 74
0 2 F INVMSTR UC F 256 12 0 3 F CARDIN IP F 96 9	8R 6AI 1 1 DISK MFCU1	Ш
104 FBOL15T 0 F	Ø PRINTER	
o 5 FMONSUMRYO F 3		
0 6 F		\mathbb{H}
		儿
F		
IBM	International Business Machines Corporation Form X2 Printed in	
Date	RPG INPUT SPECIFICATIONS 1 2 75 76 77 78 7	79 80
Program	Punching Instruction Graphic Page Program Identification Punch Punch	Ш
Programmer		
Ri gj Ri	ecord Identification Codes Field Location Field	
Bi Indicator	2 3 \gamma\ \gamma\	
Form Type Recurrence Number (1-N) Option (0) Record Identifying out (N) Not (N) CiZID		
Figure 1 Property 1 Pr	Position Compared	ition
0 1 I / NVMSTR AA 03	27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72	73 74
0 2 1	64 68 NOHAND Update field	++-
0 3 1 0 4 1	64 68 NOHAND Update field 22 26 UCO57 Field to be accumulat 16 21 UPRICE Field to be accumulat	-
0 5 1 CARDIM BB . Ø1		rea
0 6 1	64 68@QSHPD	Ш
0 7 1 0 8 1	64 68 0 SHPD 69 74 0 RDATE 53 58 0 C USTNO 59 63 0 0 RD 2 7 1 TEMNO 4 Chaining field	+++
		+++
10 1	2 7 17 EMNO Chaining field	Ш
1 1 I I I I I I I I I I I I I I I I I I	┞┧╎╎┧╏╏╏╎┆╎╏╏╏╏	++1
13 1	╒┋┋╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒	++1
┇ ╒╶┩╒╶╏╒┊┈┆┈┆┈┆┈┆┈╏┈╏┈╏┈╏┈ ╏	┎╒┍┍┍╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒	+

Figure 12. Job 8: Using the Printer-Keyboard as an Output File (Part 2 of 3)

IBM	International Business Machines Corporation	Form X21-9093 Printed in U.S.A.
l [°] R	PG CALCULATION SPECIFICATIONS	
Date	Punching Graphic	1 2 75 76 77 78 79 80
Program	Instruction Punch	Identification
Programmer		
		Resulting
Indicators		Indicators Arithmetic
		Plus Minus Zero
Line Line Line And And Factor 1	Departion Factor 2 Result Field Field Ength	Compare Comments
	Adjal	1>2 1< 2 1 = 2
Not Not Not	land	Lookup Table (Factor 2) is
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28		High Low Equal
	OMP OORD	54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
O 2 C ØI QSHPD A	DD TOTSHP GO	
	DD TOTSHP TOTCST 82	
04 C Ø3 U1 UPRICE A	DD TOTPR TOTPR 82	
O 5 CLI TEMNO C		9999
ole cL1N99 NOHAND 5	UB TOTSHP NOHAND	
07 CL1 # 2	-ADDØ	
OBCLRUI	UB TOTCST PROFIT 62	
0 9 C		
Lines 03, 04, and 08 are conditioned by U1 sine		
are the ones performed when the monthly sumi	nary report is needed.	
1 2 C		
1 3 C		
Oate	Punching Instruction Punch	1 2 75 76 77 78 79 80 Page Program Identification
Programmer (iii) (iii) Space Skip Output Indic	ators Commas Zero Ball to Pri	- NO Sign Ch - A Monte Ctenting
Type Hom Type Ho	Field Name (a) Field Name (a) Foiton (b) Field Name (c) Foiton (c) Field Name (c) Foiton (c) Field Name (c) Fie	1 A J Y = Date Sign Position
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 5	54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
		
0 2 0 0 0 0 R 0 F	A LOUISTONED	╻┊┊┊┊
0 3 0	8 CUSTOMER 15 ITEM	┼┼┼┼┼┼┼┼┼┼┼┼┼┼┼┼┼┼┼ ┼┼┼
0 5 0	28 'SHIPPED	
0 6 0	38 ORDERED	
0 7 0	48 DATE	
0 8 0 D 1 50		
	CUSTNO 6	
1 0 0	ITEMNO 17	
	QSHPD Z 27	
1 1 0	OORD Z 37 Since MONSUM	IRY is an output file conditioned
1 1 0		
1 1 0 1 2 0 1 1 3 0 1 1 1 3 0 1 1 1 1 1 1 1 1 1 1	ORDATEY 50 by U1 on the F	ile Description sheet, MONSUMRY
1 1 0 1 2 0 1 3 0 1 4 0 /WVM5TR T LIW99	DEPT SET DESCRIPTION OF THE PROPERTY OF THE PR	ile Description sheet, MONSUMRY nditioned by U1 on the Output
1 1 0 1 2 0 1 3 0 1 1 3 0 1 1 1 1 1 1 1 1 1 1 1 1	by U1 on the F must also be co sheet. (If an in	ile Description sheet, MONSUMRY nditioned by U1 on the Output put file had been conditioned by
1 1 0 1 2 0 1 3 0 1 4 0 1 MVM5TR T LIN99	by U1 on the F must also be co sheet. (If an in U1 on the File	ile Description sheet, MONSUMRY Inditioned by U1 on the Output put file had been conditioned by Description sheet, U1 would con-
1 1 0 1 2 0 1 3 0 1 4 0 /WVM5TR T LIW99	by U1 on the F must also be co sheet. (If an in U1 on the File dition that file	ile Description sheet, MONSUMRY Inditioned by U1 on the Output put file had been conditioned by Description sheet, U1 would conon the Input sheet.)
1 1 0 1 2 0 1 3 0 1 4 0 / MVM5TR T LIN99 1 5 6 MONSUMRYT LR U1	by U1 on the F must also be co sheet. (If an in U1 on the File dition that file 30 MONTHLY	ile Description sheet, MONSUMRY Inditioned by U1 on the Output put file had been conditioned by Description sheet, U1 would conon the Input sheet.)
1 1 0 1 2 0 1 3 0 1 4 0 1 NVM5TR T LIN99	by U1 on the F must also be co sheet. (If an in U1 on the File dition that file	ile Description sheet, MONSUMRY nditioned by U1 on the Output put file had been conditioned by Description sheet, U1 would con-

Figure 12. Job 8: Using the Printer-Keyboard as an Output File (Part 3 of 3)

Besides being used to display fields, the Printer-Keyboard can be used as a second printer. It is so used when the printer is being used. When the Printer-Keyboard is used as a second printer, the file must be described as an output file.



In Job 8 (Figure 12) the master inventory file is used again. The previous jobs updated the file and printed a back order listing. Depending on the number of transactions, this job may be performed every week or every day.

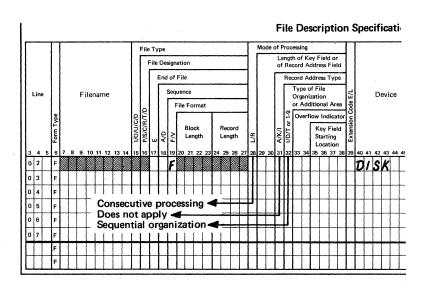
At the end of the month, however, along with updating and printing the back order report, a summary report of that month's profit is needed. Profit is calculated by accumulating unit cost and unit price for each item, and then subtracting the total cost from the total price.

The Printer-Keyboard is used to print profit; thus it is an output file. For this month-end run, U1 must be turned on by an OCL card prior to running the job. For the next run, the summary report is not needed. Thus, U1 must be turned off for that run. External indicators are used, then, to regulate when certain functions are performed.

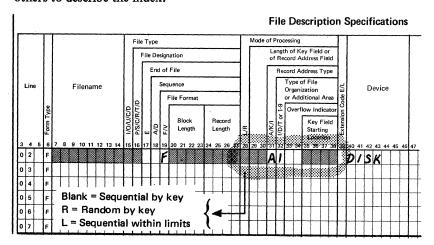
REVIEW: RPG II ENTRIES FOR DISK FILES

When a sequential file is processed consecutively, all records are read one after the other in the order they were created. For any job where a sequential file is used, the same disk entries are required on the File Description sheet.

Columns 28, 31, and 32 identify file organization and processing method. For sequential files processed consecutively, these three columns are always blank.



For indexed files the same entries are required, plus five others to describe the index.



If a job involves additions to a file or updating, the same entries are required for both sequential and indexed files.

File Addition/Unordered File Type Number of Tracks File Designation Extent Exit for Cylinder Overflo for DAM End of File Record Address Type Number of Extents Type of File Organization or Additional Area Symbolic Name of Line Filename Device Device Label Exit File Format File Overflow Indicate Core Index Key Field Location DISK DISK DISK DISK Blank, L, or R depending on method of processing

File Description Specifications

The first part of this chapter explained the RPG II entries used to perform such functions as adding records, updating records, and creating files. These functions and their corresponding entries were explained separately; however, a combination of these functions can be performed in one job. For example, during one job, inventory records are updated and added. As a result of shipping, the items-on-hand quantity (in master record) is decreased by the quantity shipped; this is an update functions. When a new item is added to the inventory, a master record containing information for this item must be added to the master file. This is an add function: a new record is added.

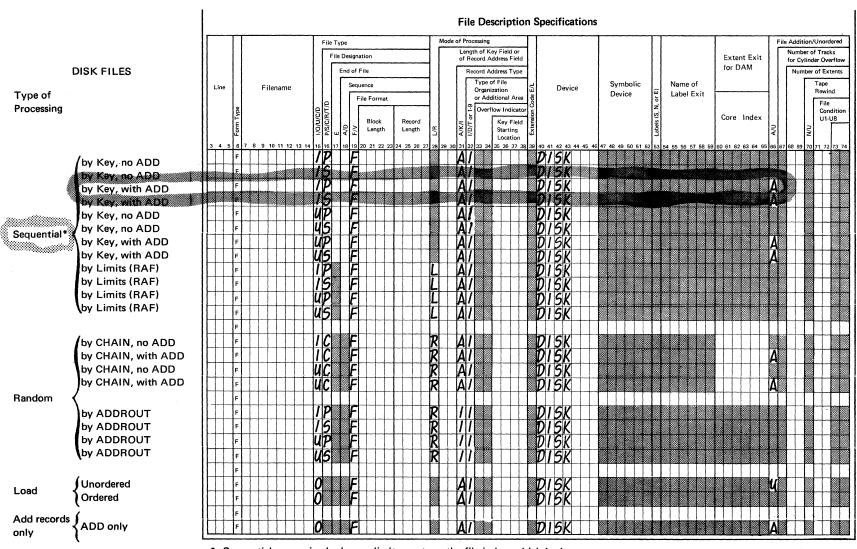
The Disk System RPG II Reference Manual contains several pages of File Description sheets that show various combinations of entries. You can use these sheets to determine the entries required to do a certain function or combination of

functions. Figure 13 shows processing methods for indexed disk files. For example, the third line shows the disk entries needed to add records to an indexed file that is processed sequentially by key.

THREE SAMPLE PROCESSING JOBS

The next three jobs (Figures 14, 15, and 16) show typical processing of disk files. Job 9 (Figure 14) is an example of detail printing from a sequential file. Name and address labels are printed from a name and address file. The labels are used to mail brochures and catalogs to customers.

Job 10 (Figure 15) shows how matching records are used with a sequential file, and Job 11 (Figure 16) shows how matching records are used with an indexed file.



* Sequential processing by key or limits must use the file index, which is always arranged in ascending sequence. When an indexed file is processed record by record from beginning to end, the file is processed through the index using the sequential by key method.

Note: Shaded columns do not need to be coded.
Unshaded columns must or may be coded.

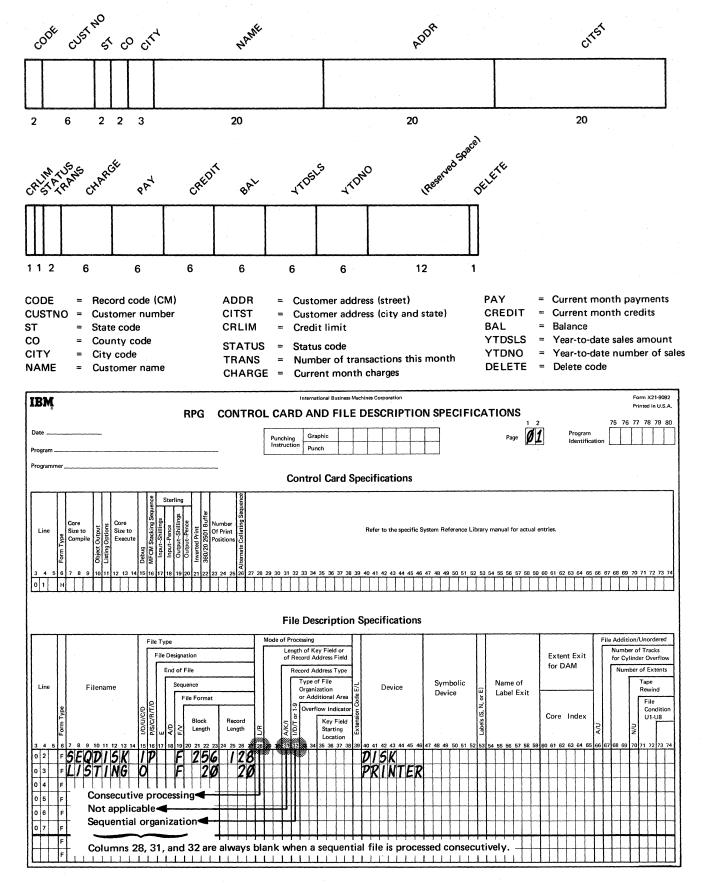
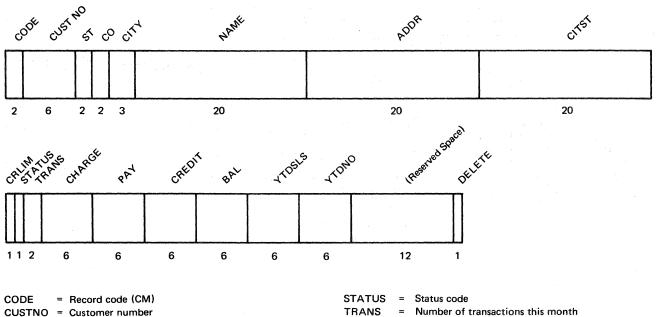


Figure 14. Job 9: Detail Printing from a Sequential File (Part 1 of 2)

IBM	Į																		il Busin																						21-9094 in U.S.A.	
															RF	PG	11	NPL	JT S	PE	CH	FIC	Ά	TIO	NS	;							1 2					75	76 7	7 78	79 80	
Date				-											Punch			iraphic			I										Pa	age (1 2		Prog Iden		ation		П			7
Program														Ĺ	Instru	ction	P	unch														4	Livin					-				_
Program	mer .																																									1
	П					T		Π	ator	L			Rece	ord lo	enti	ficat	ion	Code	s					F	ield	Loca	tion		T								Fi					1
						-			Record Identifying Indicator			1			:	2				3]	nary					_	1				(67		tion	L	In	dicat	ors			
Line			File	nam	e			2	tifying	5			$\ \cdot\ $								П	t	B = Bi						ations	Field	Nan	ne	11)	elds or	d Rela		ĺ			St	erling	
	Type				-		nce.	Number (1-N)	i den	Po	osition	9	te	Pos	ition	9	1	Po	sition	9	C/Z/D	er Sele	/payo	Fro	om		То	1	la Pos				Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	١,	Plus	Minus		Po	sition	
	Form						Sequence	Number (1	Record			Not (C/Z/D Character			Not (1	C/Z/D			Not (C/Z/C	Stack	P = Pa					1	Decimal				Contro	Match	Field				Blank			
3 4	5 6	7 8	9 10	0 11	12 13			17 15	1	1	22 23	- }	26 27	ł	30.3	11	- 1	1	16 37 3	- 1		1	1 1	44 45	46 4	7 48	49 5	51 5	52 5	3 54 5F	56.5	57 5R	59 6	0 61 6	63.6	64 6	5 66	67 68	69.70	71 7	2 73 74	
0 1	I	SE	OI	71	SK	1/	NΑ		Ø			T		Ť	П		1		TT	1	Ť	1	Ħ	Т	Π̈́	ŤŤ	Ť	TT	1		П	T	T	T	T	1		T		T	П	
0 2	ı					T	,	IT		1	\top	1	Ħ	II		\dagger	1	\Box	\top	T	Ħ	T	П	\top	14	6	3	5	1	IAM	E	T	H	Ħ	Ħ	†	\top	\top	ff	Ħ	TT	1
0 3	I								П			I										L			16 36 56	á	E	5	1	IAM DD IIT	R											
0 4	1		Ш	П		Ц		Ш	Ш	Ш			Ш			Ш		Ш				\perp	Ш	\perp	50	ó	7	5	(17	9	T	Ш		П		Ш			Ш	Ш	
0 5	1	-	\sqcup	$\perp \downarrow$	1	Ц	+	\sqcup	11	\sqcup	44	\perp	\Box	Ш	Ш	\sqcup	4	$\perp \downarrow$	11	1	\perp	Ļ	Ц	\perp	Ц	\perp	4	Ш	1	4	Ш	_	Ш	\coprod	11	4	$\bot \downarrow$	+	\sqcup	11	4	4
0 6	I	\vdash	H	H	+	\vdash	+	${\mathbb H}$	+	\mathbb{H}	+	+	-	\vdash	H	\mathbb{H}	\perp	\dashv	44	\perp	\dashv	+	H	+	\dashv	11	+	+	4	\vdash	H	+	\vdash	H	\sqcup	+	\mathbb{H}	+	\vdash	H	+	\parallel
0 7	I	\vdash	1	+	+	H	+	1	#	+	\dashv	+	₩	\vdash	H	+	+	+	44	+	\vdash	+	H	+	\vdash	+	+	++	+	++	H	+	\vdash	$^{+}$	\mathbb{H}	+	\mathbb{H}	+	<u> </u>	₩	44	-
0 9	1	-	╁	\mathbb{H}	+	\vdash	+	H	++	+	+	+	\vdash	\vdash	\vdash	Н	+	+	+	+	+	+	Н	+	\vdash	+	+	+	+	++	H	+	\vdash	₩	H	+	+	+	\vdash	+	++	-
1 0	I	\vdash	\vdash	+	+	+	+	H	++-	+	+	+	+	\vdash	\vdash	\forall	+	+	+	-	+	╁	H	+	+	+	+	+	+	H	H	+	H	+	H	+	+	+	\vdash	+	++	╢
1 1	I	\vdash	\vdash	$\dagger\dagger$	+	H	+	$\dag \uparrow$	$\dagger \dagger$	\forall	+	+	$\dagger \dagger$	\vdash	H	$\dagger \dagger$	+	$\dagger \dagger$	+	+	+	$^{+}$	H	+	$\dagger \dagger$	+	+	$\dagger\dagger$	†	++	H	+	H	++	$\dagger \dagger$	+	+	+	H	$\dagger \dagger$	+ +	1
1 2	1	\vdash	\vdash	Ħ	+	$\dagger \dagger$	+	Ħ	$\dagger \dagger$	$\dagger\dagger$	\top	\dagger	$\dag \dag$	H	H	$\dagger \dagger$	†	$\dagger \dagger$	$\forall t$	\dagger	\dagger	†	H	T	H	$\dagger \dagger$	\dagger	$\dagger\dagger$	\dagger	$\dagger \dagger$	Ħ	\dagger	\vdash	H	$\dagger\dagger$	+	$\dagger \dagger$	\dagger	\vdash	$\dagger \dagger$	$\dagger \dagger$	1
1 3	Ì		П	П	\top	П	T	IT	T	\sqcap		T	\sqcap	\Box	П	\parallel	1	\prod	11	1	T	T	П	T	П	\top	1	П	1	П	П	T	T	П	П	1	$\top I$	T		T	$\dagger \dagger$	11
1-1-1						_	_					_	-	-		-													+				1 !	-	-		-			orm Y	21-9090	7
IBN	ļ													_		٠.,			nal Busi																						n U.S.A.	1
													RP	Ji		UU	IP	UI	- F(JKI	MA	. 1	SP	EC	11-1	CA	110	JNS	>				1 2					75	76	77 78	79 8	30
Date _														- 1		hing uctio	-	Graph						<u> </u>							Pa	ige 🕻	1 2			ogra entif	m icatio	n				
Program	-													l	mstr	uctio		Punch						<u>L</u> .	<u></u>	L						-										-
Program	mer_																																									
							flow (F)	Space	e s	Skip		Out	put In	dicat	ors	T				_					_	_												_				
1						1	Overflo	_			\perp		,			1			1	6					~		mma	Ze	ero E	alances		it Co Sign			×	= F	Remov	/P		1		
Line			File	nam	e	1	atch 0	$\ \ $								1	Field	l Nan	ne 🏻		E	nd		inary		-	Yes	+		Print es	+-	1	A	+	- Y	F	Plus Si Date			Si	erling gn	II
	e d					Ł	lect/F	Before			\vdash	A	ind	And	ГΤ	┨			- 1	Blank After (B)		osito	ا ۱	= Packed/B = Binary		1	Yes No		7	lo 'es		2	B	K	z	- 2	ield E Zero	ļ		"	sition	
	Form Type					1	ker Selec	Before	Before	After	Not		Not	Not						Blank Afte	0	utpu ecord	t i	acked		L	No		- 1	lo	<u>L</u> .	4	D	М			Suppre	ess		1		П
3 4			0.40		10 10	- 16	Stac				200	20	26 27		20/2		22.2		. 27		1			d 45	40 4	7 40	40 E		- n -	Cons	tant	or E	dit V	Vord			E 66	67 60	60.70	J., ,	2 72 7/	$\ $
0 1	0	ĹĬ	57		NG	1		Η̈́	TT	T		Ø1	20 27	20 23	30 3	٦	J. J.		7/13	0 39		T		-	ΪŢ		T	TT	1	1	M	7	II	TT	TT	T	Ĭ	T	Π̈́	П	TT	11
0 2	О			П		П	Τ	П	T	\Box	П		\prod			N	41	1E	$\top \top$	T	П	2	Ø	T	П	П	T	Π	T	П	П		П	\prod	\prod		П		П	П	\prod	71
0 3	0	-+-	П	\prod		1)	2		П		01							\prod	I		L				П	I	П		П	П				П	\perp		I	П	П	\prod]
0 4	0		Ш	\sqcup	_	Ц	\downarrow			Ш	$\perp \mid$,	Ш	Ш		Α	DI	R	Щ.	1	Ц	2	Ø	\perp	Ц	11	\perp	Н	1	1	Ш	1	Ш	Ш	11	4	$\perp \downarrow$	\perp	Ш	\sqcup	44	41
0 5	0		Н	$\downarrow \downarrow$	\perp	1	4	22	1	\sqcup		01	Ш	1	Ц				$\perp \!\!\! \perp$	\perp	Ц			\perp	Н	$\perp \perp$	4	Н	4	11	Н	-	1	4	\coprod	4	44	+	4	\sqcup	44	41
0 6	0			Н	_	Н	1	\mathbb{H}	11	\coprod	\perp	4	\sqcup	\perp	Щ	C	<u> </u>	5	\Box	1		2	Ø	\perp	Ц	$\perp \downarrow$	1	4	-	11	Н	_	1	#	11	+	44	4	Н	H	4	41
0 7	0		<u> </u>	\mathbb{H}	+	\parallel	\perp	\vdash	4	\mathbb{H}	11	-	1	Н	\sqcup	\perp	4	\perp	4	\perp	H		\perp		-	$\perp \downarrow$	\perp	$\!$	+	#	H	+	H	44	11	+	+	+	H	\sqcup	++	41
0 8	0		+	++	+	\mathbb{H}	+	⊬	₩.	+	+	+	-	H	\vdash	\mathbb{H}	+	++	+	+	H	-	\sqcup	-	H	++	+	+	+	+	H	+	H	+-	+	+	+	+	1	\mathbb{H}	++	41
0 9	0		\vdash	+	+	H	+	H	╁┼	+	+	+	\vdash	\vdash	H	+	+	+	+	+	Н	+-	H	-	H	+	+	${\mathbb H}$	+	+	H	+	+	+	++	+	+	+	\vdash	H	+	$\ \ $
1 1	0		\vdash	\forall	+	+	+	\vdash	+	+	+	+	\vdash	+	H	+	+	++	+	+	\vdash	+	+	+	H	++	+	H	+	++	H	+	+	╁┼	+	+	+	+	H	H	++	\parallel
1 2	0	⊢ +−	\vdash	H	+	+	╁	H	╁┼	+	+	+	\vdash	\vdash	H	+	+	+	+	+	\vdash	+	\forall	+	+	+	+	H	+	╁┼	H	+	╁	+	+	+	+	+	\vdash	+	++	11
1 3	0	-	+	+	+	H	+	H	+	$\dagger\dagger$	$\dagger \dagger$	+	+	+	†	\dagger	+	+	+	+	H	+	\dashv	+	H	$\dagger\dagger$	+	$\dagger\dagger$	+	+	\forall	+	+	++	$\dagger\dagger$	+	$\dagger \dagger$	+	1	$\dagger\dagger$	+	11
1 4		⊢+-	$\parallel \parallel$	\forall	\top	11	1	$\dagger \dagger$	$\dagger \dagger$	11	$\dagger \dagger$	+	H	$\dagger \dagger$	\dagger	$\dagger \dagger$	†	$\dagger\dagger$	$\dagger \dagger$	\dagger	\vdash	\dagger	\vdash	\dagger	$\dagger \dagger$	$\dagger\dagger$	\dagger	$\dagger\dagger$	+	$\dagger \dagger$	H	\dagger	$\dagger \dagger$	tt	$\dagger \dagger$	\dagger	$\dagger \dagger$	\dagger	$\dagger \dagger$	11	$\dagger \dagger$	11
1 . 1 . 1	0	1 (+-+	+-	++	+	+	++	++	+	+	+-	+	\vdash	1	+	11	11	+	ΓŤ	\top	\sqcap	\top	TT	+	+	11	+	++-	+	+	+	+-+-	++	\rightarrow				1	11	٦1
1 5	0							11			11				11		-	1 1		-		1		- 1	1.1	1	_		J		LI									\perp !		
	0				\pm	H	+	+	H	\forall	\parallel	+		\parallel		\dagger	\pm	\parallel	$\pm \dagger$	\pm	+	T	H	\pm			†	\parallel	1		$\parallel \parallel$			\coprod	\coprod	1		$\frac{1}{2}$	H	\coprod	\coprod	
	0 0				+							+					+												+							+						
	0 0 0											+					+												1													
	0 0											+																														

Figure 14. Job 9: Detail Printing from a Sequential File (Part 2 of 2)



CUSTNO = Customer number ST = State code

CO = County code CITY = City code NAME = Customer name

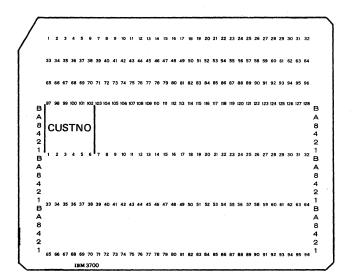
= Customer address (street) ADDR CITST = Customer address (city and state)

CRLIM = Credit limit CHARGE = Current month charges PAY Current month payments CREDIT Current month credits

BAL Balance

YTDSLS Year-to-date sales amount YTDNO Year-to-date number of sales

DELETE = Delete code



A previous program checked the master customer file for customers whose credit was below the credit limit. The output from this file was a deck of cards with the customer numbers punched in columns 1-6. This program will print a list of those customers with poor credit.

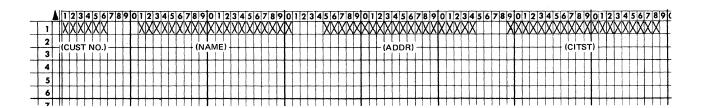
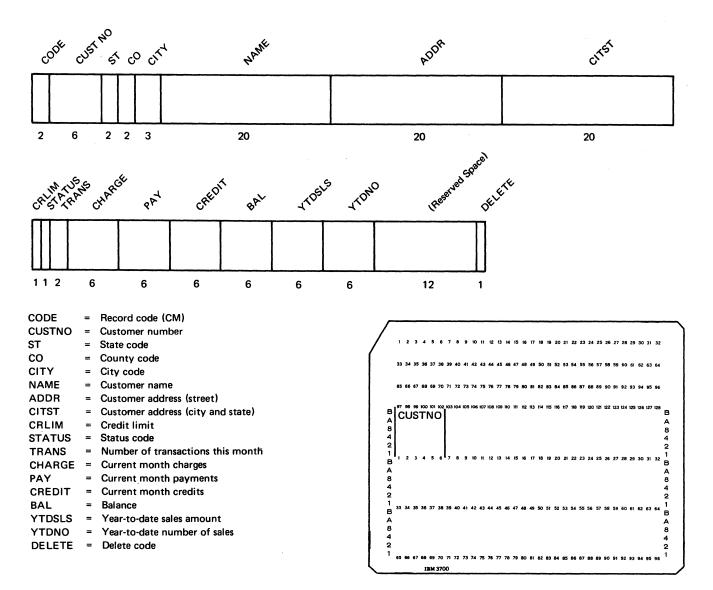


Figure 15. Job 10: Using Matching Records with a Sequential File (Part 1 of 2)

			File Description Sp	pecifications										
Line	Filename 8844-1 807 8 9 10 11 12 13 14	File Type File Designation End of File Sequence File Format Block Record Length 15 16 17 18 19 20 21 22 23 24 25 26	Mode of Processing Length of Key Field or of Record Address Field Record Address Type Type of File Organization or Additional Area Overflow Indicator Key Field Starting Location 22 28 38 31 32 [33:38] 35 36 37 38 39 44	Device Symbolic Device	Extent Exit for DAM Name of Label Exit 7 9 Core Index 2 53 54 55 56 57 58 59 60 61 62 63 64 65 66	File Addition/Unordered Number of Tracks for Cylinder Overflow Number of Extents Tape Rewind File Condition U1-U8								
0 2 0 3 0 4 0 5	F CARDIN F LISTING F Columns 28,	P AF 256 12 15 AF 96 9 9 9 9 9 9 9 9	6	0/5K MFCU1 PRINTER fined, and it will be pro-	cessed consecutively.									
1DM	F	<u> </u>	International Business Machi		<u> </u>	Form X21-9094								
TĐW	Printed in U.S.A. RPG INPUT SPECIFICATIONS 1 2 75 76 77 78 79 80													
Date			Punching Graphic Instruction		Page Program Identification									
Program .	ner		Instruction Punch											
Line	Filename Filename	Sequence Number (1-N) Option (0) Record Identifying Indicat or ** Not (N) CZZ/D CZZ/D	1 1111 11	Find the state of	Control Level (L1-L9) Matching Fields or Chaining F	ield ndicators Zero Sterling Sign Position								
0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 0 1 1 1 1 2	I SEQDISK	BB Ø2	M1 is assigne files will be considered (shown on the	3 86 16 35 36 55 75 1 6 6 75 1 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	CUSTNO MI NAME ADDR CITST CITST SCUSTNO MI Gields. These fields from the is found. At that time the irn on, and the fields defined printed.	e two input MR indicator								
IBM			International Business Mac			Form X21-9090 Printed in U.S.A.								
Program Programm	ner		PG OUTPUT - FORM Punching Graphic Unstruction Punch	IAT SPECIFICATIONS	1 2 Program Identification	75 76 77 78 79 80 on								
Line	Filename Filename Filename Filename	And Arter Select Fatch Overflow (F) 19 19 19 19 19 19 19 19 19 19 19 19 19	And Field Name (a) (a) (b) (c) (c) (d) (d) (d) (d) (d) (d) (d) (d) (d) (d	Position P Yes No	Edit Codes	Sign Sterling Sign Position								
0 1 0 2 0 3 0 4 0 5 0 6	o L / ST / NG o o o o o o o o o o o o o o o o o o o	D 2 MR	7,28,29 30,31,32,33,34,35,36,37,38,99,44 1	6 30 54 78 both on. Since	1 00 00 00 00 00 00 00 00 00 00 00 00 00	07 08 69 70 71 72 73 74								
0 8	1.1	cords O1 is used on the (

Figure 15. Job 10: Using Matching Records with a Sequential File (Part 2 of 2)



This job does the same function as Job 10; however, the file is indexed.

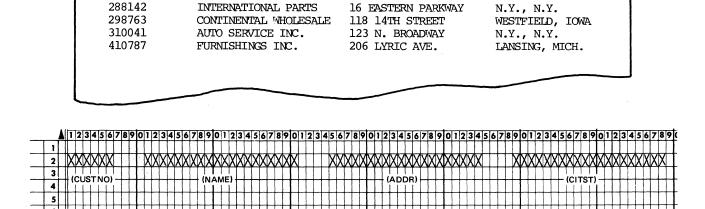


Figure 16. Job 11: Using Matching Records with an Indexed File (Part 1 of 2)

		File Description Specification	ons	
Line Filena 3 4 5 6 7 8 9 10 1 0 2 F CUSMS 0 3 F CARD	GO CONTROL OF THE FORMAT File Format Q	Mode of Processing Length of Key Field or of Record Address Field Record Address Type Type of File Organization or Additional Area of Cord address Type of File Organization or Additional Area of Cord address Type of File Organization or Additional Area of Cord add	Symbolic Name of Label Exit No. 2 No	Extent Exit for DAM File Addition/Unordered Number of Tracks for Cylinder Overflow Number of Extents Tape Rewind Core Index
0 4 F LTST	ING O F 80	dexed file. Since column 28 is blank,	the file will be processed sec	
IBM		RPG INPUT SPECIFICATI	ONS	Form X21-9094 Printed in U.S.A.
Program Programmer		Punching Graphic Instruction Punch	Page 0	2 75 76 77 78 79 80 2 Program Identification
Line Filen.	Position (O) u Position	Charac Charac (Charac Charac Charac Charac Charac Charac	Field Location	Field Indicators Field Indicators Sterling Sign Position Field Sign Field Indicators Sterling Sign Position Field Indicators Sterling Sign Position Field Indicators Sterling Sign Position Field Indicators
0 1 1 CUSM 0 2 1 1 0 3 1 1 0 4 1 1 0 6 1 CARD	STR AA Ø1		3 8ØCUSTNO 16 35 NAME 36 55 ADDR 56 75 CITST 1 6ØCUSTNO	MI
0 8 I		to the two CUSTNO fields. These fields and the fields defined on the	elds will be compared until a	a match is found. At that
IBM	tille will will	International Business Machines Corporation		Form X21-9090
DateProgrammer		Punching Graphic Instruction Punch	CIFICATIONS Page	Printed in U.S.A. 2
Line Filena	State (H/O/T/E) State Select/Ferch Overfit Before After Afte	put Indicators Comparison	Yes Yes 1 Yes No 2 No Yes 3 No No 4 Constant or Edit	CR
	ING D 2 MR			

Figure 16. Job 11: Using Matching Records with an Indexed File (Part 2 of 2)

For every program you run Operation Control Language (OCL) statements are needed. You must supply OCL statements to the system through the MFCU (cards) or the Printer-Keyboard (typed).

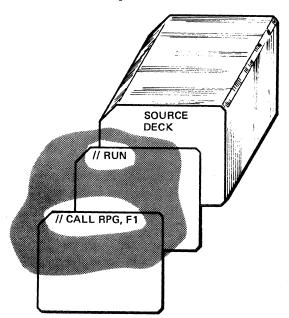
This chapter describes the statements needed to:

- Compile an RPG II program.
- Process a card file.
- Process a disk file.
- Process two disk files.
- Create a disk file.
- Process a disk file that uses external indicators.
- Perform several jobs, one after the other.

The function and the format of each statement are explained. Coding rules are also discussed.

OCL CARDS REQUIRED TO COMPILE AN RPG II PROGRAM

After your RPG II program is written and recorded in cards, it must be compiled. To compile an RPG II program two OCL cards are required.

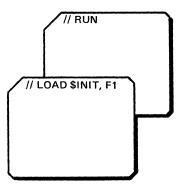


The first card, // CALL RPG, F1, tells the system to get the RPG II Compiler program from the fixed disk. The second card, // RUN, tells the system to run the compiler program. Your source deck must follow the // RUN card.

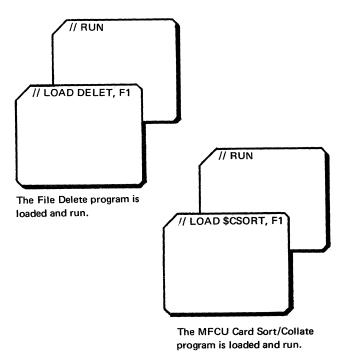
All OCL cards must contain // in columns 1 and 2, followed by at least one blank column and then a word like CALL or RUN.

OCL CARDS REQUIRED TO LOAD AND RUN SOME IBM PROGRAMS

Several IBM programs require only two OCL cards, // LOAD and // RUN. The following three examples show the OCL cards needed to load and run three IBM programs.



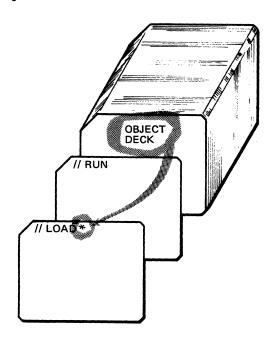
The Disk Initialization program is loaded and run.



61

OCL CARDS REQUIRED TO LOAD AND RUN AN OBJECT PROGRAM THAT USES CARD FILES

To run a certain job, the object program must be loaded into storage. The result of compiling a program is an object program. To load an object program that is on cards (object deck), an * must follow the word LOAD. The * tells the system that an object deck follows the RUN card. // LOAD * and // RUN are the only two OCL cards needed to load and run RPG II programs that use no disk files. For example, only these two cards are required for a program that prints data from a transaction card file.



OCL CARDS REQUIRED TO LOAD AND RUN A PROGRAM THAT USES ONE DISK FILE

To load and run an object program that uses a disk file, another OCL card is required: a FILE card. Three items of information must follow the word FILE: the name of the file, the name of the disk pack the file is on, and the location of the disk pack on the disk drive.

Before discussing the specific entries some terms need defining. OCL statements can contain two types of information: statement identifiers and parameters. A statement identifier tells one statement from another. CALL, LOAD, RUN, and FILE are statement identifiers. A parameter is additional information required with statement identifiers. Some OCL statements do not require parameters; RUN is such a statement.

A parameter is information defined by you or by IBM. For example, in the LOAD * statement, the * is defined by IBM. (The * means an object deck is being loaded from

cards.) The information you define is variable information, such as the name of a file which you determine or the disk on which an IBM program is stored.

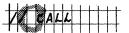
OCL has positional parameters and keyword parameters. A positional parameter contains information that must be supplied in a specific order; that is, information that occupies a specific position in an OCL statement. Such a statement is:

// CALL RPG, F1

CALL is the statement identifier. RPG must be specified first, followed by F1; thus RPG and F1 are positional parameters.

Certain coding rules must be followed for positional parameters.

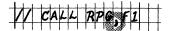
1. Leave at least one blank between // and the statement identifier.



2. Leave at least one blank between the statement identifier and the first parameter.



3. Use a comma to separate parameters.



<u>Note:</u> When several parameters are needed, they must appear in a specific order.

Keyword parameters contain an IBM defined word (keyword) followed by information that you supply. Keywords are required to tell one parameter from another. The FILE statement, introduced earlier, requires three keywords: NAME, PACK, and UNIT. The order in which they appear is not important.

The other information is either defined by you or IBM may offer several options. For example, to process a job with one file, such as Job 9, the keywords and the information you supply are:

NAME-(file name)

PACK-(name of disk pack)

UNIT-(location of disk pack)

In Job 9 SEQDISK, the name of the disk file, resides on a removable disk pack that is named VOL1. The keyword parameters for Job 9 are:

NAME-SEQDISK

PACK-VOL1

UNIT-R1

Both SEQDISK and VOL1 are names determined by you. Since the file resides on a removable disk, R1 is used with UNIT. R1 is one of four options defined by IBM. These options and their meanings are:

R1 = Removable disk on drive one

F1 = Fixed disk on drive one

R2 = Removable disk on drive two

F2 = Fixed disk on drive two

You must supply the code that tells on which unit your file resides.

When several keyword parameters are needed in an OCL statement, certain coding rules must be followed.

1. Leave at least one blank between // and the statement identifier.



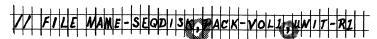
2. Leave at least one blank between the statement identifier and the first keyword.



 Separate the keyword and the information you supply with a hyphen (-).



4. Use a comma to separate keyword parameters.



Note: When several keyword parameters are needed, they may appear in any order.

To follow these rules and ensure your OCL statements are correct, you can use a 96-column coding sheet. The OCL statements for Job 9 (Figure 14) are coded as:

IBM

International Business Machines Corporation 96 COLUMN GENERAL PURPOSE DATA RECORDING DOCUMENT

System				Pun	ching Instructions
Program		Graphic			Can
Programmer Date		Punch			
1 4 8 12 16 20 24 28 32 36 40 44 48	52	56	60	64	68 72
7/	TŤT	ППП	ΪП	Ť	TÎT TÎ
	+++	++++	HH	+H	++++++
// FILL WAME SEQUISK PACK WOLL UNIT RI	+++ +	 	HH	+	
THE STATE OF THE S	++++	†† † †††	HHH	+ 1 +	
	++++	†† † ††	$ \mathbf{I} $	11	†† † ††† †
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	 	†† † ††	HH	+	

OCL CARDS REQUIRED TO CREATE A DISK FILE

To create a disk file, sequential or indexed, you must tell the system the size of the file and the use of the file. To state the file size, two keywords are available: TRACKS and RECORDS. You may use one or the other, but not both.

If you use RECORDS, the system calculates the disk space required and converts it to tracks for you. The system performs the same calculations as described in Chapter 2.

If you use the TRACKS parameter, there is no need for the system to perform these calculations.

The RETAIN parameter tells the system how to classify the use of a file. A file can be classified as scratch, temporary, or permanent. A scratch file is normally used only once. When the program using it ends, all or any part of the area containing the file can be used by the next program. Data in the scratch file can be accessed until the area containing the file is used for another file.

A temporary file is usually used more than once and possibly by more than one program. The area containing a temporary file can be used by some other file under any one of the following conditions:

- 1. A FILE statement defining a scratch file is supplied for the temporary file. This converts the temporary file to a scratch file.
- 2. Another file is loaded into the *exact* area occupied by the temporary file.
- 3. The File Delete program is used to delete the file.

A permanent file is like a temporary file. However, the area containing a permanent file cannot be used for any other file until the File Delete program deletes the permanent file.

A file is classified as scratch, temporary or permanent when it is created. If at that time the RETAIN parameter is omitted, the file is assumed to be a temporary file.

RETAIN must be followed by a code that indicates the classification of the file. The possible codes are:

S - scratch

T - temporary

P - permanent

The OCL statements required to create the name and address file used throughout Chapter 3 are as follows.

IBM

International Business Machines Corporation 96 COLUMN GENERAL PURPOSE DATA RECORDING DOCUMENT

System			Punching Instruction	ns
Program		Graphic		ard Fo
Programmer	Date	Punch		
1 4 8 12 16 20 24 28 32 36 40 44	48 52	56 60	64 68 72	·
 				
/// LIOAD &				П
// F/LE NAME-SERDISK, PACK-VOLL, UMIT-RI, RECORDS-	6720 RE	TAIN-P	† !	ff
VI RUM I I I I I I I I I I I I I I I I I I I		1 7 7 7	┞┋┼┼┼╂┼┼	+
┖┍┪╒╫╒╫╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒╒	▎ ▋┤┼┤╏┤	╎╎╏╎╎╏╎	┞╋┖┼┼╂┼┼┼	H
┍╶╸╸╸╸╸╸╸╸╸╸╸╸╸╸╸╸		╎╎╏╎╎╏ ╎┼	┖┋┋ ┼┼┼┼	

Since this file is a master file, it is classified as *permanent*. (RETAIN-P).

The TRACKS or RECORDS and RETAIN parameters tell the system how much disk space to use and how long to keep this space.

OCL CARDS REQUIRED TO LOAD AND RUN A PROGRAM THAT USES MORE THAN ONE DISK FILE

One FILE card is required for each disk file used in a job. To load and run an object program that uses two disk files, two FILE cards are required. In Chapter 3, Job 4 explained how to re-create a sequential file. Two disk files were used, one as input (the file to be re-created) and one as output (the newly created file); thus, two FILE cards are required.

IBM

International Business-Machines Corporation 96 COLUMN GENERAL PURPOSE DATA RECORDING DOCUMENT

System				Punching	Instructions
Program	Gra	aphic.	П		Card Fc
Programmer Date	Pui	nch		$\Pi\Pi$	
·					
1 4 8 12 16 20 24 28 32 36 40 44 48	52 50	8 60	6-	4 68	72
// LJOAD *					
VI FILLE MAME-SEQDISK, PACK-VOLZ, UMIT-RZ	ППП	$\Pi\Pi\Pi$	$\Pi\Pi$	$\Pi\Pi$	
VIFILE WAME-NEWSER, PACK-VOLL, UNIT-RI, RECORDS-6720.	RETAI	U-P	$\Pi\Pi$		
		1	111	$\mathbf{I} \cap \mathbf{I}$	
		11111	+++	1	!
┣╇╅╃╋╃╃╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇╇	 	1-1-1-1			

OCL CARDS REQUIRED TO RUN A JOB THAT USES A DISK FILE AND EXTERNAL INDICATORS

In Chapter 3, Job 8 (Figure 12) used external indicators. You were told that these indicators are set by an OCL statement. This OCL card is // SWITCH. For Job 8, (Figure 12) the OCL cards required are:

IBM

International Business Machines Corporation 96 COLUMN GENERAL PURPOSE DATA RECORDING DOCUMENT

System						Punching Instructions
Program					Graphic	Card Fo
Programmer				Date	Punch	
1 4 8	12 16	20 24 28	32 36 40	44 48	52 56	60 64 68 72
// LOAD *						
/ FILE W	AME-INVMS	TR, PACK-VO	11.4W/T-R2			
// SWITCH	10000000					
/ RUN						

In the SWITCH statement, a 1 is followed by seven zeros. Since RPG II allows up to eight external indicators to be used, the SWITCH statement must tell which of these indicators to set and how to set them.

In the SWITCH statement the eight characters correspond to the eight external indicators.

U1	U2	U3	U4	U5	U6	U7	U8
1 ON	0 OFF	OFF .	0	0	0	0	0
1							

A 1 means set the indicator on; a 0 means set the indicator off. Job 8 uses only one external indicator (U1); thus a 1 appears in the U1 position.

Note: The SWITCH statement can also contain an X. X means do not change the indicator status.

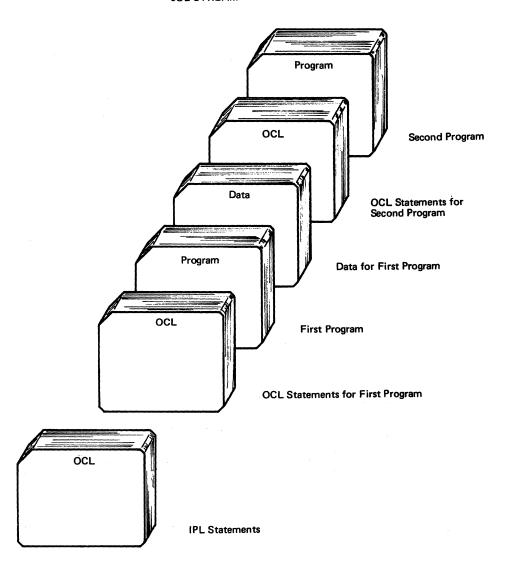
OCL CARDS REQUIRED TO RUN SEVERAL JOBS, ONE AFTER THE OTHER

After the system reads a set of OCL statements for a job, it loads the program for that job and runs the job. When that job is completed, another job can be run.

The OCL statements you supply for several jobs form a job stream.

Note that the first group of OCL statements are called *IPL* statements. IPL statements are those read by the system during Initial Program Load (IPL). IPL is an operating procedure the operator uses to initially start the system.

JOB STREAM



// DATE STATEMENT

Before the system will run any of your jobs, it must read a special OCL card, // DATE, at IPL time. The date supplied at IPL time is called the *system date*. The system date is saved by the system and is available to your RPG II programs. For example, to print a date in the heading of a report, the RPG II program can reference the system date by using the RPG II reserved words: UDATE, UDAY, UMONTH, and UYEAR.

// DATE must be followed by a date. The date can be entered in several ways:

mmddyy mm/dd/yy mm-dd-yy ddmmyy

dd-mm-yy

However, you must specify an edit word to format the system date at output.

The DATE card can also be used to specify a date different than the system date. For example, at the beginning of each IPL you supply a system date reflecting that day's date. However, during that day you may want some reports, bills for example, with a different date. The OCL cards in Figure 17 illustrate how to specify a date different than the system date.

program's OCL cards also change the date.

SUMMARY - OCL STATEMENTS

For every job you run, a set of OCL statements is required. Some jobs, such as compiling an RPG II program, require only two statements: CALL and RUN.

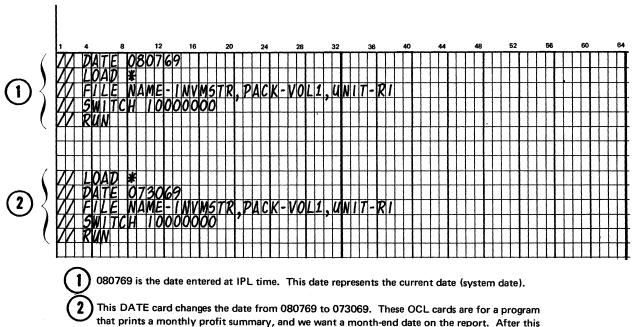
Other jobs require more statements, depending upon the requirements of the job. All jobs involving disk files must have one FILE statement for every disk file in the job.

Various keywords are required in the FILE statement depending upon what the job does. To process a disk file, NAME, PACK, and UNIT are required. To create a disk file, NAME, PACK, UNIT, TRACKS or RECORDS, and RETAIN are required.

If a job uses external indicators the SWITCH statement is required. Other jobs may require DATE.

There are several important points to remember about OCL statements:

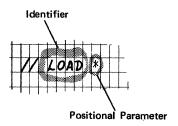
- 1. OCL statements contain identifiers. LOAD, CALL, RUN, and FILE are identifiers.
- Some OCL statements contain only identifiers. RUN is such a statement.



program is run, the system date (080769) will be available to the next program to use, unless that

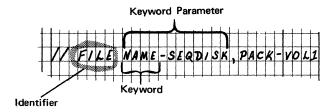
Figure 17. Using the DATE Card

3. Some OCL statements contain postional parameters.

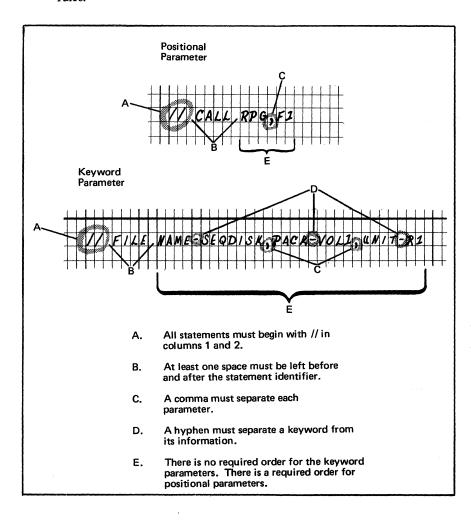


The IBM System/3 Disk System Reference Manual, Form C21-7512, contains a thorough explanation of all statements described in this chapter.

4. Some OCL statements contain keyword parameters.



- 5. The OCL statements, the programs, and the data for several jobs form a job stream.
- 6. OCL statements must be coded according to certain rules.



In Chapter 1, the need for sorting a disk file was discussed as a factor to consider when choosing a file organization method. Often a master file is used in various applications to produce a variety of reports. Some of these reports may require data printed in an order different than the order of the disk file. The IBM Disk Sort program is available to sort disk files. It can produce two types of output disk files that can be used as input to your RPG II programs. This chapter explains these two types of output files.

For additional information on the Disk Sort program, see the *IBM System/3 Card Sort/Collate Programmer's Guide*, Form C21-7539, and the *IBM System/3 Disk System Utility and Sort Programs Reference Manual*, Form C21-7522.

This chapter assumes you know how to:

- 1. Write Header lines that:
 - a. Specify sum of lengths of control fields
 - b. Specify sequence (ascending or descending)
- 2. Write Record Type lines that:
 - a. Include all records
 - b. Omit certain records
 - c. Use the AND relationship
 - d. Use the OR relationship
 - e. Use the EQ, LE, NE, GT, GE, and LE relationships
 - f. Specify constants in Factor 2
 - g. Specify fields in Factor 2
- 3. Write Field lines that:
 - a. Indicate the order of importance of the fields
 - b. Specify type of field (normal, opposite, data, or forced)

TAG-ALONG SORT

With both the Card Sort/Collate program and the Disk Sort program, you can sort all records or only certain records. You specify which records you want sorted by describing them on Record Type lines. However, with the disk sort an additional function is provided.

Your output records can contain all the data from the input records, or your output records can contain only certain fields. For example, the customer master file described in Chapter 2 and created in Chapter 3 will be used to produce a report called Sales Analysis (Figure 18).

Note that this report contains four fields: three codes (state, county, and city) and a sales amount. Since the master file is in customer number order, it must be sorted in order by state, within county, within city. So these three codes plus sales amount are the only fields needed to produce the report. The three codes are control fields. The other field, sales amount, is a *data field*. Data fields are those fields (in the output record) that *tag-along* with control fields. A tag-along sort is a job where data fields are included in the output record.

A tag-along sort can produce two types of output records. The output records can either retain or drop the control fields. The Sales Analysis report is an example of when control fields are retained. Since the control fields are printed, they must be retained in the output record. However, if you have some reports that must be in a specific order, but you do not want to print the control fields, then these fields can be dropped from the output record.

Describing Output Records that Retain Control Fields

To produce an output record that retains control fields, three new entries are required on the Sequence Specification sheet. Two entries are required on the Header line, and one entry is required on the Field lines. On the Header line, SORTR (columns 7-11) identifies the job as a tag-along sort. You must also specify the length of the output record in columns 29-32.

	SALES ANALYSIS						
	Cmama	COUNTY	CITY		SALES A	MTT	
	STATE	COUNTY	CIII		SAULO A	7	
	2	103	100	\$	1,000.00		
			104	\$	2,750.00		
			208	\$	1,830.00		
				\$	5,580.00	COUNTY TOTAL	
		107	75	\$	3,550.00		
			210	\$	1,770.00		
			275	\$	2,990.00		
			325	\$	3,350.00		
				\$	11,660.00	COUNTY TOTAL	
				^	17 040 00	CUID III II CUID T	
			•	\$		STATE TOTAL	
	6	293	400	\$	940.00		
4							

Figure 18. Sales Analysis Report

On the Field lines, you must describe your data fields. Data fields are described in the same way as control fields, with one exception. You must enter a D in column 7; the D specifies that the field described on this line is a data field. A D card must not follow N, O, or F cards for a given record type.

IBM	International Business Machines Corporation	Form X21-9089 Printed in U.S.A.
IBM	SEQUENCE SPECIFICATIONS Header	Page
Line Job (i) Match (ii) Sopr Sum of (ii) (iii) (SORTA	1 2
SORT Sum of Lengths of Stacker Select No. Stacker Select	Output Job Description Record Length 8 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 69 60 61 62 63 64	85 66 67 68 69 70 71 72
OOOHSORTR	Record Type	
Line Factor 1 Rel. Factor 2 (F	Field or Constant) Comments	
I ├──────────		
Number Solution Control Contro	Record Name	
3 4 5 6 7 6 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 0 1	8 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64	65 66 67 68 69 70 71 72
	┤┤╎╎╎╎┼┼┼┼╏╎┼┼┼┼┼┼╏	
0 3		
0 4		
0 5		
0 6		
	Control Field	
Line Forced	Comments	
Number G(Z) Location Locati	Field Name N	65 66 67 68 69 70 71 72
0 7 F		
0 8 F		
0 9 F D		++++++
1 0 F	┤╎╎╎╎╎╎╎╎╎╎	
1 2 6		++++++
1 3 F	┦╎╎╎╎╎╎╎╎╏ ╎┼┼ ╏ ╎┼┼┼┼┼┼┼┼┼┼┼┼┼┼┼	
1 4 F		

In a tag-along sort, output records are always written in the same way.

- Control fields are placed first in the new record. On the Sequence Specifications sheet, the control field described first (most important) is placed first in the output record; the control field described second (second most important field) is placed second, and so on.
- 2. Data fields are placed immediately following the control fields.

Figure 19 shows the specifications needed to produce the file that will be used to print the Sales Analysis report.

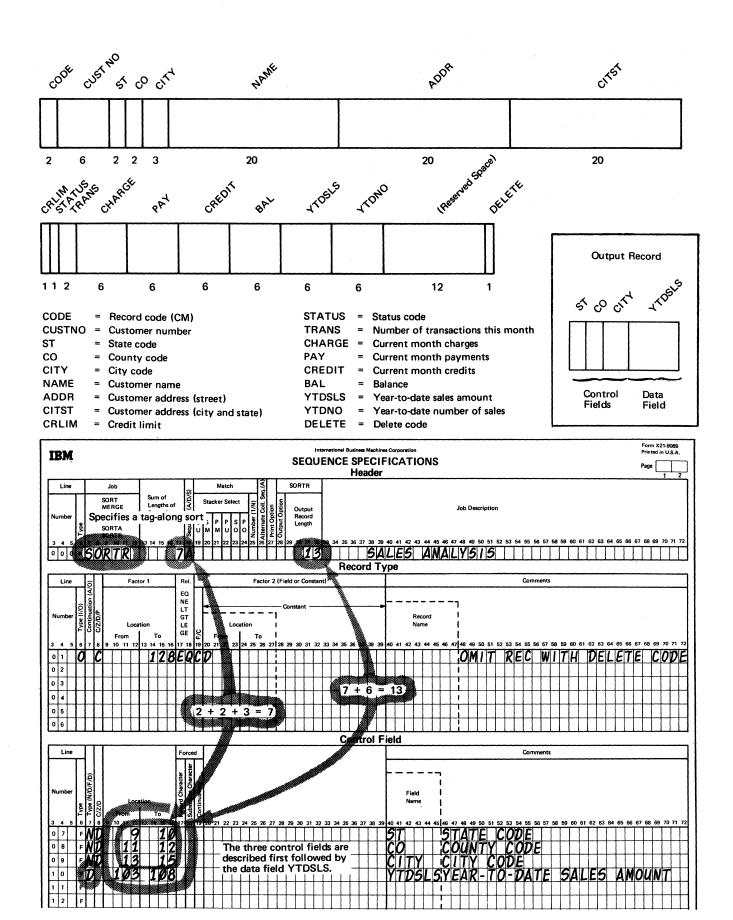
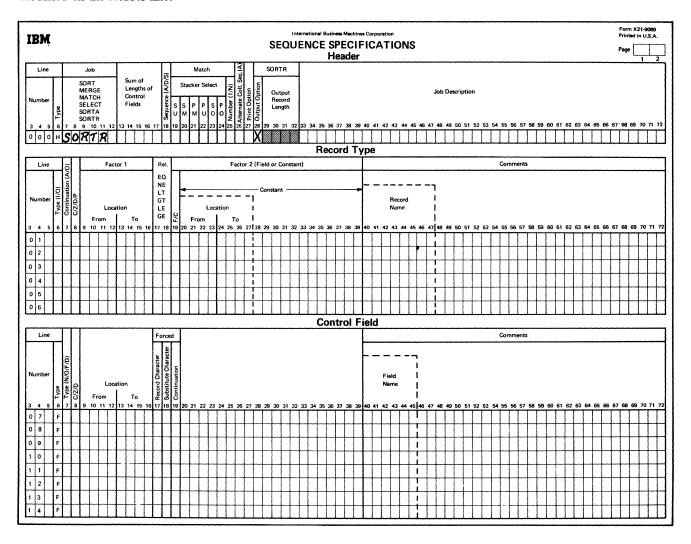


Figure 19. Tag-Along Sort Job

Describing Output Records that Drop Control Fields

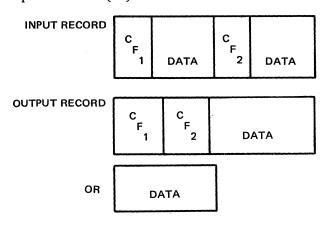
To drop control fields from the output record, an additional entry is required and one other entry must be modified on the Header line.



The X in column 28 tells the Disk Sort program to drop the control fields from the output record. Columns 29-32 are still used to indicate the length of the output record. However, you must *not* include the sum of the lengths of the control fields, since control fields will not appear in the output record.

REVIEW-TAG-ALONG SORT

A tag-along sort produces output records that retain or drop control fields (CF).



The Header line and Field lines contain entries that identify the type of output record desired.

Header Line

- 1. SORTR (columns 7-11) identifies a tag-along sort.
- 2. An X in column 28 means drop control fields; a blank means retain control fields.
- 3. The number in columns 29-32 specifies the length of the output record. (Do not include control field length if X is entered in column 28).

Field Lines

- 1. Control fields are described in order of importance.
- 2. Data fields (D in column 7) are described following the control fields.

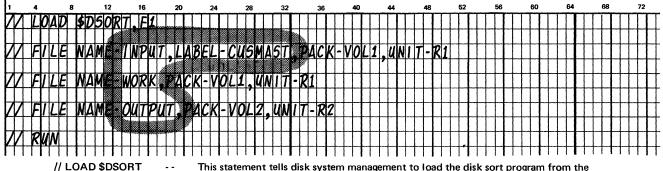
ADDROUT SORT

The other type of disk sort is ADDROUT. An ADDROUT sort produces a disk file that contains only the relative record numbers of the desired records.

To produce an ADDROUT file, SORTA is placed in columns 7-11 on the Header line. No other disk sort entries are required on the Header line. However, you must describe your control fields on Field lines. Even though an ADDROUT file contains only relative record numbers these numbers are sorted in the sequence you specify in column 18 on the Header line. The control fields are needed to produce this sequence.

OCL REQUIRED WITH DISK SORT PROGRAM

You use the Disk Sort program when you want to perform an ADDROUT or a tag-along sort job. For the Disk Sort program to do the job you specify, you must use certain OCL statements. Figure 20 illustrates and explains these statements.



This statement tells disk system management to load the disk sort program from the fixed disk. \$DSORT is the IBM name for the Disk Sort program. F1 is a fixed disk.

// FILE

As you know, every file used in a program must be defined by a // FILE statement. For a disk sort job, you must always define three files: an input file, a work file, and an output file. The input file is the file you want sorted. The work file is space on the disk that the program uses to do the sort job. The output file is the new file created from the input file as a result of the sort job.

The keyword parameters for NAME are INPUT, WORK, and OUTPUT. These three words are predetermined by IBM. For the program to use a file in one of these three ways, you must correctly use these three words.

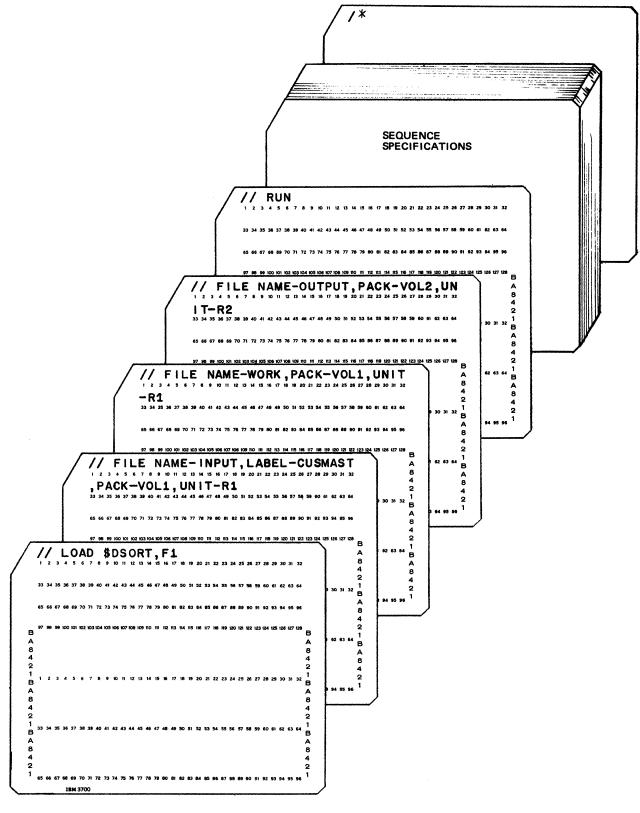
// RUN

A RUN statement is always the last OCL statement for a job.

Figure 20. OCL Statements for a Disk Sort Job

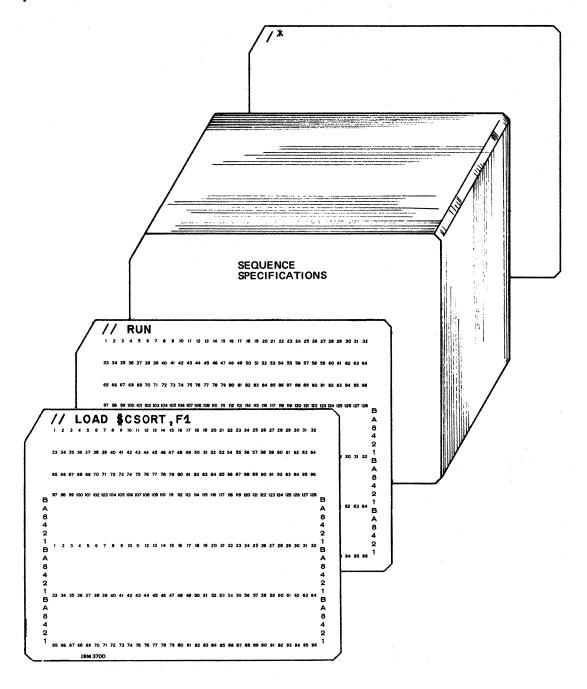
The OCL cards precede the Sequence Specification cards in the MFCU.

/* is needed to tell disk system management that the last specification card has been read. At that time, the Disk Sort program will begin the specified sort job.



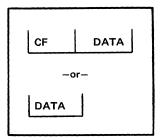
OCL REQUIRED WITH CARD SORT/COLLATE PROGRAM

You use the Card Sort/Collate program when you want to sort, merge, match or select cards. The Card Sort/Collate program resides on disk. For it to do the job you specify, you must use certain OCL cards.



The relationship of these cards to the Sequence Specifications is the same as for a disk sort.

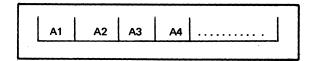
The name of the Card Sort/Collate program is \$CSORT. The LOAD card must contain this name. LOAD and RUN are the only OCL cards required for a card sort. Again /* must be the last card after the Sequence Specifications. As for a disk sort, /* tells the system that all specifications have been read.



REVIEW-SORT

The Disk Sort program can produce two types of output files, tag-along and ADDROUT. The tag-along sort can produce output records with or without control fields.

The ADDROUT file contains the relative record numbers of the sorted records.



Coding considerations for tag-along and ADDROUT sorts are shown in Figures 21 and 22.

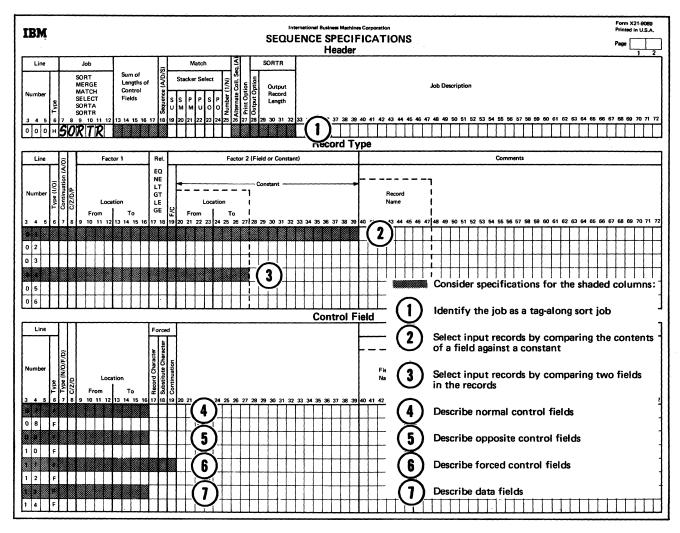


Figure 21. Column Summary (Tag-Along Sort)

r			Form X21-9089
IBM	International Business Machines		Printed in U.S.A.
•	SEQUENCE SPECIF Header	-ICATIONS	Page 1 2
Line Job	Match € SORTR		
SORT Sum of Action of Acti	S S P P S P D S D D D D D D D D D D D D	Job Description 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	5 66 67 68 69 70 71 7 2
OOOHSORTA	1 Record Ty	/pe	ЩШ
Line Factor 1 Rel.	Factor 2 (Field or Constant)	Comments	
Number 1	Constant Location From To	Record Name 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	5 66 67 68 69 70 71 72
3 4 5 6 7 6 9 10 17 12 13 14 15 16 17 16	19 20 21 22 23 24 25 26 27 26 29 30 31 32 33 34 35 36 37 36 39		
0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3	Consider specifications for the share	ded columns:
	Control F	ield	
Line Force		1 Identify the job as an ADDROUT Select input records by comparing of a field against a constant	. 1
Number O O Location D D O O O O O O O O		Field Name Select input records by comparing in the records	two fields
0 8 F	5	Describe normal control fields	-
1 0 6 1 1 1 1 1 1 1 1 1	6	5 Describe opposite control fields	·
1 3 F 1 1 4 F		Describe forced control fields	

Figure 22. Column Summary (ADDROUT Sort)

// CALL statement 61	creating a disk file
// DATE statement 68	coding the file description sheet 21
// FILE statement 62	definition 3
••	OCL 64
NAME parameter 62 PACK parameter 62	with indexed file organization 5, 22
	with sequential file organization 4, 22
RECORDS parameter 64	cyclic, definition 3
RETAIN parameter 64	• •
TRACKS parameter 64	data field, definition 71
UNIT parameter 62	DATE statement 68
// LOAD statement	delete code
// LOAD* statement 62	definition 6
// RUN statement 61	record layout consideration 13
// SWITCH statement 66	deleting records
- Airitan of - 6:1- 0	definition 3, 6
activity of a file 9	with indexed file organization 6
adding records to a file	with sequential file organization 6
definition 3	designing a record (see record layout)
with indexed file organization 5, 23	disk address, definition 2
with sequential file organization 5, 30	Disk Copy/Dump program 6
ADDROUT sort	disk file characteristics (see file characteristics)
file considerations 10	disk file creation (see file creation)
function 10, 76	disk file maintenance (see file maintenance)
OCL statements 76	disk file organization (see file organization)
attributes, of a file (see file classification)	Disk Sort program
1 . 1	ADDROUT 76
batch processing, definition 9	file considerations 9
block, definition 14	OCL 76
block length	tag-along 71
coding 21	display file, definition 39
definition 15	DSPLY operation code 43
blocked records 14	
anlawla tima	expansion considerations 17
calculating	external indicators
index space 17	for calculation conditioning 47
record space 17	for file conditioning 47
calculation operations, conditioning of 47	for output conditioning 47
CHAIN expection and 25	
CHAIN operation code 35	
DSPLY operation code 43	file acrivity 9
CALL statement 61	file characteristics
card file organization 1	activity 9
Card Sort/Collate program	size 9
for decreasing processing time 1	use 7
functions 78	volatility 8
OCL 79	file classifications 64
card sorters, use of 1	file conditioning (by external indicators) 47
chaining, definition 35	file creation
CHAIN operation code 35	coding the file description sheet 21
characteristics, file (see file characteristics)	definition 3
choosing file organization 6	OCL 64
classification, of a file 64	of indexed files 5, 22
compiling an RPG II program 61	of sequential files 4, 22
conditioning	file description sheet
of a file 47	adding records 23
of calculation operations 47	creating a disk file 21
of output operations 47	with indexed file organization 22
consecutive processing 1	with sequential file organization 22
control field, definition 1	updating records 31

file expansion considerations 17	loading (see also creating)
file location 18	an object program 62
file maintenance	definition 3
adding records	location of a disk file 18
to indexed files 5, 23	
to sequential files 5, 30	maintenance, of a file (see file maintenance)
definition 3	master file, definition 7
deleting records 6	
reorganizing a file 6, 8	NAME parameter 62
updating records	
in indexed files 34	object program
in sequential files 34	definition 62
file organization	loading 62
card 1	OCL (operation control language) 61
choosing 6	for ADDROUT sort 76
definition 1	for Card Sort/Collate program 78
indexed 1	for external indicators 66
	for Disk Sort program 76
(see also indexed file organization)	loading an object program 62
orderered 2	
sequential 1	to create a disk file 64
(see also sequential file organization)	on-line processing, definition 9
unordered 2	operation control language (OCL) 61
file size	(see also OCL)
determining 9, 16	ordered file organization 2
expansion considerations 17	organization, file (see file organization)
FILE statement 62	output-format sheet
NAME parameter 62	coding for adding records 23
PACK parameter 62	coding for updating records 31
RECORDS parameter 64	output operations, conditioning of 47
RETAIN parameter 64	output records (tag-along sort)
TRACKS parameter 64	dropping control fields 75
UNIT parameter 62	retaining control fields 71
file use 7	10mining out and 110100
file volatility 8	PACK parameter 62
ino volumity o	parameters
identifiers, statement 62	definition 62
index, definition 2	keyword 62
indexed file organization	positional 62
advantages 2 creation 5, 22	permanent files, definition 64
	positional parameters 62
definition 2	Printer-Keyboard
ordered 2	as a second printer 51
maintenance	for updating records 39
adding records 5, 23	processing
deleting records 6	batch 9
reorganization 6	consecutive 1
updating 6, 34	randomly by key 3, 35
processing 2	sequential 2
unordered 2	sequential by key 2
index space calculations 17	sequential within limits 3
indicators, external 47	bod and the state of the state
initial program load (IPL) 67	random processing by key 3
input sheet, coding to update a file 31	record addition
inquiry, definition 7	definition 3
INI CIRTURAL RECORDED IN COLUMN IN C	to indeped files 5 23
IPL (initial program load) 67	to indexed files 5, 23
IPL statements 67	to sequential files 5, 30
IPL statements 67	to sequential files 5, 30 record deletion
	to sequential files 5, 30 record deletion definition 3, 6
IPL statements 67 job stream, definition 67	to sequential files 5, 30 record deletion definition 3, 6 to indexed files 6
IPL statements 67 job stream, definition 67 keyfield, definition 2	to sequential files 5, 30 record deletion definition 3, 6 to indexed files 6 to sequential files 6
IPL statements 67 job stream, definition 67	to sequential files 5, 30 record deletion definition 3, 6 to indexed files 6 to sequential files 6 record design 13
IPL statements 67 job stream, definition 67 keyfield, definition 2 keyword parameters, definition 62	to sequential files 5, 30 record deletion definition 3, 6 to indexed files 6 to sequential files 6 record design 13 (see also record layout)
IPL statements 67 job stream, definition 67 keyfield, definition 2 keyword parameters, definition 62 layout, of a record 13	to sequential files 5, 30 record deletion definition 3, 6 to indexed files 6 to sequential files 6 record design 13 (see also record layout) record layout 13
ipL statements 67 job stream, definition 67 keyfield, definition 2 keyword parameters, definition 62 layout, of a record 13 (see also record layout)	to sequential files 5, 30 record deletion definition 3, 6 to indexed files 6 to sequential files 6 record design 13 (see also record layout) record layout 13 block length 14
IPL statements 67 job stream, definition 67 keyfield, definition 2 keyword parameters, definition 62 layout, of a record 13	to sequential files 5, 30 record deletion definition 3, 6 to indexed files 6 to sequential files 6 record design 13 (see also record layout) record layout 13

extra space 13 field size 13 naming fields 13	sub-multiple, definition 15 SWITCH statement 66 system date 68
record length 14 record space calculations 17 relative record numbers (ADDROUT sort) 76 reorganization, of a file 6, 8 RETAIN parameter 64 RUN statement 61 RECORDS parameter 64 scratch files, definition 64	tag-along sort 71 OCL 76 output records dropping control fields 75 retaining control fields 71 temporary files, definition 64 TRACKS parameter 62 transaction file, definition 7
selection, of file organization 6 sequence specification sheet 73 sequential by key processing 2 sequential file organization advantages 1 creation 4, 22 definition 1 maintenance adding records 5, 23 deleting records 6 reorganizing 6 updating 6, 34 ordered 2 processing 1 unordered 2 sequential processing 2 by key 2 within limits 3	UDATE 68 UDAY 68 UMONTH 68 unblocked records 14 UNIT parameter 62 unordered file organization 2 updating records by 5471 Printer-Keyboard 39 definition 3, 6 in an indexed file in a sequential file 34 all records 34 randomly by key 35 some records 35 use, of a file 7 UYEAR 68
sequential within limits processing 3 size, of a file 9	volatility, definition 8
Sort Program (see Disk Sort program) sorting, a master file 7	work areas (Disk Sort program)
statement identifiers 62	5471 Printer-Keyboard as a second printer 51 for updating records 39

READER'S COMMENT FORM

IBM System/3 Disk System Concepts and Programming, Programmer's Guide Form GC21-7503-0

Your answers to the questions on this sheet will help us produce better manuals for your use. If any of your answers require comments, or if you have additional information you think would be helpful, please use the space provided. All comments and suggestions become the property of IBM.

			
		Yes	No
1.	Is the manual easy to read?		
2.	Is any of the information unclear?		
3.	Is additional information needed?		
4.	Is any of the information unnecessary?		
5.	Did you read the Preface?		
6.	Did you use the Table of Contents?		
7.	Did you use the Index? *		
8.	Did you take the tests? *		<u> </u>
9.	How did you use the manual:	* Not inc	cluded in
	Instructor in a class Student in a class Reference material Self-Training Other (Explain)		
Have you h	nad previous computer or programming training?		
What is you	ur present job?		
What busin	ness is your company engaged in?		
COMMEN.	TS		

• Thank you for your cooperation. No postage necessary if mailed in the U.S.A.



IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601
(USA only)

IBM World Trade Corporation 821 United Nations Plaza, New York, New York 10017 (International)