SR30-0017-1

GENERAL
SYSTEMS
DIVISION
EDUCATION

STUDENT TEXT

System/32 RPG II Programing

Second Edition (January 1976)

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Address comments concerning this publication to:
IBM Corporation, General Systems Division
Education Development, Dept. 71G
P.O. Box 2150
Atlanta, Georgia 30301

Comments and suggestions become the property of IBM.

IBM SYSTEM/32 RPG II PROGRAMMING

TABLE OF CONTENTS

INTRODUCTION

This self study book was written to accommodate two audiences. First, for the person who has no programming experience. Second, for the experienced programmer who has not programmed in the RPG II language.

The book is comprised of an optional topic, "Fundamentals of Programming" and nine chapters that teach you to use the IBM System/32 RPG II language to solve data processing problems.

The total time needed to study this course is about 75 hours on the average excluding the optional topic. If you need more time, that's all right. If you take less time and understand what you are studying, that's fine too. You should set aside periods of about 1 to 2 hours for study each time and should try to study in a location that is relatively quiet.

Study at your own rate. Take time to re-read information and look carefully at each illustration as you progress. Do not expect to remember everything you read. You may refer to books for information when you are on the job, and it should be the same when you study. There is one exception. Some chapters include Self Test questions which you may be asked to try to answer from memory after completing them.

A self study course presents information, shows examples and asks you to do things. You will get the most from this course when you actively participate. To do so you will need a pencil, some scratch paper, a copy of the IBM System/32 RPG II Language Reference Manual, some blank RPG II coding forms, and some blank print charts. Here's a list for reference.

Form Number	Name
SC21-7595	System/32 RPG II Language
GX21-9092	Control and File Description Specifications
GX21-9091	Extension and Line Counter Specifications
GX21-9094	Input Specifications
GX21-9093	Calculation Specifications
GX21-9090	Output Specifications
GX20-1816	Print Chart

To prepare for attendance in the IBM System/32 RPG II Workshop class, you are expected to complete chapters 1-6 of this text. Chapters 7-9 contain information about additional facilities of the RPG II language.

At this time, take the "Pre-test" to determine what you need to learn about programming fundamentals before studying about System/32 RPG II programming.

SYSTEM/32 RPG II PROGRAMMING

PRE-TEST: Write your answers to these 18 questions on scratch paper. When you finish, check your answers against those shown on the following pages.

1. What is a compiler program?

2. What is the difference between a source program and an object program?

3. Which devices on the System/32 may be used as...

   a. input devices?

   b. output devices?

4. How can disk data on the System/32 be saved for future reference?

5. Disk file records on the System/32 may be organized in three different ways. What are they?

6. What is meant by "relative record number" when dealing with disk files?

7. What is a "key field" in a disk file record?

8. Describe each of the three disk file organization methods in general terms.

9. Disk file records on the System/32 may be processed in three major ways. What are they?

10. Explain the difference between matching records and chaining.

11. Describe each of the three major disk file processing methods in general terms.

12. What is meant by the expression, "define the problem"?

13. Describe a "branch point" as it would appear on a flowchart.

14. What is a control group?

15. What is a control field?

16. What is meant by "test data"?

17. How is "test data" used by programmers?

18. What is the difference between an input file and an update file?

SYSTEM/32 RPG II PROGRAMMING, PRE-TEST: ANSWERS

A. If all of your answers are correct, bypass the optional topic and begin your study of System/32 RPG II Programming, Chapter 1.

B. If you had a few errors, you may wish to read portions of the optional topic that deal with those points. Page numbers are shown in parentheses in front of each answer so that you can quickly reference the study areas.

1. (4) A compiler program is one that is used to analyze and translate source language program statements into an object program. Compiler programs are made available to users of coding languages.

2. (4) A source program is the set of instructions written by a programmer using a coding language. Source programs cannot be run by the computer until they are translated by the compiler program for that language. An object program is usable directly by the computer to run a job.

3. (6)

System/32 Device	May be used as...
Console Keyboard	Input
Disk	Input or Output
Printer	Output
Display Screen	Output
Diskette	...not used directly...

4. (14) Disk data may be saved for future reference by copying it onto diskettes at designated points in time. The saved data can be reloaded from diskette as needed to establish a "restart" status. This copying and reloading is done by an operator using special System/32 commands, not with RPG II programs.

5. (17) Three types of disk file organization are sequential, indexed and direct.

6. (25) A relative record number refers to the location of a disk record according to its position relative to the first record in the file. Relative record numbers apply to disk files organized as either sequential or direct.

7. (19) A key field in a disk record is the field that contains data used to sequence records when the file is created, and to search for specific records when the file is being processed. On System/32 the term "key field" applies only to indexed files.

8. (23) The sequential method is used to store records in the order that they are entered in adjacent positions on the disk.

   (24) The indexed method is used to store records in the order entered in adjacent positions on the disk also. In addition, an index is automatically created for the file, recording the record's location in the file, and placed just in front of the record set. The index is always arranged in ascending sequence by key field value.

   (20,25) The direct method is used to store records in positions calculated from a specified formula. The sequence of input records has nothing to do with their placement within the space reserved for the file. Individual records may be retrieved directly through the use of the same formula that calculated their placement positions during the loading process.

9. (28) Three major ways of processing disk files are consecutive, sequential and random.

10. (42-44) Matching records involves the sequential processing of records in at least two files. One file is designated as primary while all others are called secondary. All files must be in the same sequence.

    (38-41) Chaining involves the searching of a file for a particular record. Two files are involved: one that is the searching file and another that is the chained file. Records in these files do not need to be in the same sequence.

11. (28) The consecutive processing method may be used to process records in files organized as either sequential or direct files. Records are processed one after the other in the order they appear within the file.

(28) The sequential processing method is used to process records in files organized as indexed disk files. Processing may be done for either all records in the file or for some records in the file. In each instance, records are processed in ascending key sequence (the index is in ascending record key sequence).

(28) The random processing method is used to process individual records in any System/32 disk file...

To randomly process records in an indexed file, reference is made to key field data and the index.

To randomly process records in either a sequential or direct file, reference is made to relative record number.

In each case of random processing, the "chaining" method is used to locate individual records.

12. (46) "Define the problem" means that you must identify what is desired as output, what is available as input, and what processing is needed in addition to available input to produce the required output.

13. (48,50) A branch point on a flowchart is a place where a decision needs to be made so that one of two alternative paths may be followed.

14. (34) A control group is a set of input data records that have identical data in certain fields used for group identification.

15. (34) A control field is the field in which identical data, for the purpose of grouping records, is recorded.

16. (51) Test data is a set of made-up or available records that can be used to test a flowchart or a program to determine its accuracy.

17. (51) Test data is used by programmers in two ways.

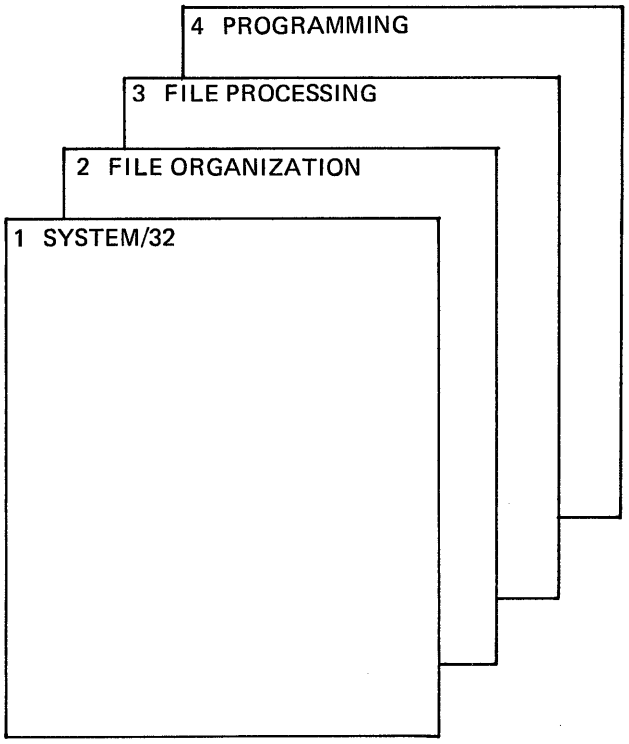
First, to assure that the procedure or flowchart designed to solve a particular problem is accurate.

Second, to assure that the compiled object program derived from that procedure or flowchart runs correctly.

18. (15,58) An input file only provides data for processing. An update file functions as both an input and an output file, in that it provides data for processing and then accepts and stores processed or new data in place of the old.

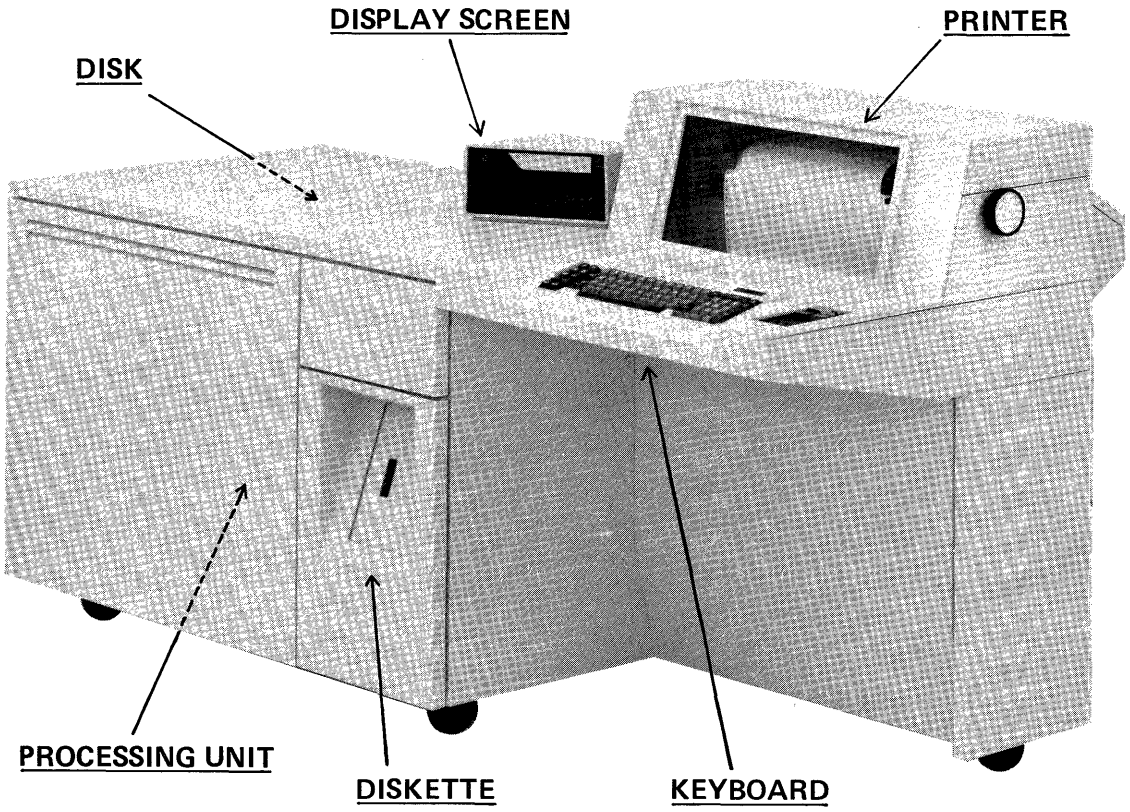
OPTIONAL TOPIC: Fundamentals of Programming

This topic is divided into four sections. The first presents information about the IBM System/32 computer. The second presents information about the various types of files that may be stored on the System/32 and how they may be organized. The third deals with the methods used in processing files of records that are stored on the system. The fourth provides information about programming as a way of handling data processing problems on the computer.



The IBM System/32 Computer

The IBM System/32 computer is an operator-controlled data processing system featuring keyboard data entry and disk processing. Familiarize yourself with its components.



Keyboard: used by operator to initiate job activity, enter data, respond to messages and halt activity

Printer: produces printed reports and messages

Display Screen: "prompts" operator when action is required, may also be used to display information

Disk: stores instructions (called programs) and files of data

Processing Unit: directs and controls all processing after operator initiates activity; also contains arithmetic circuitry, logic circuitry and processor storage

Diskette: provides for recording data from the disk as "backup" to protect against loss of programs and data due to accident.

When a job is to be run on the System/32, the operator does these steps:

1. readies the system by turning on the power, placing paper in the printer, entering the data through the keyboard and looking at the display screen for messages,

2. enters the information needed to locate and store in the processing unit (load) the proper program on the disk and identifies any data files on the disk that will be needed to do the job,

3. enters keyed data if required during the job,

4. responds to messages as they are displayed,

5. terminates the job when necessary,

6. removes printed reports and/or printed messages, and

7. records "backup" information on a diskette if required, and turns off the system.

Naturally, power is not turned off at the end of one job if another is ready to be started. To insure accuracy in operator activity, a "run sheet" is prepared for each job in advance in order to describe exactly what programs, files and responses to predictable messages (record out of sequence, for example) must be used.

The IBM System/32 may be used to do a variety of data processing jobs.

1. produce business reports

2. create files of disk data and store them

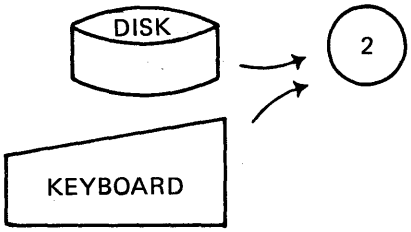
3. sort records in a data file prior to using them in a job

4. create programs and store them on disk

The kinds of jobs shown in steps 1, 2 and 3 are done by the operator as described earlier. The job in step 4, "create programs and store them on disk", involves a process known as "compilation". In this kind of a job, the program to be located on the disk is the RPG II compiler, and the data to be processed is called the source program. Your job as programmer involves the coding of source program statements on special RPG II language specification sheets. After compilation, the source program has been translated into a machine-readable program called an object program. This object program is stored on the disk for use as needed to run a job in the future.

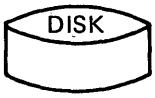
RUNNING A JOB

locate the program for the job, and

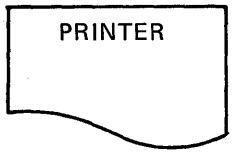


use data from disk files and/or keyed data, and

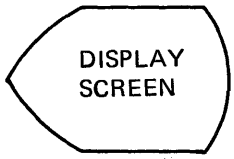
produce new information on



or



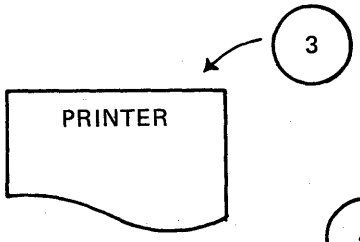
or the



COMPILING AN RPG II SOURCE PROGRAM

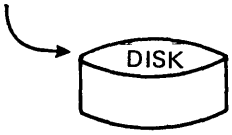
locate the program for the job; this program is the RPG II compiler;

use the source program already stored on the disk as data;



produce a list of the source program statements, and

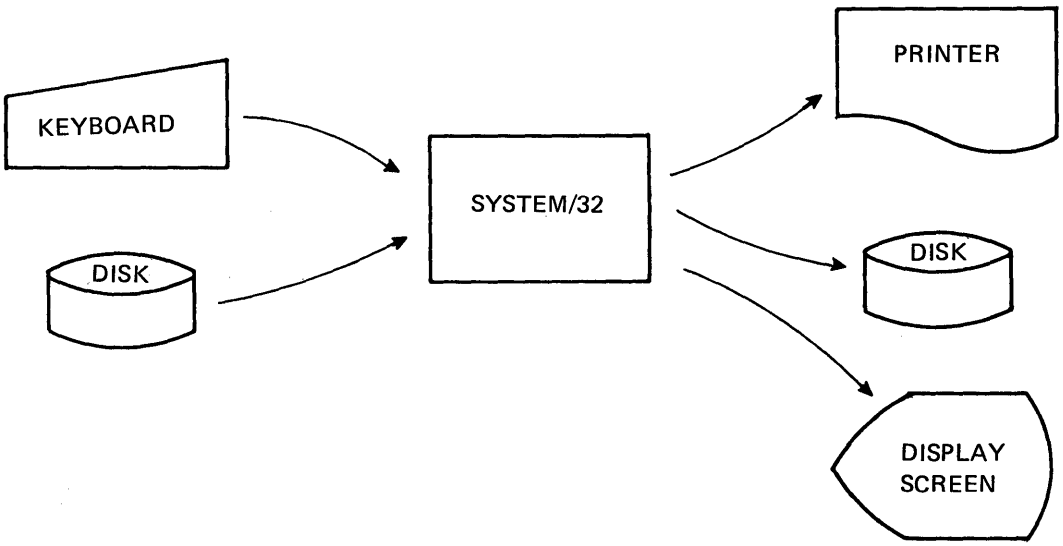
store the compiled object program on the disk for future use.



As you can see, running a job that compiles a program is no different to the operator than running any other job. We will be studying more about compiling programs from the programmer's point of view during this course.

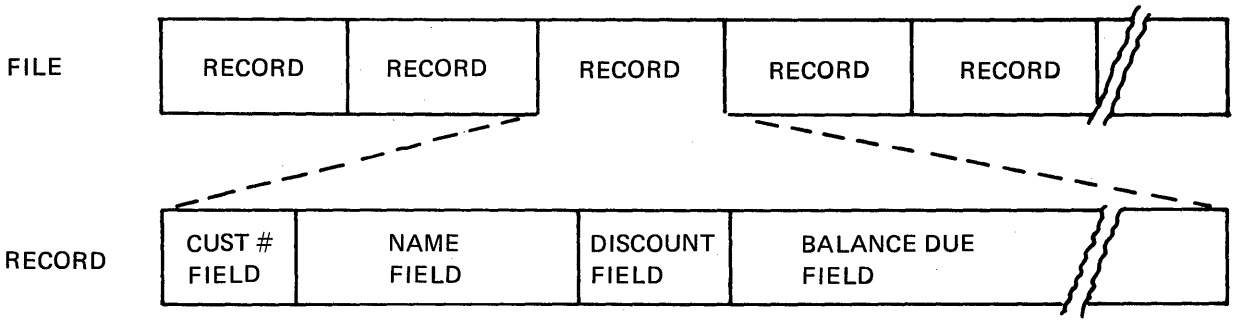
Any computing system processes input data in order to create output data. The System/32 devices are used in this manner.

Input data from ... Output data to ...



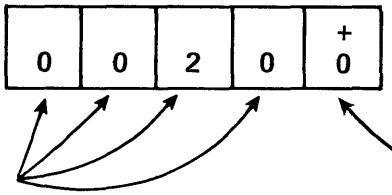
An input data file provides information to the computer for processing. The information is "read" by the computer one record at a time. This record is temporarily stored in the processing unit while all required calculations and checking of codes or re-arrangement of its fields takes place.

Processing is done on fields of data.



Some fields have numeric information while others contain what is known as alphameric information. Numeric fields contain digits in each position. A sign (+ or -) is included in the rightmost position of each numeric field. Zeros are used to fill all unused spaces in a numeric field.

NUMERIC FIELD



DIGITS ONLY

A DIGIT AND A SIGN

EXAMPLES

+
00200
–
00006
+
76543

Alphameric fields may contain any kind of character, including letters, numbers, special characters and blanks. Each character uses one space.

ALPHAMERIC FIELD

S	E	P	T	E	M	B	E	R

Examples

S E P T E M B E R

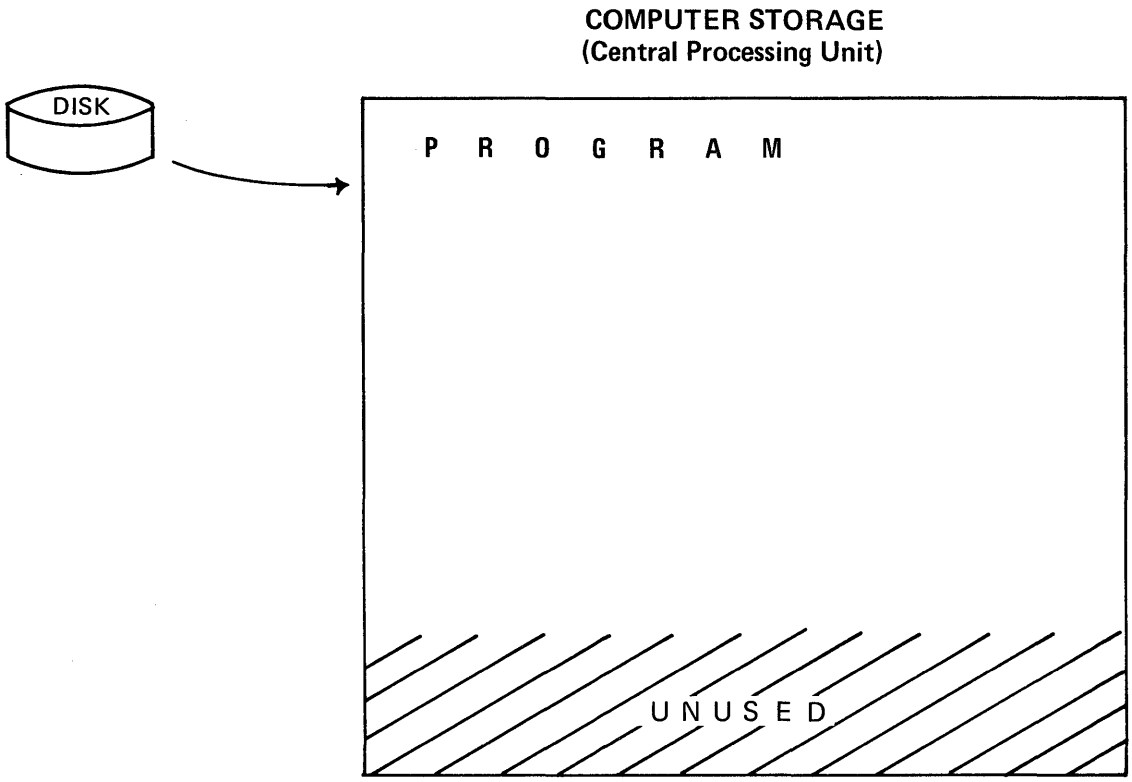
P L U G , 3 / 8

3 2 M A I N S T

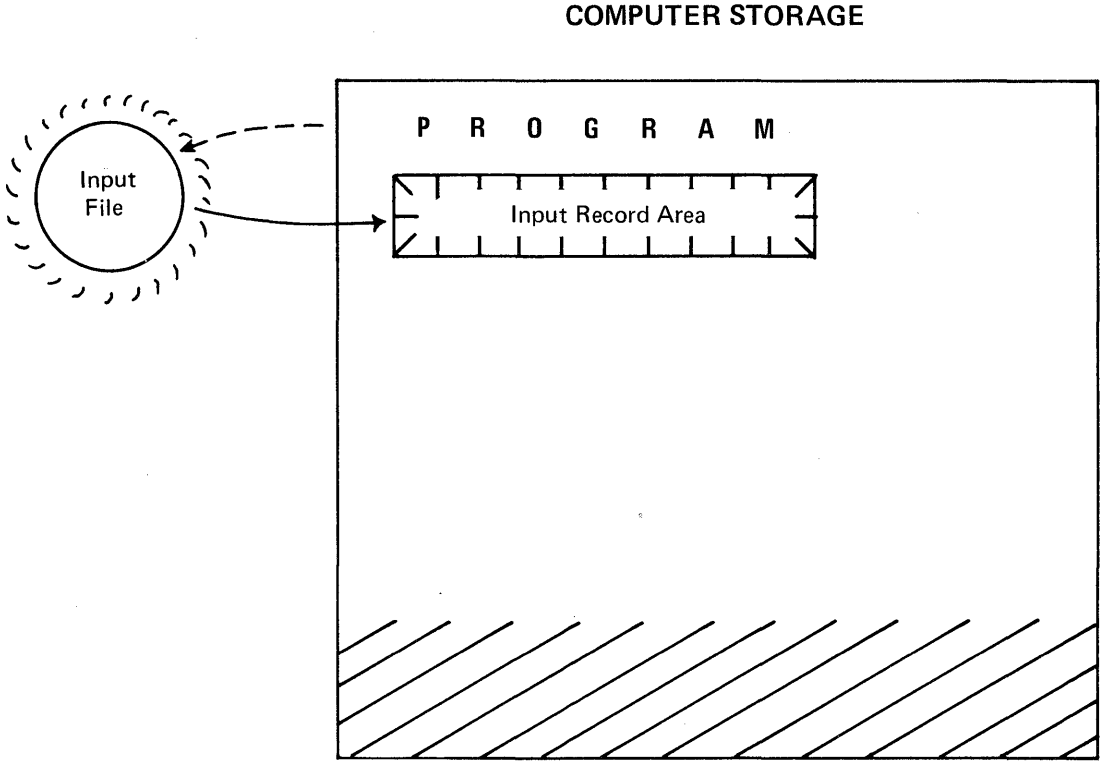
In certain applications it is useful to have an alphameric field that is completely blank, that is, filled with blanks in every position. Also, an alphameric field might be filled with numbers in every position.

The size of a field may range from 1 position to 15 positions for numeric fields and from 1 to 256 positions for alphameric fields when using RPG II.

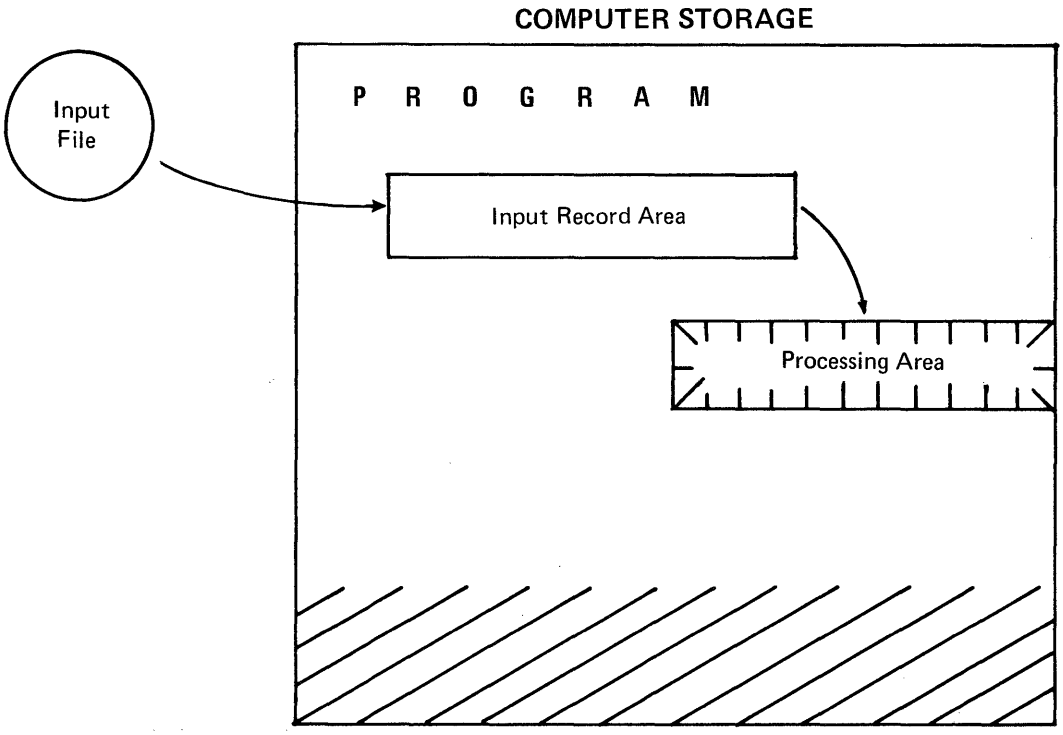
As we said earlier, the operator enters information needed to locate and store the selected program in the computer so that a job can be started.



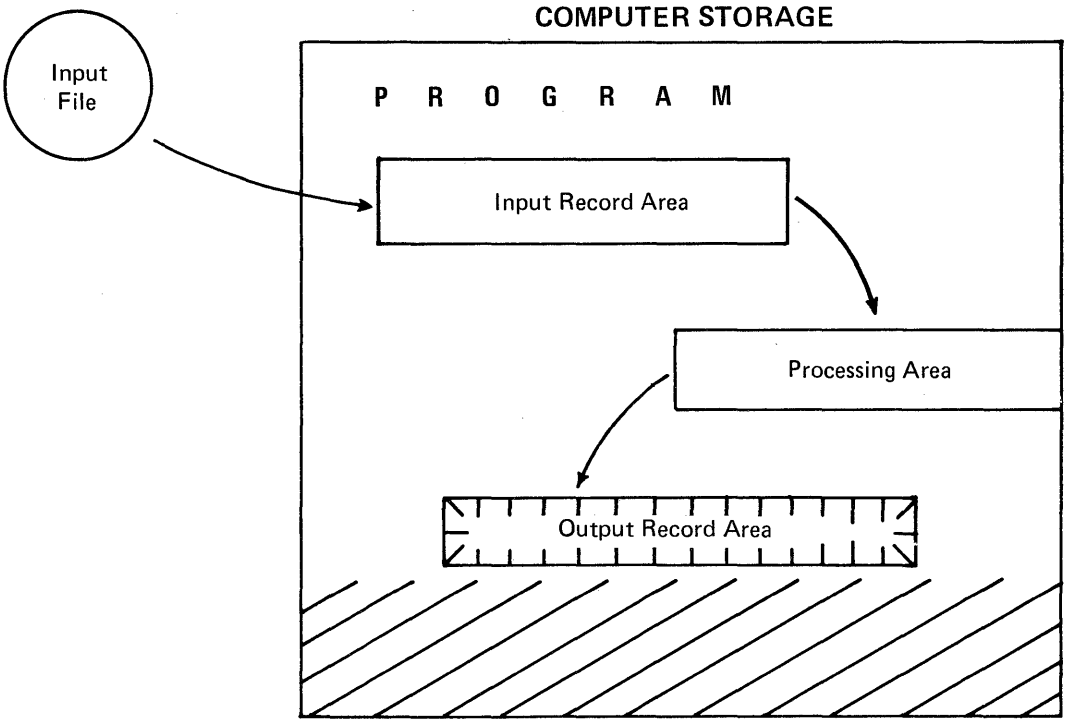
Next, the program reads one input record into an Input Record Area of computer storage.



After a record is read into the computer's storage area, instructions for processing one or more fields of its data are activated by the program.

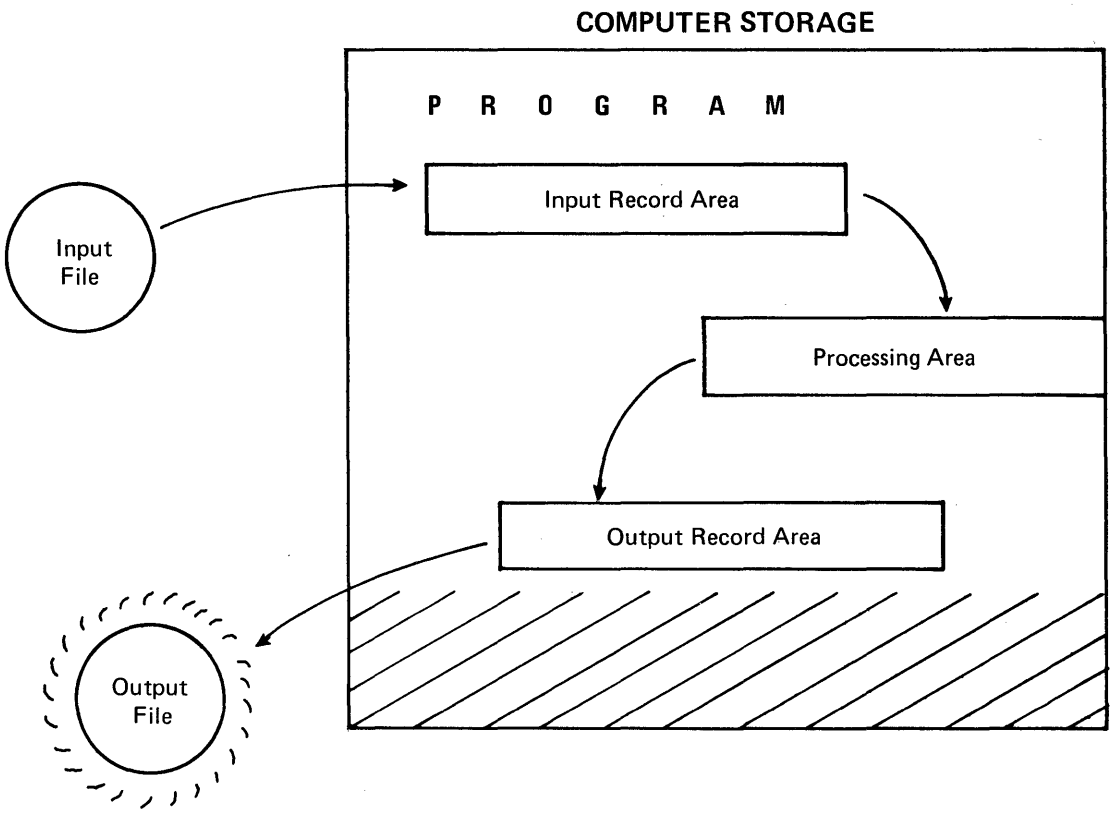


After all fields have been processed, an output record is "built". Instructions in the program direct the putting together of the output record.



At this point in time, the computer contains a program, one input record, a set of processed fields and one output record. What should the program do next?

The next step is to "write" one record onto an output file. Just as input files are processed one record at a time, output files are created one record at a time. The process is repeated for each input file record until all have been processed and their related output file records have been created.



As you might have deduced, as a new input record is read into the Input Record area, the old record is completely replaced by the new one. However, the contents in the processing area and output record area are not changed as yet.

When processing takes place, individual fields of processed data replace the old ones. Likewise, when the new output record is built, the old one in computer storage is lost.

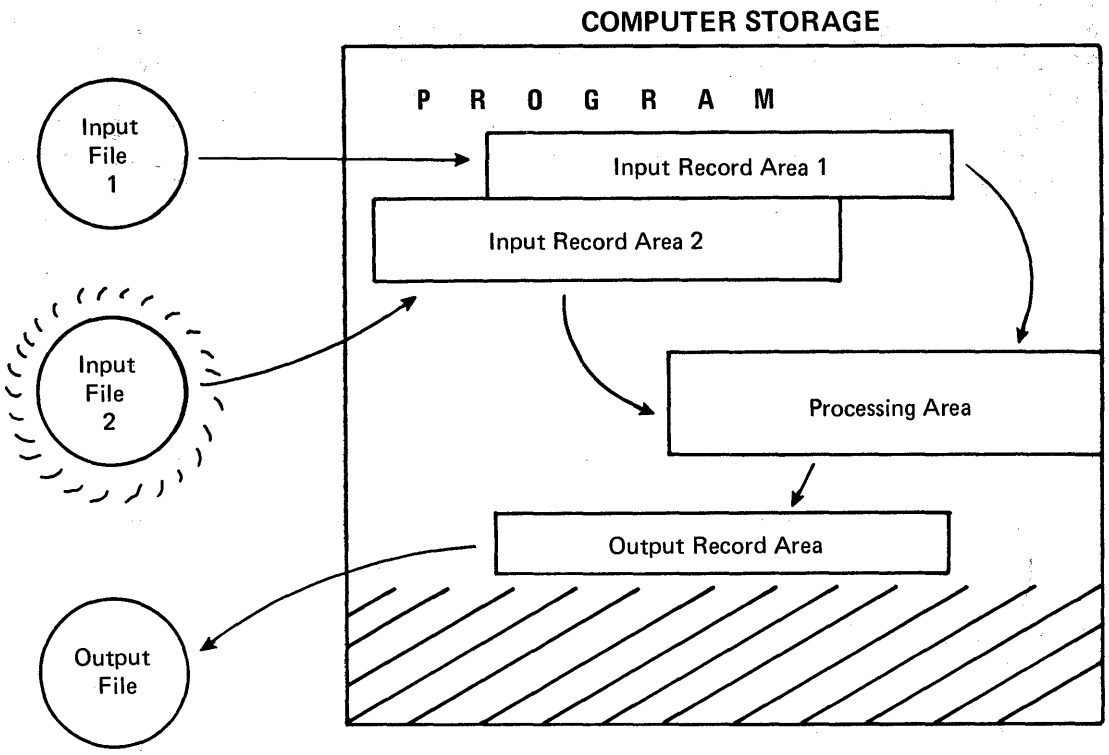
What about the Output File record that was written out previously? Is it replaced by the next one being written?

No, each new output record is recorded in the next available space in the output file. Keep in mind that an entire file is being created by the process just described.

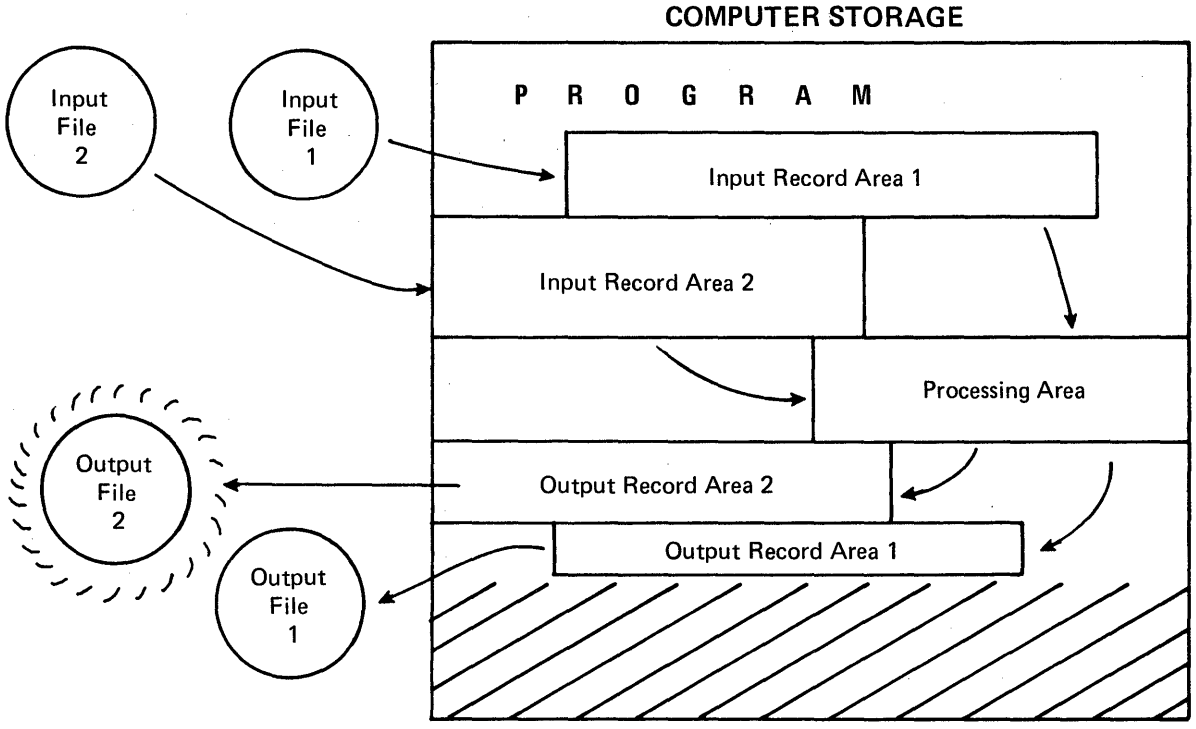
Earlier we said that an input record is temporarily stored in the computer so that processing of its fields may take place and then an output record can be built prior to its being written as an output file record.

The technique of using computer storage as temporary storage permits the processing of large files of input data in a relatively small space.

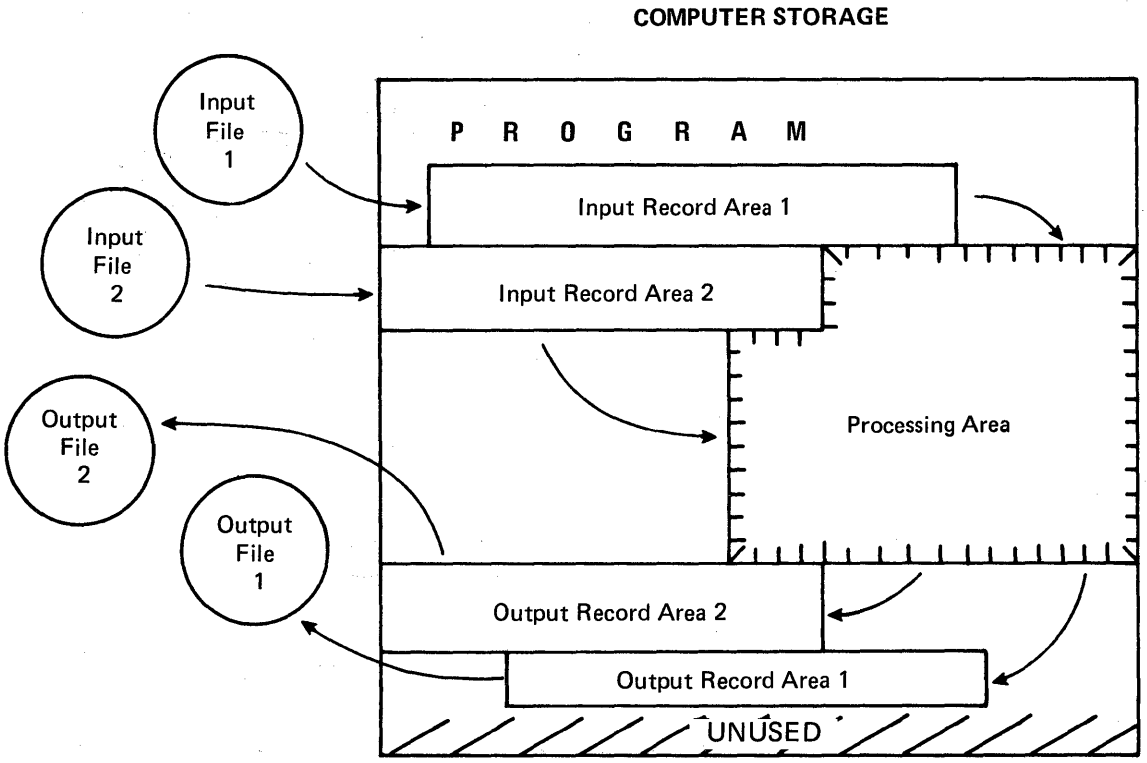
The System/32 is not limited to processing records from one input file in order to create one output file. The next illustration shows the use of two input files as another possibility.



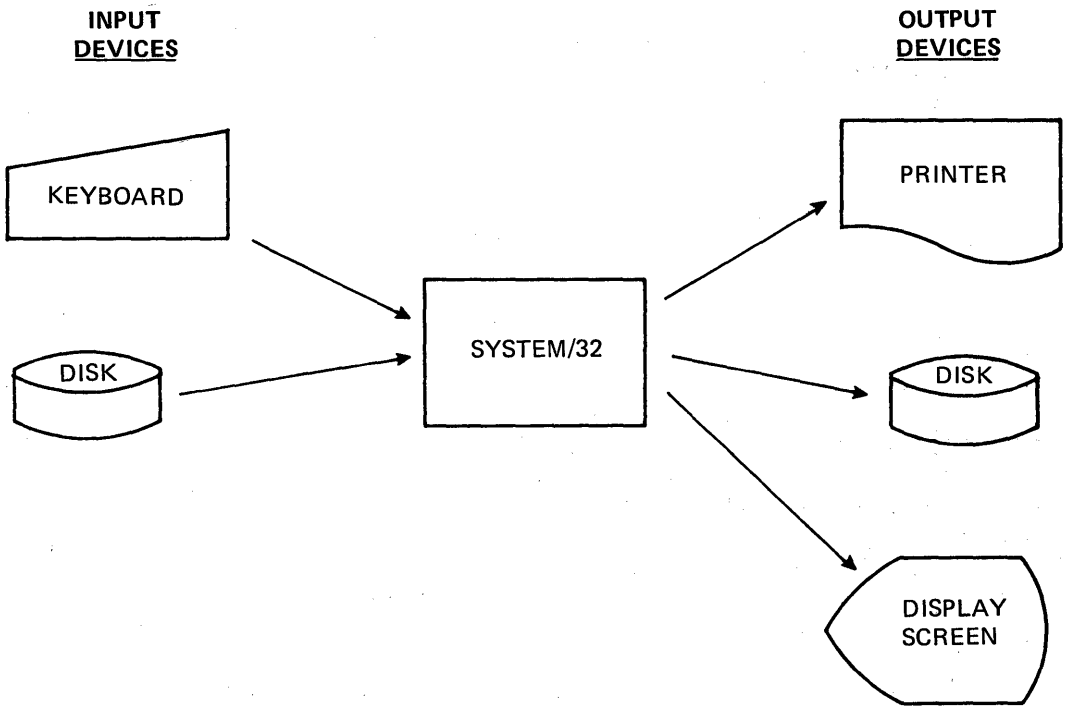
And that's not the limit either. Look at this.



To be honest, we need to point out that a program that can direct and control the processing of records from two input files probably takes more space than a program that directs and controls processing records from one file. Also, we probably need more processing area in order to handle more fields of data. Here's a more reasonable illustration of that last example.



When you code solutions in the RPG II language, you need to know which devices on the System/32 can read input file records and which devices can create (write) output file records.

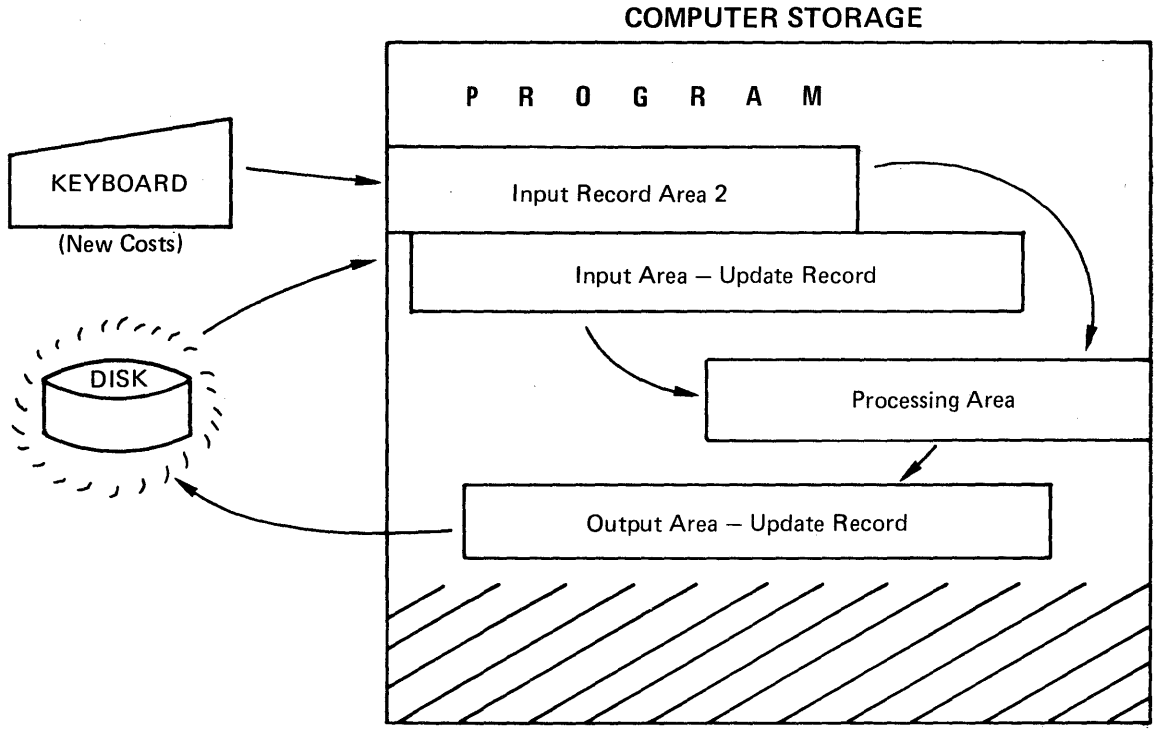


Even though a diskette may be used to copy information from a disk for backup protection, the diskette is not used as either an input nor an output device for System/32 RPG II programs.

The display screen holds up to 6 lines of 40 characters as a temporary display. It is used to display messages as needed to direct the operator. These messages are like a very small output file.

The disk is a device that may be used as either an input or an output device. In fact, there are many applications in which it serves both functions during the same job. When used in this manner, a disk file is called an "update" file. After an update file record is processed the new output record replaces the original disk record. Here's an example.

"Update the cost figures in the item file on disk. The new cost figures are keyed in from the console keyboard".



An update file is really used both as an input and an output file.

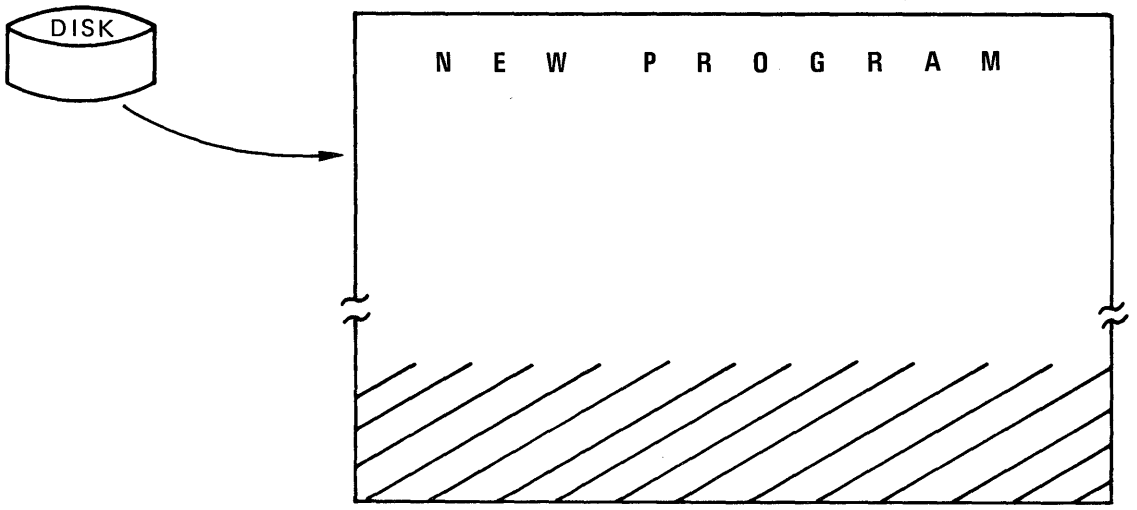
We have looked at the IBM System/32 as a computing system that is capable of processing records from one or more input files, in order to produce or update records in one or more output files.

Computer storage is temporary storage. It is used to store a program, provide space for input records, provide space for processing fields within those records, and provide space for building output records. The amount of computer storage needed to do these things depends upon the number of files to be processed, the length of their records and the complexity of the processing steps used in the program.

After all input file records for one job have been processed, the program terminates. A message is displayed for the operator indicating that the job ended, at which time information may be removed from the printer.

A look at the run sheet or run book directs the operator to load the next program. When this is done, the first program is replaced by a new program.

COMPUTER STORAGE



The first record is read by the new program and the entire cycle begins again.

As you can see, the operator readies the system, loads the program and keys in information as needed. Your job as programmer requires you to write programs that do each job accurately.

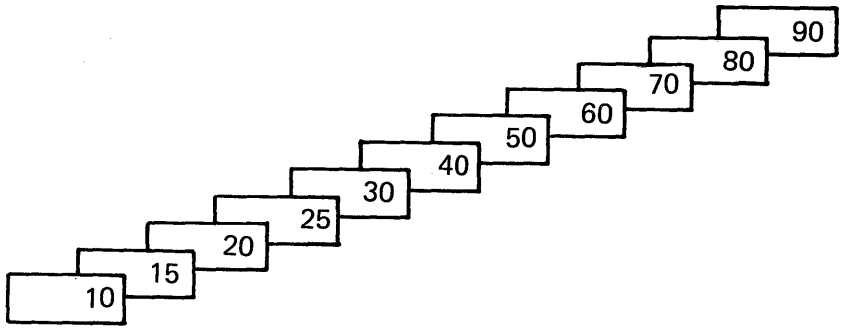
File Organization

File organization is the arrangement of records in a file. Three types of file organization are used on the System/32 and each may be described in the RPG II language. When a file is created it must be organized as either:

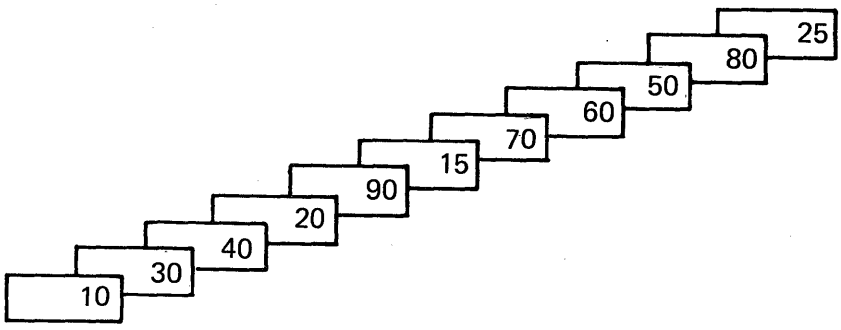
1. a sequential file,
2. an indexed file, or
3. a direct file.

1. A sequential file is one in which the records are stored in the order they are read. Here are two examples.

EXAMPLE 1



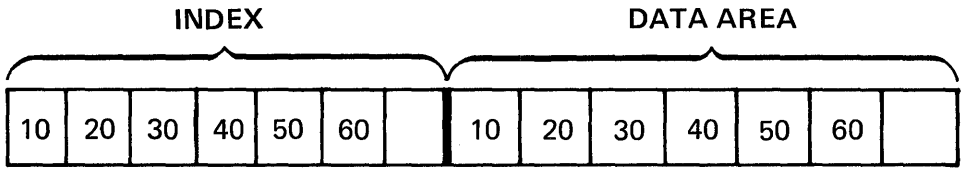
EXAMPLE 2



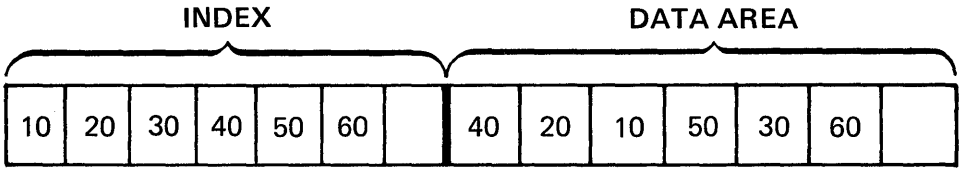
2. An indexed file is a disk file in which the location of records is stored in a separate portion of the file known as the index.

   Here are two examples.

EXAMPLE 1



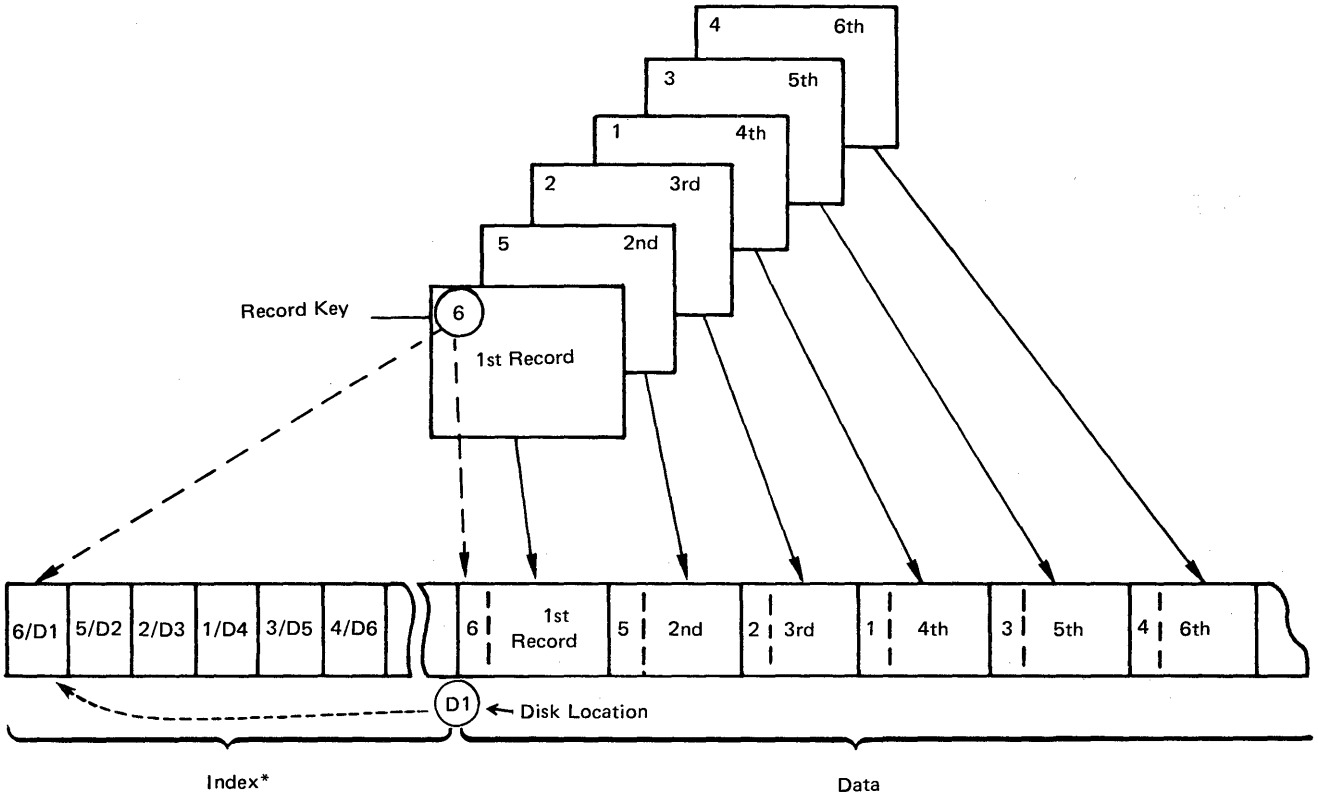
EXAMPLE 2



In the second example, the records were loaded "unordered" after which the index values were automatically sorted by the system and stored in ascending sequence.

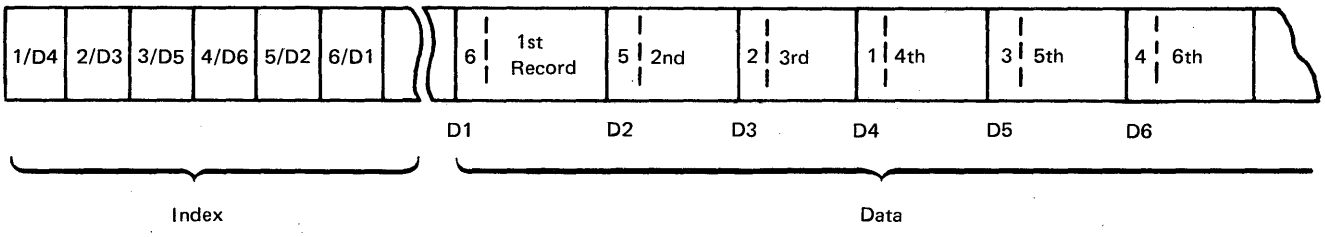
Index entries are always placed into ascending sequence (either numeric or alphabetic) by the system after the indexed file data records have all been recorded on disk.

Look at a more complete example of indexed file organization. The index actually contains both the record key and the disk location of every record in the indexed file. A record key is a field of data that identifies a particular record, such as customer number in a customer record.



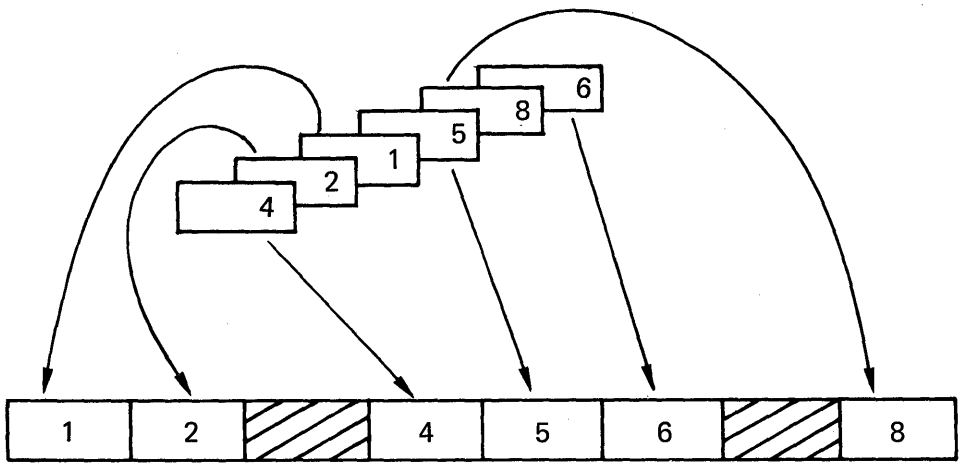
*Entries are of the form record-key/disk-location (D1 = 1st disk location, D2 = 2nd disk location, and so on)

After the last entry is made in the index, the index entries are sorted into ascending record key sequence.

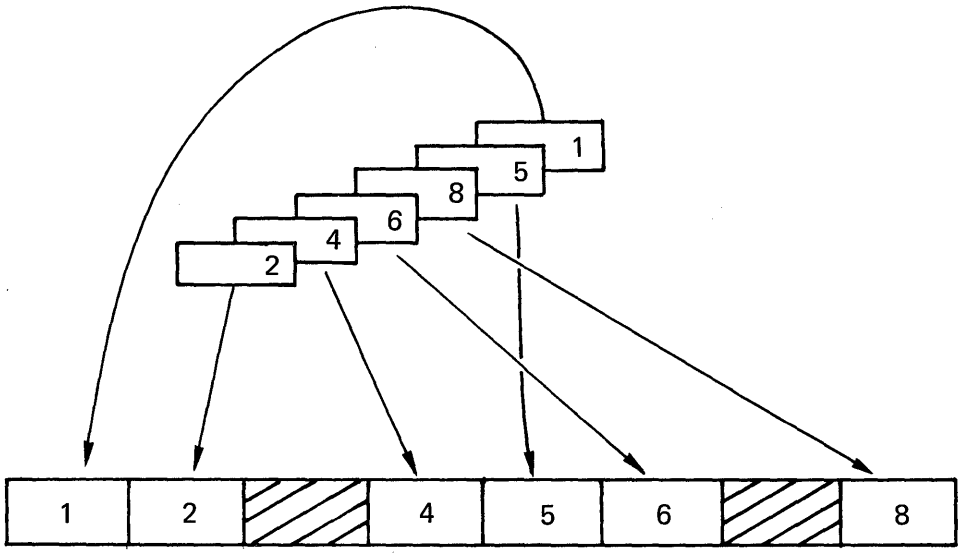


Notice that the order of the data records is not changed, only the index entries are sorted.

3. A direct file is a disk file in which records are assigned specific record positions within the area reserved for storing the file. Relative record numbers identify the relative position of each record within the file.



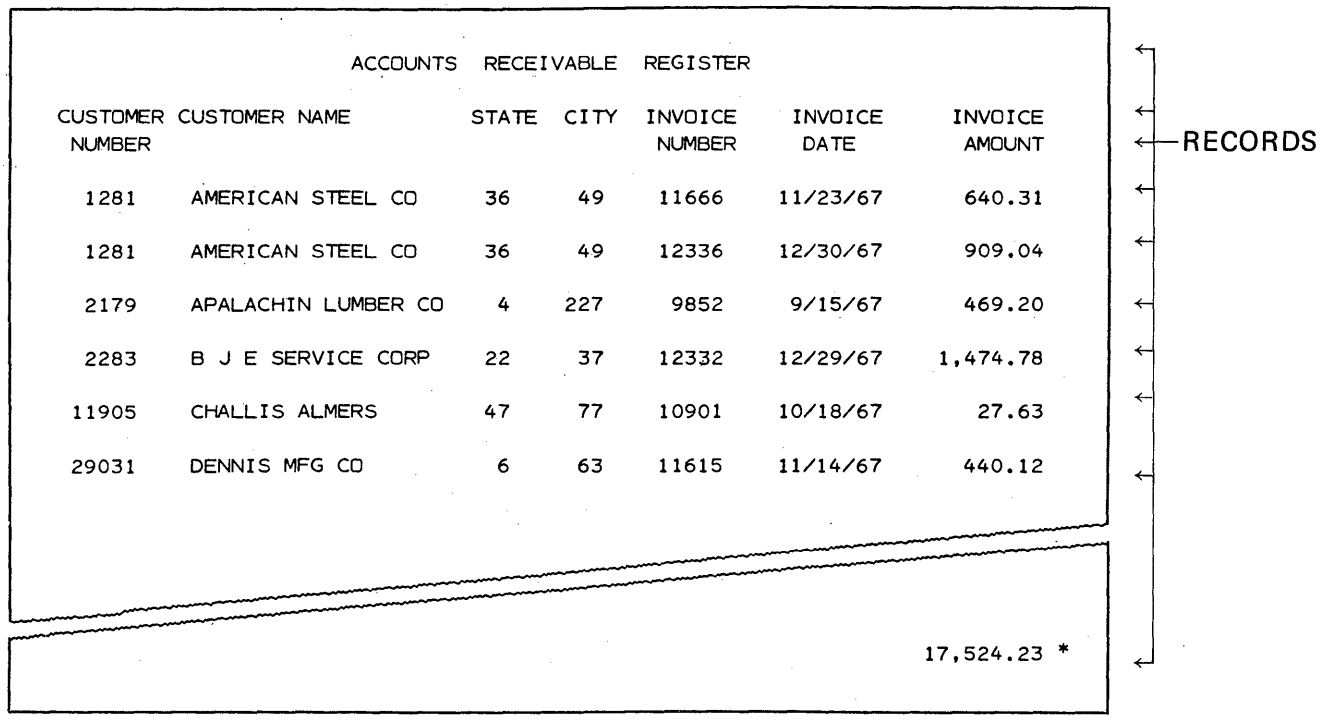
Regardless of the order in which these records are arranged prior to being put into the file, they always occupy their assigned (relative record number) positions within the file.



In these two examples, spaces are reserved for two records (relative record numbers 3 and 7) that may be added to the file at a later date.

Indexed files and direct files are always stored on disk. Sequential files may be stored on disk or some other storage medium. On the System/32 we have two examples of sequential files that are stored on a medium other than disk.

Printed Reports! Each printed line is a record in the file. The entire report is the file.



ACCOUNTS RECEIVABLE REGISTER

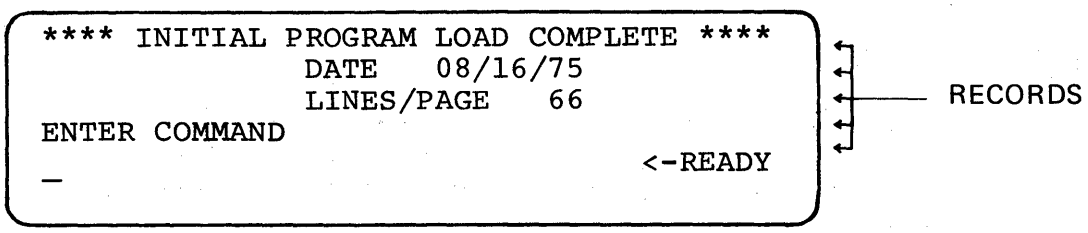
CUSTOMER NUMBER	CUSTOMER NAME	STATE	CITY	INVOICE NUMBER	INVOICE DATE	INVOICE AMOUNT
1281	AMERICAN STEEL CO	36	49	11666	11/23/67	640.31
1281	AMERICAN STEEL CO	36	49	12336	12/30/67	909.04
2179	APALACHIN LUMBER CO	4	227	9852	9/15/67	469.20
2283	B J E SERVICE CORP	22	37	12332	12/29/67	1,474.78
11905	CHALLIS ALMERS	47	77	10901	10/18/67	27.63
29031	DENNIS MFG CO	6	63	11615	11/14/67	440.12

17,524.23 *

As we stated earlier, "a sequential file is one in which the records are stored in the order they are read". The report title, "Accounts Receivable Register", is the first record in this sequential file. The second record in the file is the line that contains the top word for each columnar heading (from left to right), "Customer Customer Name State City Invoice Invoice Invoice". What is the third record? What is the last record?

The third record is the line that contains the bottom word for each columnar heading. In this example, the third record contains, "Number Number Date Amount". The very last record in this sequential file is the total amount printed at the bottom of the page 17,524.23 *.

The other medium that can store a sequential file is the display screen. Each displayed line represents a record.



Here's a chart that relates System/32 output devices to types of file organization.

OUTPUT DEVICE	ORGANIZATION OF FILE(S)		
	SEQUENTIAL	INDEXED	DIRECT
DISK	X	X	X
PRINTER	X		
DISPLAY SCREEN	X		

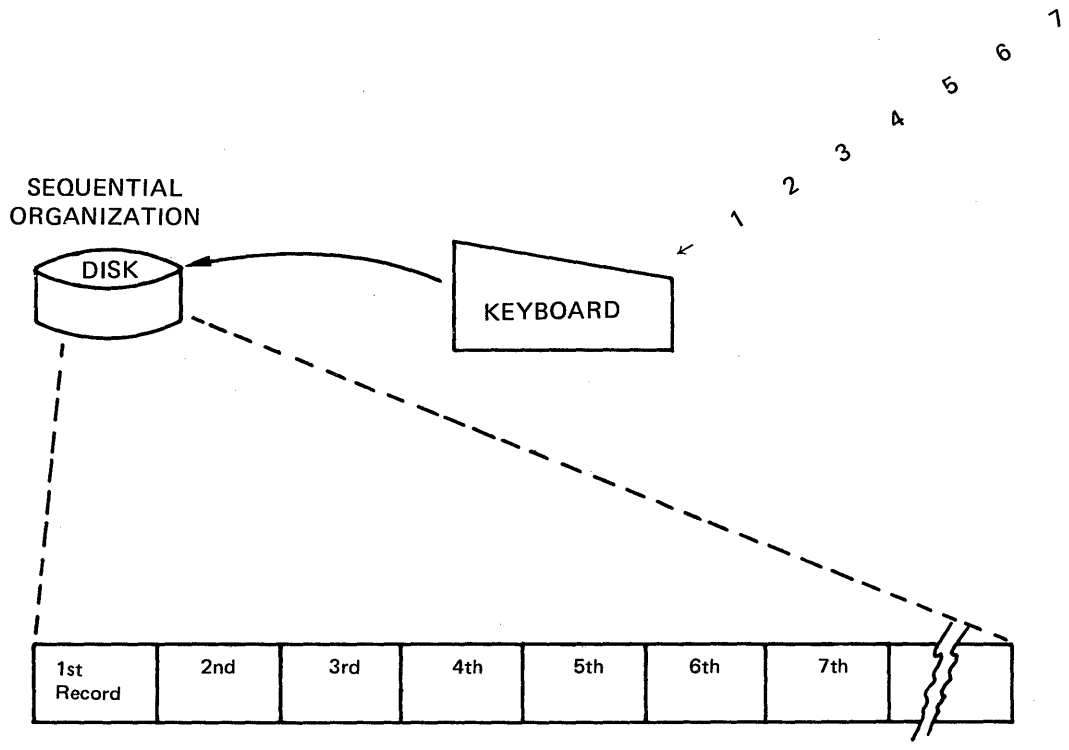
Disk file organization is planned prior to the coding of RPG II programs that create the files. As a programmer you need to know that there are three types of disk file organization. You will be taught to describe (code) disk files later on in the course. The characteristics of each organization method follows.

SEQUENTIAL ORGANIZATION

1. Records are stored in the order they are loaded.

2. Normally, they are sorted for convenience in processing, but sorting is not required.

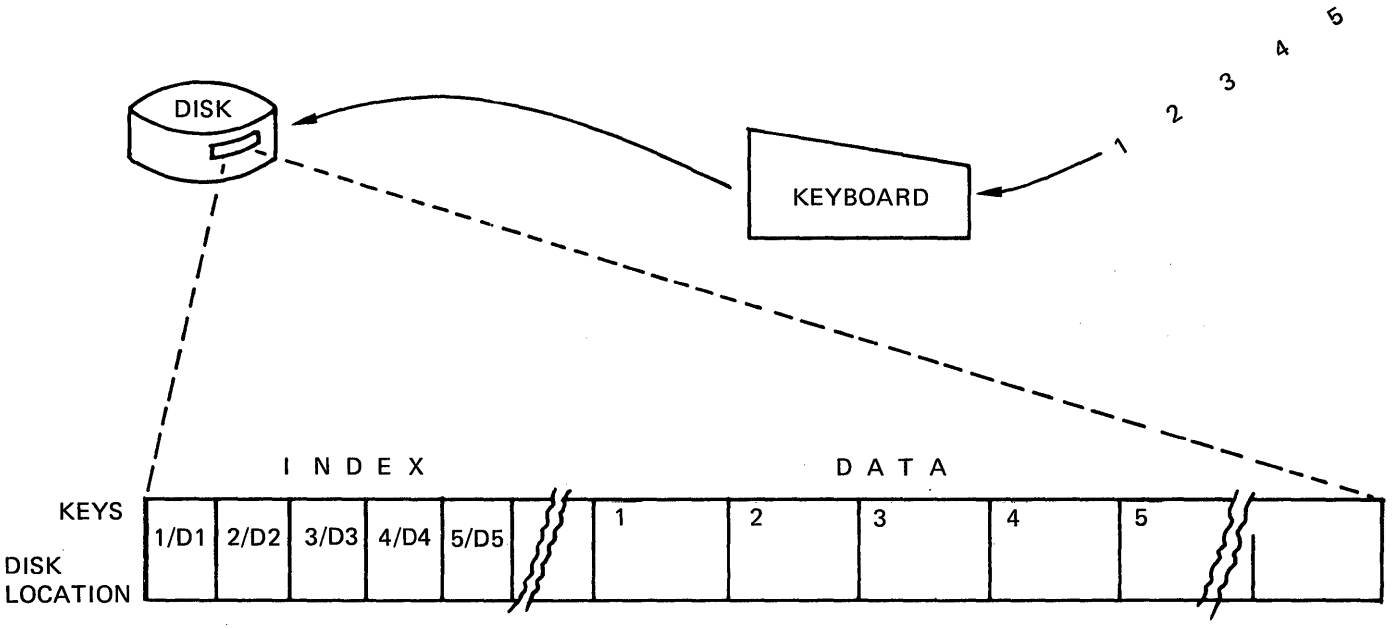
3. No spaces appear between records, but space may be left at the tail end of the file so that new records may be added at some later time.



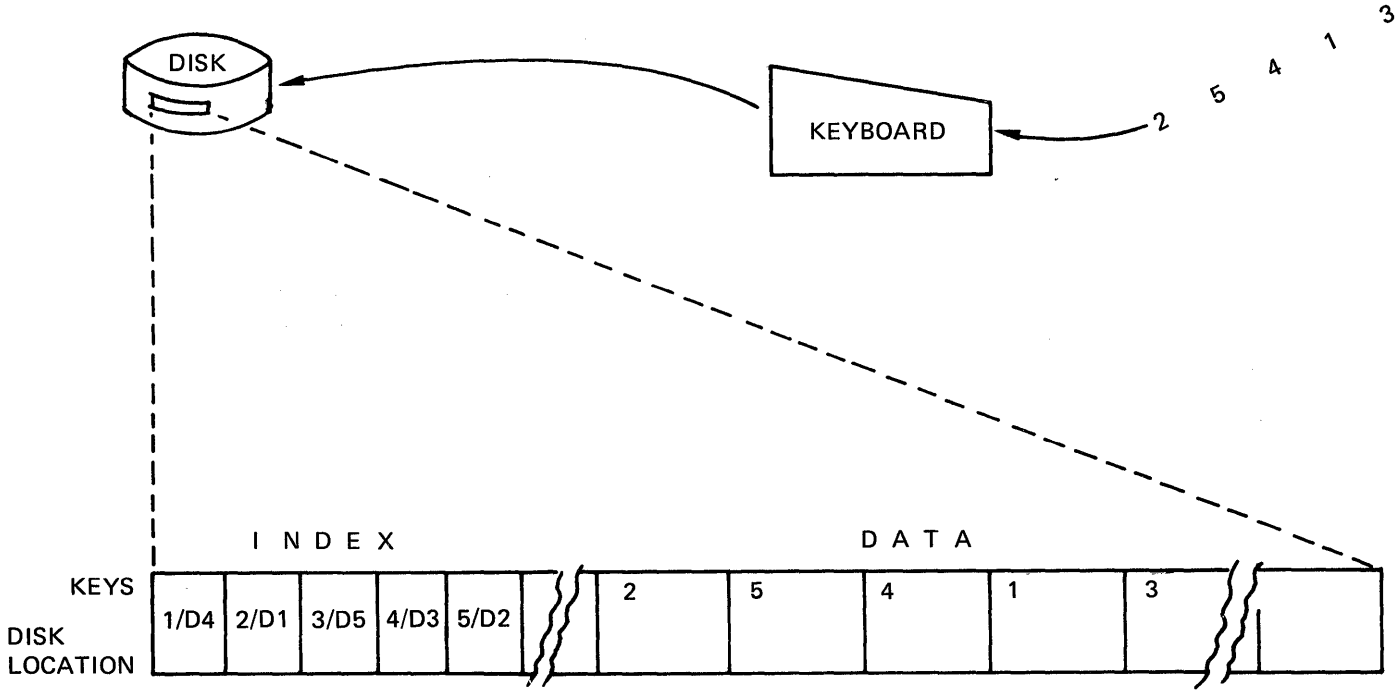
INDEXED ORGANIZATION

1. Data records are stored in the order they are loaded.
2. Index entries are stored in ascending key sequence. Automatic sorting of index entries follows an unordered load.
3. No spaces appear between data records or between index entries.
4. Space may follow the index and the data records so that new records and their index entries may be stored at a later time.

Example A: Ordered Record Keys



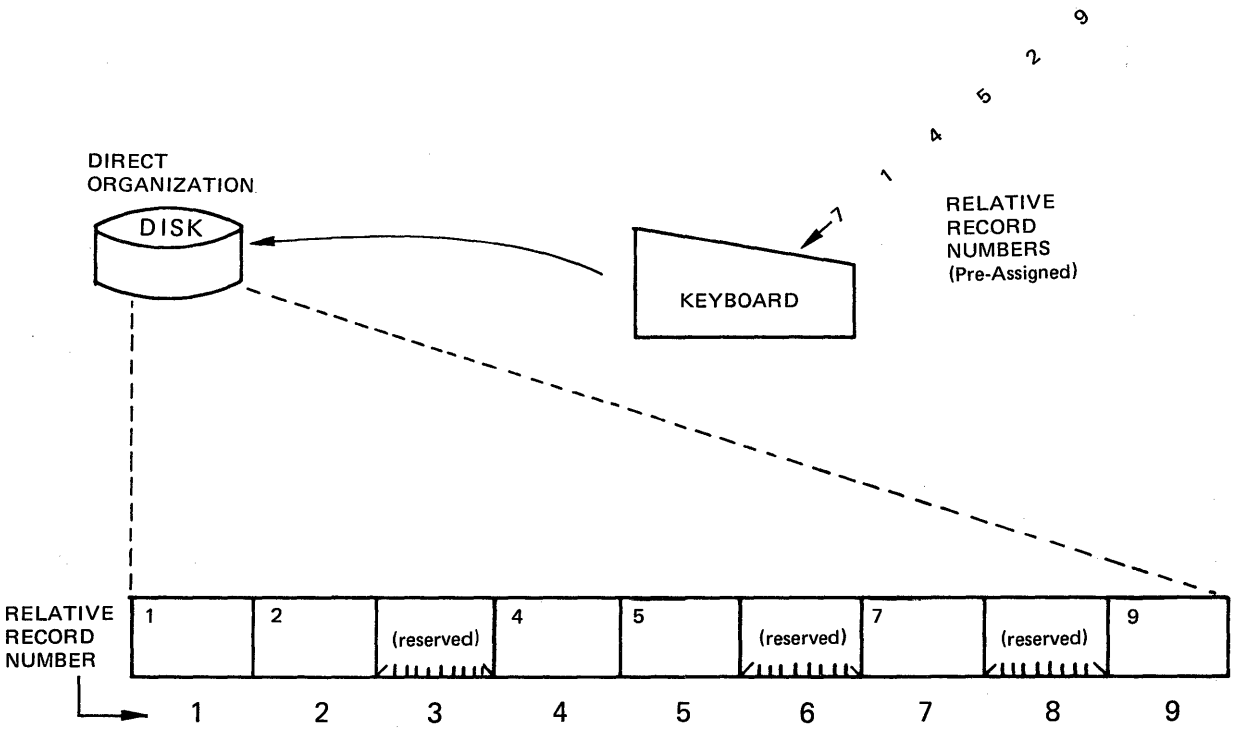
Example B: Unordered Record Keys



DIRECT ORGANIZATION

1. Records are stored in the relative position that is pre-assigned.

2. Unused record positions are reserved. During future processing a new record can be placed into its reserved space.



Selecting the best file organization method when creating disk file records depends upon how the records are to be processed in the future.

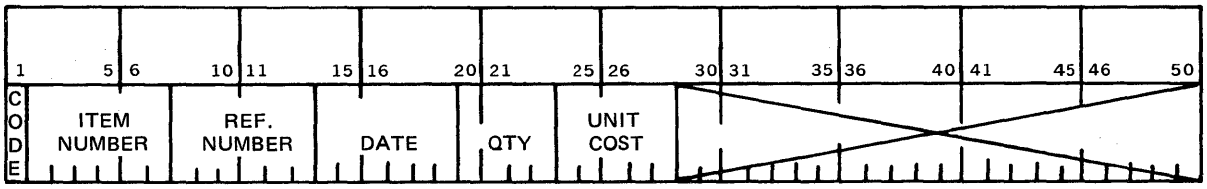
DOCUMENTATION

Planning computer and file usage includes the arrangement of data fields within records. There are special forms for this activity called Record Layout forms and Print Charts.

This is an example of a record layout for a disk file that has 28-character records broken down into 6 fields of data. The title information on the form includes the fact that this file is organized sequentially (ORG. SEQ.).

PROPORTIONAL RECORD LAYOUT FORM

RECORD NAME AND REMARKS
INVENTORY TRANSACTION RECORD LENGTH 28 ORG. SEQ.



The six data fields are:

Code in position 1,

Item Number in positions 2-7,

Reference Number in positions 8-13,

Date in positions 14-19,

Quantity in positions 20-23, and

Unit Cost in positions 24-28.

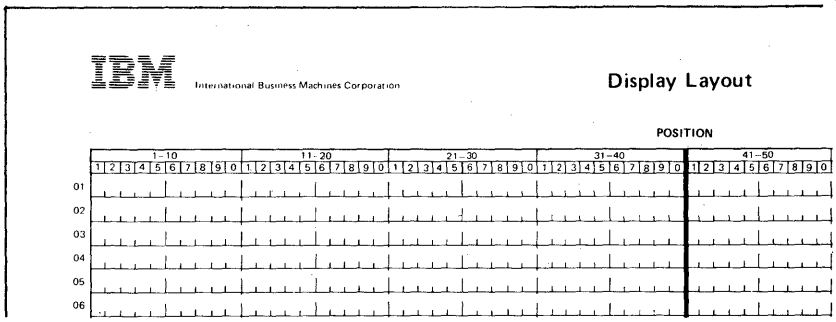
Here's an example of a part of a print chart as used to plan for the printing of an "Inactive Item List". The numbers at the top are print positions. Fields of data and headings are filled in for reference during RPG II coding. You will be taught to use print charts later on in the course.

PRINT CHART

		1 2 3 4 5 6 7 8 9 0 1 1 1 2 3 4 5 6 7 8 9 0 2 2 2 2 3 4 5 6 7 8 9 0 3 3 3 3 3 4 5 6 7 8 9 0 4 4 4 4 4 5 6 7 8 9 0 5 5 5 5 5 6 7 8 9 0 6 6 6 6 6 7 8 9 0 7 7 7 7 7 1 2 3 4 5 6 7
	1	
	2	
H	3	XX/XX/XX INACTIVE ITEM LIST
	4	
	5	
H	6	ACTIVITY ITEM ITEM ON O.H.
H	7	DATE NUMBER DESCRIPTION HAND VALUE
	8	
D	9	XX/XX/XX XXXXXX X X X,XXX XXX,XXX.XX
	10	
	11	
T	12	NO. OF ITEMS LISTED XXX
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	

H = Heading line
D = Detail line
T = Total line

Similarly, you can design display screen layouts on a form like this. For the System/32 you use up to 6 lines of 40 characters.



File Processing

In this section we will be studying three methods used to process disk file records.

1. consecutive - process each record in a file starting with the first and continue until all have been processed.

2. sequential - process each record in an indexed file in the order of the record keys in the index.

3. random - process selected records. Selection is either by relative record number or by key field.

We will want to refer to this chart a number of times as we consider the processing of records for any kind of file.

PROCESSING ORGANIZED FILES

	FILE ORGANIZATION		
HOW FILE IS PROCESSED	SEQUENTIAL	DIRECT	INDEXED
CONSECUTIVELY	YES	YES	
SEQUENTIALLY	NO	NO	YES
RANDOMLY	YES BY RELATIVE RECORD NUMBER	YES BY RELATIVE RECORD NUMBER	YES BY KEY

First, let's consider this example.

"Print a list of every record in a disk file that is organized sequentially".

From the chart, which processing method should be used?

Since we are processing all of the records in a sequentially organized file, the method used would be consecutive. That is, we will locate the first record in the disk file and print it; locate the next record in the disk file and print it and so on until all records have been printed.

Which processing method would you use for each of these examples?

_____ 1. Print a list of every record in a direct file.

_____ 2. Print a list of every record in an indexed file in key sequence.

_____ 3. Process a sequential file in order to create an indexed file that has the same records in it.

_____ 4. Inquire into a direct file in order to find the account balance for records 2, 5, 12 and 7.

The answers to the questions should be:

1. consecutive
2. sequential
3. consecutively process the sequential file records
4. random

As we've used the term "process" when dealing with disk files it refers to doing something with records in an existing file. Which of the following file types can be processed?

_____ 1. an input file

_____ 2. an output file

_____ 3. an update file

Processing applies only to input (1) and update (3) files.

The word "processing" is a general term rather than a description of a specific kind of activity. I've listed some commonly used words and phrases that you will need to know about in order to code solutions in the RPG II language.

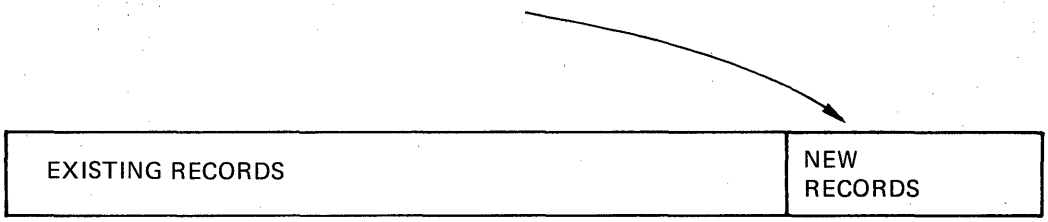
FILE MAINTENANCE

File Maintenance is performing functions to keep a file current for daily processing needs such as adding a new customer or updating an account balance.

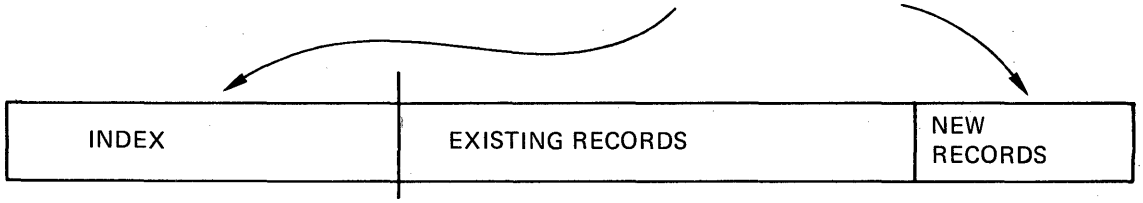
ADDING RECORDS

1. Adding records to a file - the file maintenance function used to add new records to those in an existing file.

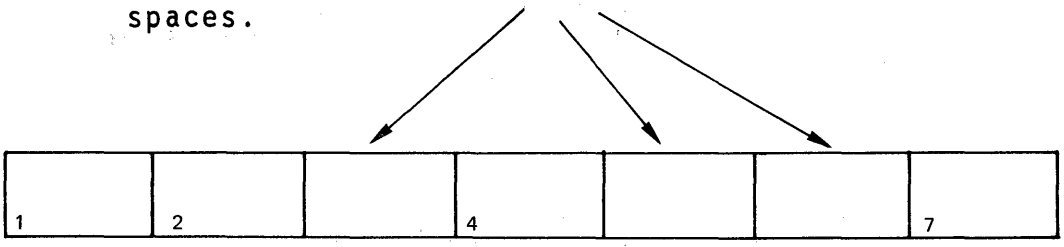
   a. To a Sequential File: new records are placed behind the existing records.



   b. To an Indexed File: new records are placed behind the existing records and new index entries are merged into ascending sequence.



   c. To a Direct File: new records are assigned relative record numbers of unused (reserved) spaces.



UPDATING RECORDS

2. Updating records - the file maintenance function in which the contents of one or more fields of data in a record are to be changed.

   Example:

   "I have just moved. Please change my address to 12 THIRD ST. NEWTOWN, OK 37115. My customer number is 4041".

BEFORE

A	4041	ME	BOX 75	WESTOP, AK 67811

AFTER

A	4041	ME	12 THIRD ST.	NEWTOWN, OK 37115

TAGGING/FLAGGING FOR DELETION

3. Tagging records for deletion - the file maintenance function used to identify those records in a file that shall no longer be active.

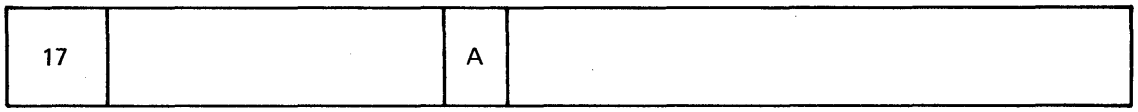
   Flagging records for deletion - same as "tagging" records for deletion.

   In every case, the record to be tagged for deletion is "updated" by changing the contents of a code field. For example, active records might contain the letter "A" in this code field while records tagged for deletion might contain the letter "D" in the same field after the updating takes place.

Example:

"Tag record 17 for deletion".

BEFORE



AFTER

REORGANIZING FILES

Reorganizing a file - the file maintenance function during which records identified for deletion at some earlier time are removed from the file.

a. A sequential file containing one or more records tagged for deletion is reorganized by copying all active records into an unused area on the disk, that is, by creating a new file. The file would be processed consecutively.

   NOTE: If records have been added to the original file, and the new file is to have its records in sequence, it should be sorted.

b. An indexed file containing one or more records tagged for deletion is reorganized by copying the active records in record key sequence. The file would be processed sequentially in record key sequence.

   NOTE: Since record keys in the index are already in ascending key sequence, no sorting is necessary.

c. A direct file is not reorganized in the same sense as the other files, because the space used by a record flagged for deletion is available for use. The relative record number assigned to a "deleted" record can be re-assigned to a new record, and the new record replaces the old.

Each of the file maintenance functions may be programmed in the RPG II language. In fact, it is possible to do more than one of these maintenance functions in a single program. Some of these functions are included in exercises in a later chapter of this RPG II programming course.

Any file maintenance function must be carefully planned and tested before being used with active files because:

1. there may be no record of original data,

2. updated records must be verified as to their accuracy,

3. after reorganization, deleted records may be lost forever.

Record Grouping

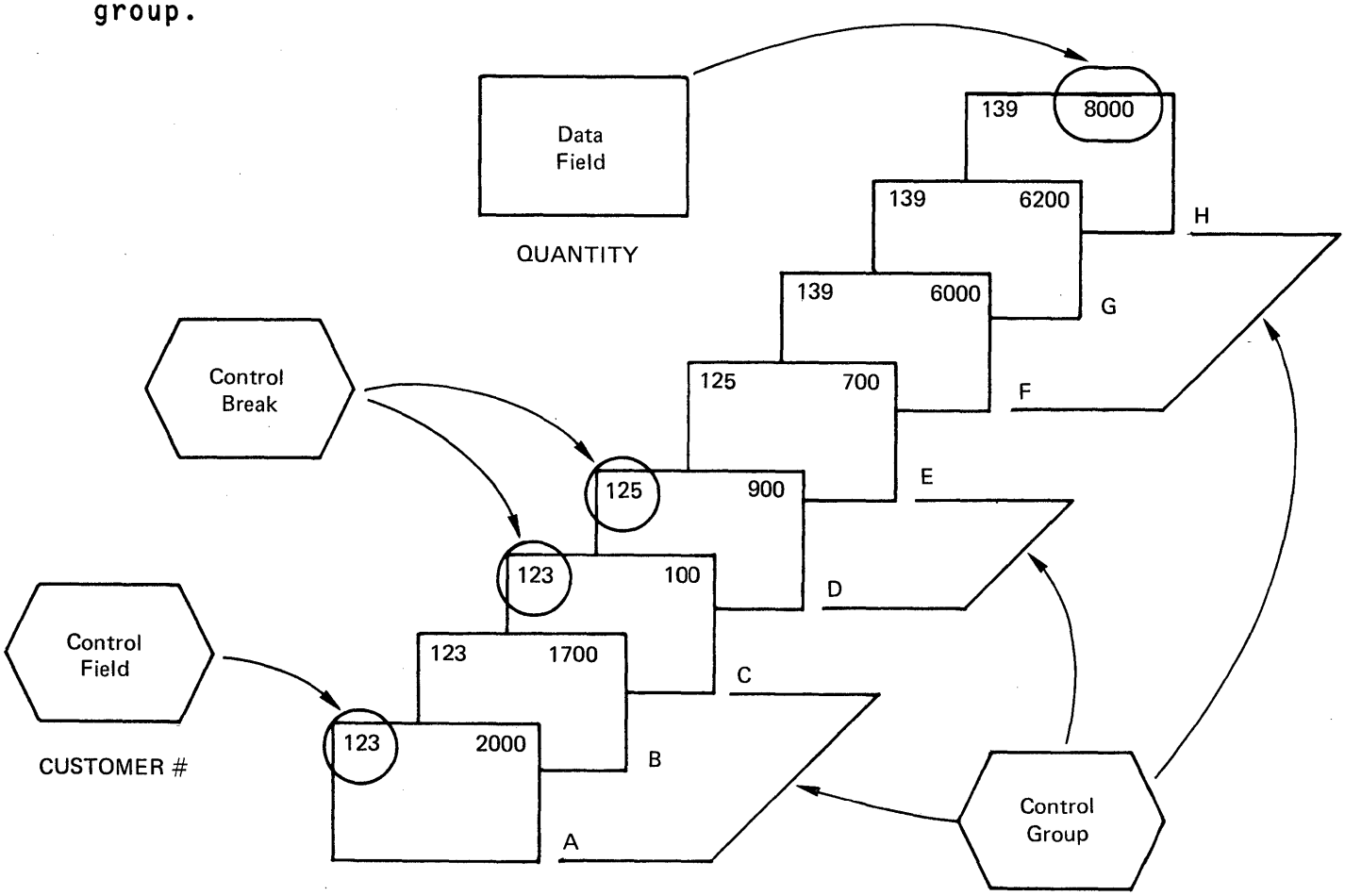
In the first part of this topic we talked about processing one record at a time in order to produce output records. Let's extend your knowledge by including information about record grouping.

1. Control group - a set of records all containing the same identifying information in one or more fields.

2. Control field - one of the fields within a record that identifies the record's relationship to other records, such as customer number.

3. Control break - the occurrence of a change in control field data from one record to the next.

Example: Find the total value of data fields for each control group.



How many records are found in each control group? How many control groups in this example? What is the total data field value for each group in this example?

Control group 123 has 3 records (A, B and C). Control group 125 has 2 records (D and E). Control group 139 has 3 records (F, G and H).

Since a control break occurs whenever there is a change in control field value from one record to the next, there are 2 control breaks in this example. One break occurs between records C and D, and a second break occurs between records E and F.

The total data field values are:

group 1 = 3,800 (2000+1700+100)

group 2 = 1,600 (900+700)

group 3 = 20,200 (6000+6200+8000)

Keep one thing in mind when working with control groups. Each record in a group contributes data value to the group. In some problems, the identifying information and the data field values are shown for each record and for each group. A report of this kind is known as a "detail-printed report", or simply, a "listing".

Detail-Printed Report

RECORD	CUSTOMER #	QUANTITY
1	123	2,000
2	123	1,700
3	123	100
4	125	900
5	125	700
6	139	6,000
7	139	6,200
8	139	8,000

A different kind of report is used to summarize values for each group of records. It is referred to as a "group-printed report". Here's a summary of the data records shown in the previous example. Can you identify the control field?

Group-Printed Report

CUSTOMER #	QUANTITY
123	3,800
125	1,600
139	20,200
	25,600*

Here, the control field is "CUSTOMER #".

One more example. For convenience, control field information is only printed for the first record of a group in some detail-printed reports. These reports are said to be, "detail-printed reports with group indication".

Group-Indicated Report

MONTHLY COMMISSION REPORT

SALESMAN NUMBER	SALESMAN NAME	INVOICE NUMBER	INVOICE DATE	CUSTOMER NAME	INVOICE AMOUNT	COMMISSION AMOUNT
4701	GEORGE WILLIAMS	10542	11/19/68	SPARE PARTS INC	764.32	30.22
		8052	11/19/68	HARRIS ENGINEERING INC	34.56	1.73
		18232	11/23/68	OVERSEAS CONTRACTS INC	947.62	47.38
		15305	11/27/68	FIX-ALL COMPANY	1,284.38	64.22
						143.55*
5710	MICHAEL GIBBS	13348	11/12/68	JACOBS AUTOS	1,033.80	51.69
		12844	11/15/68	MORRIS ENTERPRISES	532.76	26.64
		15280	11/24/68	ATKINSON SALES INC	22,005.40	1,100.27
						1,178.60*

By comparison, a group printed report for the same set of records would look like this.

Group-Printed Report

MONTHLY COMMISSION REPORT		
SALESMAN NUMBER	SALESMAN NAME	COMMISSION TOTAL
4701	GEORGE WILLIAMS	143.55
5710	MICHAEL GIBBS	1,178.60

Let's next reconsider the three processing methods presented earlier: consecutive, sequential and random. Review the chart briefly, then continue.

PROCESSING ORGANIZED FILES

	FILE ORGANIZATION		
HOW FILE IS PROCESSED	SEQUENTIAL	DIRECT	INDEXED
CONSECUTIVELY	YES	YES	
SEQUENTIALLY	NO	NO	YES
RANDOMLY	YES BY RELATIVE RECORD NUMBER	YES BY RELATIVE RECORD NUMBER	YES BY KEY

Consecutive processing means that every record in a file will be processed in the order read. Sequential processing means that every record in an indexed file will be processed in "record key" sequence. Random processing, on the other hand, means that selective processing is to be done on individual records without regard for the order in which they are placed in the file.

A good example in which random processing is required is this.

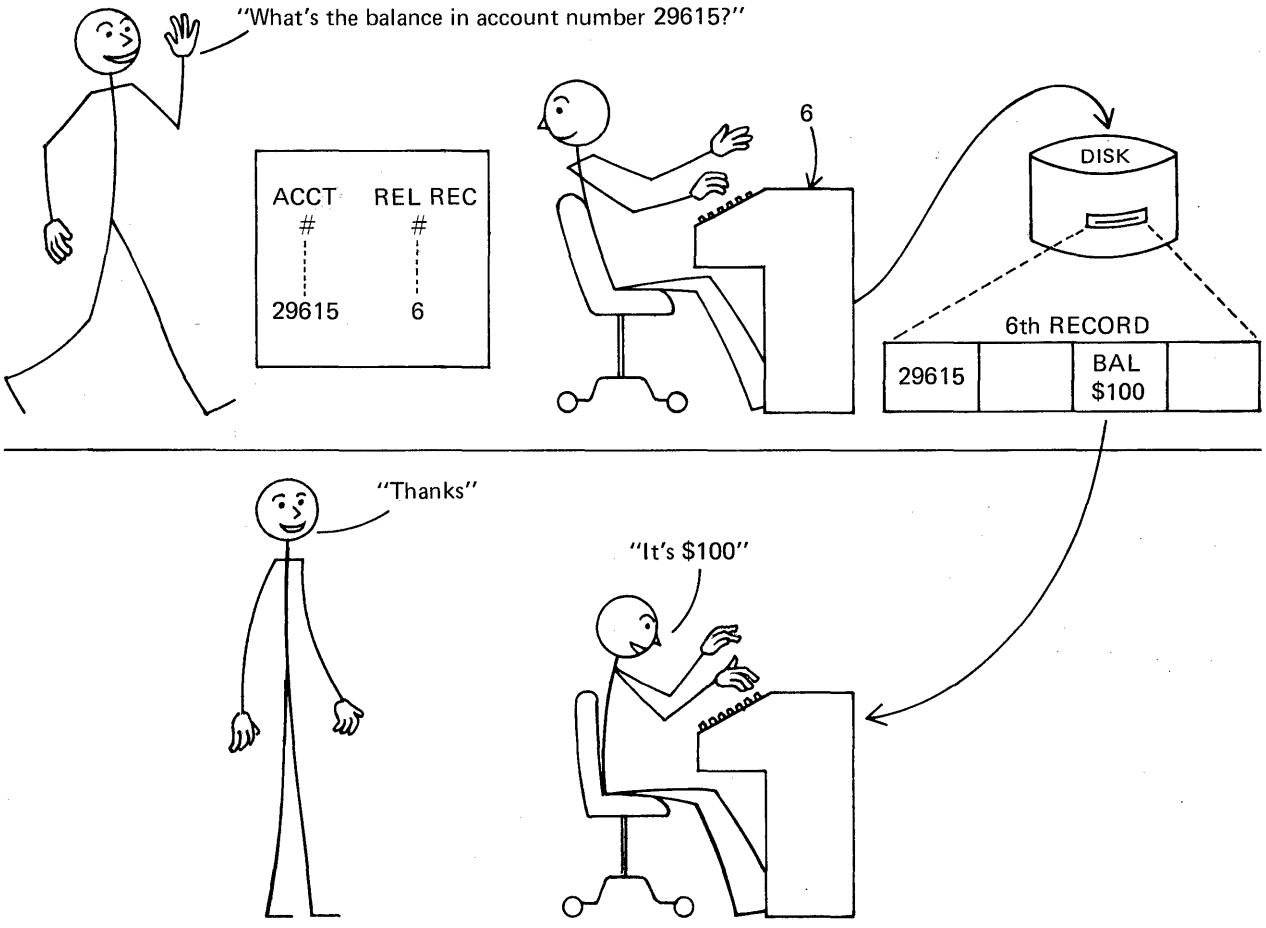
> "Update records in the customer stock purchase account file as transactions occur".

In order that such a transaction can initiate an update activity, a special, record-identifying value must be entered. The process of finding the correct record in order that the updating can take place is known as "chaining".

The process of "chaining" is done by searching a file for a particular record. This process involves the use of 2 files:

1. the "searching" file, and
2. the "chained" file.

First we will consider an example in which we want to inquire into a sequential file to find an account balance. In order to randomly process records in a sequential file we must enter the relative record number of the desired record.



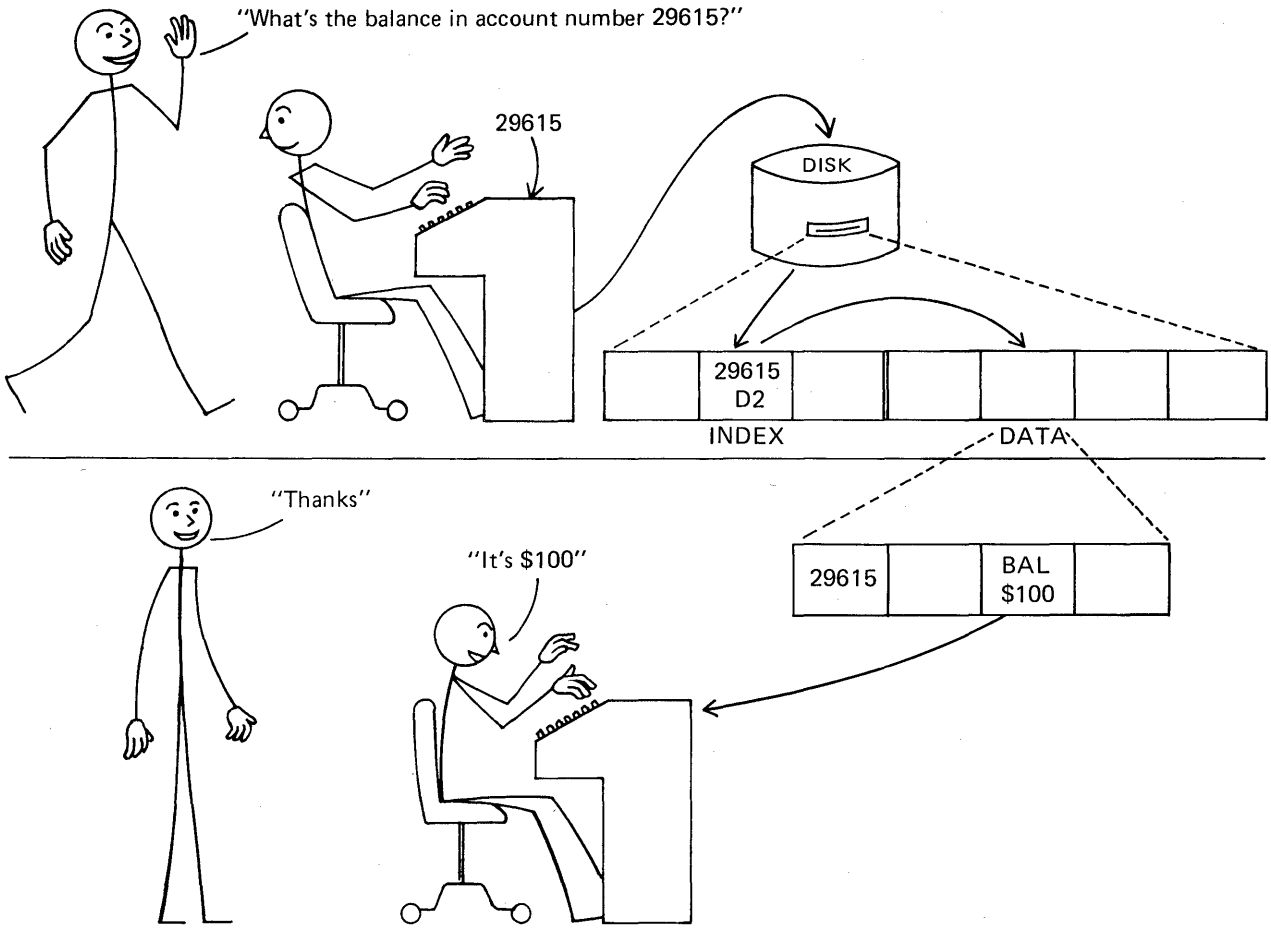
What you have just seen is true when you randomly process records in either:

1. a sequential file, or

2. a direct file.

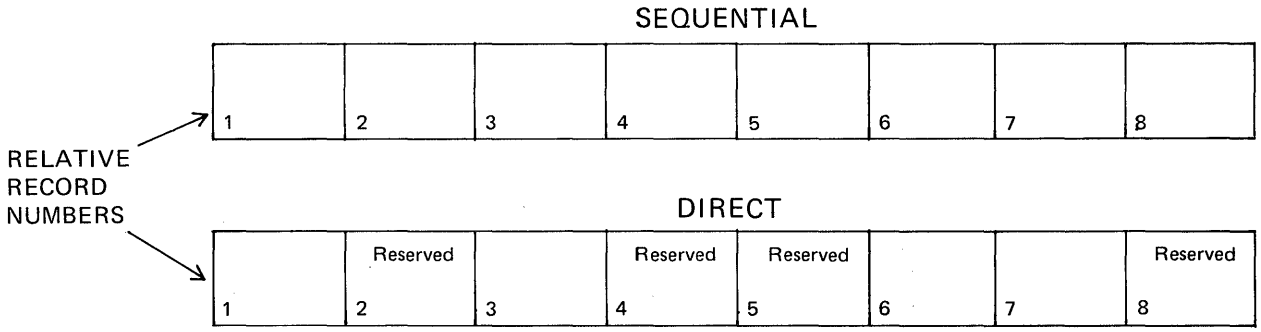
The relative record number is used to locate the desired record in the chained file.

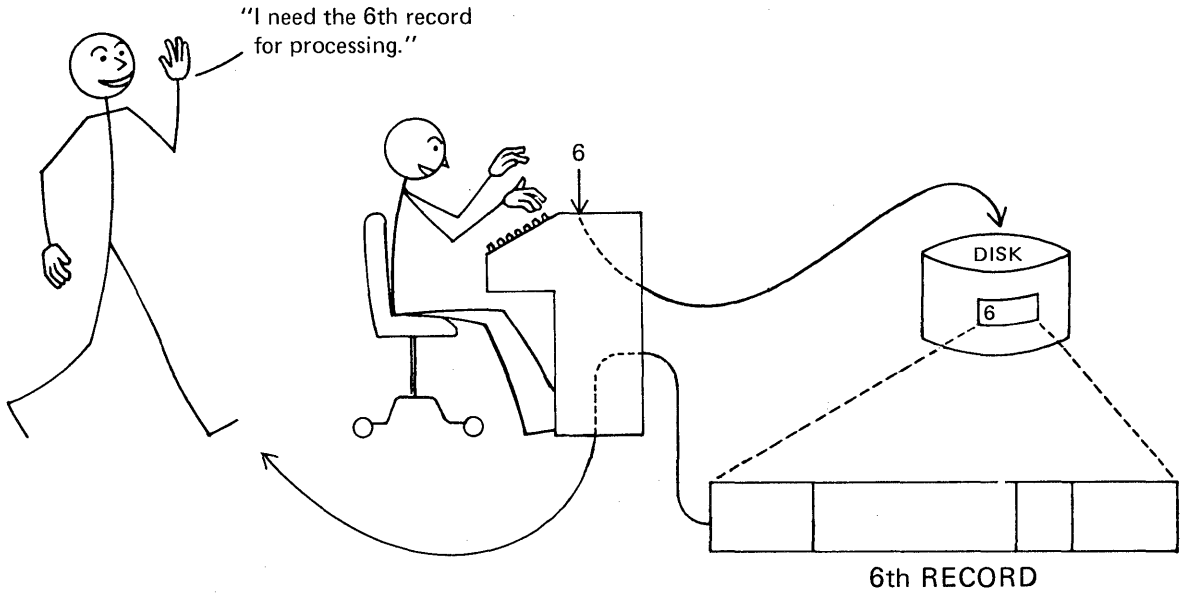
Randomly processing records in an indexed file is essentially the same. The one difference is that a "key field" value is used to search the index, which in turn locates the desired data record.



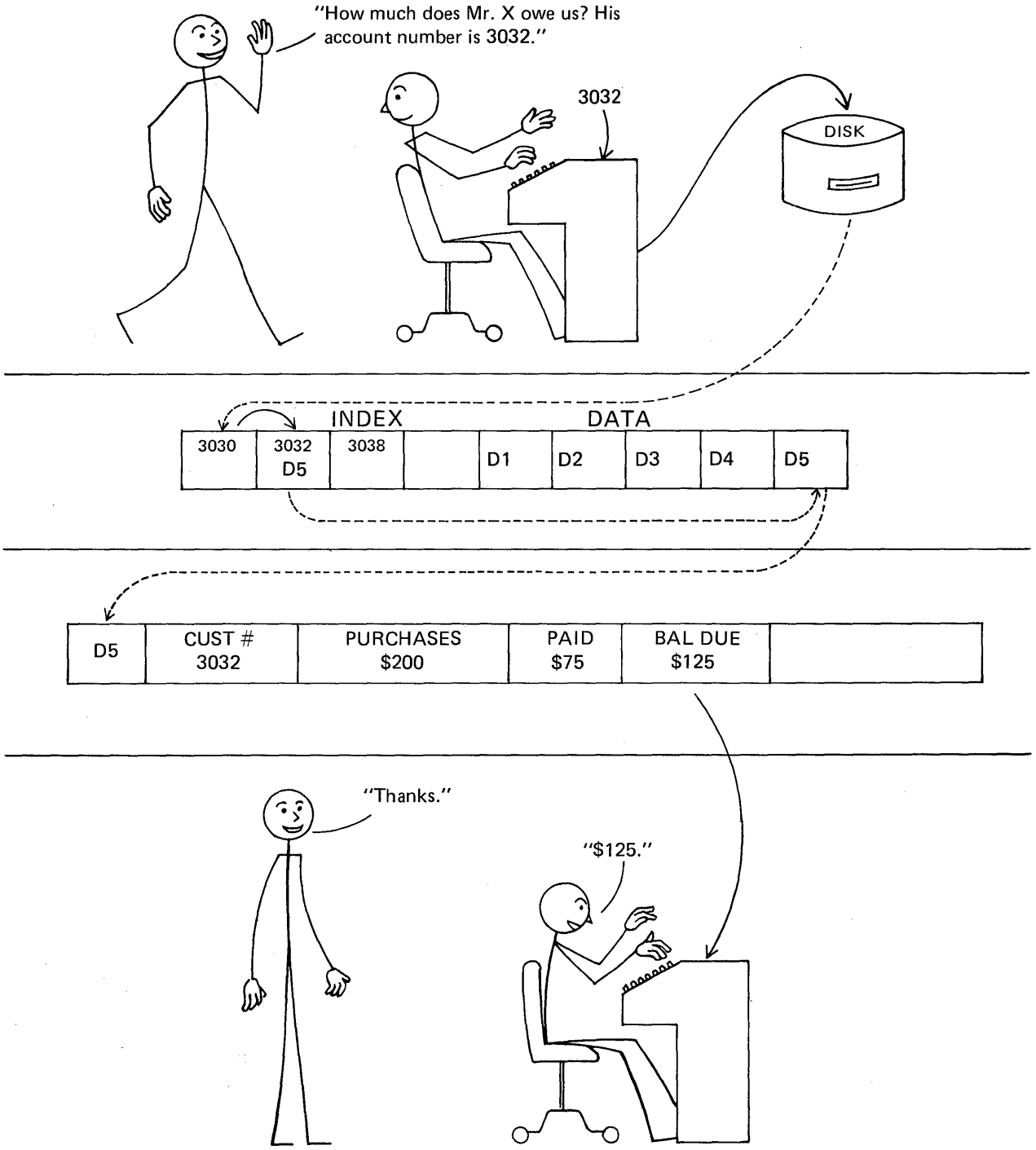
To summarize this brief presentation on the use of chaining to randomly process records in any disk file, read through these examples.

1. When using either sequential or direct file organization, searching is accomplished by using a "Relative Record Number". In this approach, the location of a particular record is known by its position relative to the first record in that file. The first position is referred to as relative record number 1. The second as relative record number two. The third as relative record number three, and so on.





2. When using <u>indexed file organization</u>, searching is accomplished by using a "Key Field". In this approach, the actual identifying information is contained in the index of the indexed file. Searching is done by comparing key field information against the information in the index, and then the desired record is retrieved and processed.



Another kind of file processing that can be described in the RPG II language is known as "multifile processing". As the name implies, two or more input (or update) files provide data to be processed by the program. A basic rule you need to remember is that one and only one of these files is designated as "primary"; all other files for the same job are designated as "secondary". The programmer makes this designation when a solution to a multifile problem is coded.

RPG II can process multiple files using either of the following approaches:

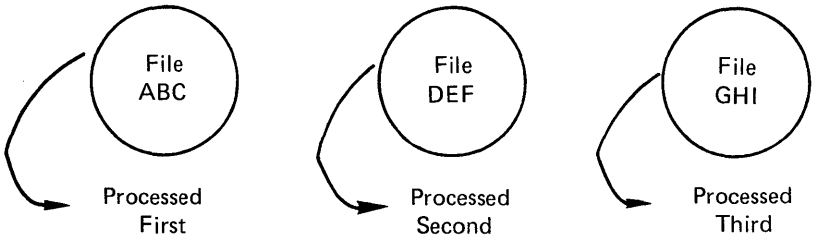
I. No special identification.

   A. First process all records in the primary file.

   B. Next process all records in the secondary file that is named next.

   C. Process all records in each of the remaining secondary files in the order that the files are named.

Example: Process the records in files ABC, DEF and GHI. The primary file is ABC.



II. Matching fields identification.

   A. A particular field of information is identified in records in each file.

   B. When a "match" occurs, the primary record with the match is processed ahead of the secondary record that also matched. If "no match" occurs, process the next record in sequence.

Example: Process the records in these two files using the matching fields approach. The sequence in both files is ascending order.

Primary File		Secondary File	
Record	Match Field Value	Record	Match Field Value
P1	27	S1	35
P2	35	S2	36
P3	42	S3	43
P4	49	S4	48
P5	56		

Here is the order in which the records will be processed. Do you agree?

P1, P2, S1, S2, P3, S3, S4, P4, P5

In order for this approach to work properly, you can see that each file must have records in the same sequence. The sequence may be either ascending or descending and the fields for matching may be numeric or alphameric.

The approach works the same way for three files.

Primary		First Secondary		Second Secondary	
Record	Match Field Value	Record	Match Field Value	Record	Match Field Value
P1	20	FS1	20	SS1	20
P2	30	FS2	30	SS2	40
P3	40	FS3	55	SS3	50
P4	50			SS4	60
				SS5	70

Records in the three files will be processed in this order.

P1, FS1, SS1, P2, FS2, P3, SS2, P4, SS3, FS3, SS4, SS5

As you work with multiple files and the "matching records" approach, you must know:

1. Which file is to be designated as primary?

   Each application must be examined before File Designation entries are specified. When information from one record in one file is needed in order to process two or more records from a second file, the first file should be designated as "primary".

2. In what order are secondary files to be named?

   Since records from secondary files are processed in the order in which the files are named, be sure to specify secondary files in the desired order for your particular job.

3. Which field of data shall be used as the basis for determining whether or not records match?

   Normally this field is the one that was used to determine the order of records when the file was first created. If the file had been sorted prior to doing a job, the match field will probably be the one on which sorting was based.

4. In what sequence are records in these files, ascending or descending?

   Either you would know this or someone would have to tell you the sequence of the records.

Programming

Programming is data processing problem solving. It starts with the question, "What's the problem?", and ends with a tested solution to that problem. In this course you will be taught to code solutions to data processing problems in a programming language known as RPG II. In this section we will be examining the kinds of activities that an RPG II programmer does except for coding which is covered in the remainder of the book.

There are a number of activities that may be referred to as "fundamentals of programming". We will present information about the following seven "fundamentals" in this section.

1. The need to define the problem to be solved,
2. the use of orderly procedures to solve the problem,
3. the creation and use of test data,
4. the technique of desk checking,
5. basic knowledge about the programming language to be used in coding a computer solution,
6. a knowledge of available reference information, and
7. the handling of either single or multiple files of data.

Fundamental: Define the problem to be solved.

Persons who are good problem solvers generally tend to ask these three questions.

1. What kind of answer is wanted?

2. What kind of information already exists about this item?

3. Which methods should I use to get the answer wanted from the information already known?

In programming, we say we have defined the problem when we know the answers to these three questions.

RULE: To define a problem in data processing,

1. determine what you want to have as output, and

2. gather all that you know about the available input, and then

3. select one or more methods to process the available input to get the desired output.

Example:

You have been elected treasurer of your neighborhood social club. At the first meeting of the new year you will be expected to present a treasurer's report. How will you solve the problem?

As you play the role of the club treasurer, I am sure you will start thinking about what is expected of you. Let's consider the upcoming first meeting and raise three questions.

1. What is wanted as output?

2. What is available as input?

3. What method should be used to prepare and present this report?

Obviously, the desired output is some sort of report. How should it look? What will it contain? Will it be a verbal or written report? Should copies be made for everyone?

Now for the input. I would start by asking, "Who was the treasurer last year?" Then I would find out from that person how much money is in the treasury now; on what kinds of items has money been spent; where does the money come from, and so on.

One more step, the processing needed. "What method should I use to get the report ready using the information available to me?" I have got to know that a treasurer's report must balance. That is, last month's balance plus income for this month minus expenses paid this month must equal the amount of money left in the treasury - or we have got a bigger data processing problem than I thought when I was elected to the office.

As you can see, a programmer is a person who must ask many detailed questions in order to do the job of solving problems. It is important that as much information be gathered as possible so that no points are overlooked in arriving at a solution.

We did not mention it, but keep in mind that the output produced is normally needed by someone who did not have a hand in its preparation. The treasurer's report is read (or listened to) by everyone at the meeting. The information must be useful to the users, not just to the preparer.

Fundamental: Use orderly procedures to solve problems.

Computers work at very high rates of speed, but they do only those things they are directed to do. It is very important that you, the programmer, prepare detailed instructions to control the computer to solve your problems as you want them solved.

One way to prepare an orderly procedure is to list all of the steps to be done in the order in which they are to be done. This method is convenient when there are few steps and they occur in a set order. It can also be used when there are just a few "branch points" in the sequence. A branch point is a step where a choice of two or more sequences of steps must be made.

Example: Write a set of procedure steps to solve this problem.

> Compute the amount of pay a person is supposed to receive for one week. To compute it, multiply the hours worked by the rate per hour. If the total exceeds $200, print a note along with the answer so it can be checked by a supervisor.

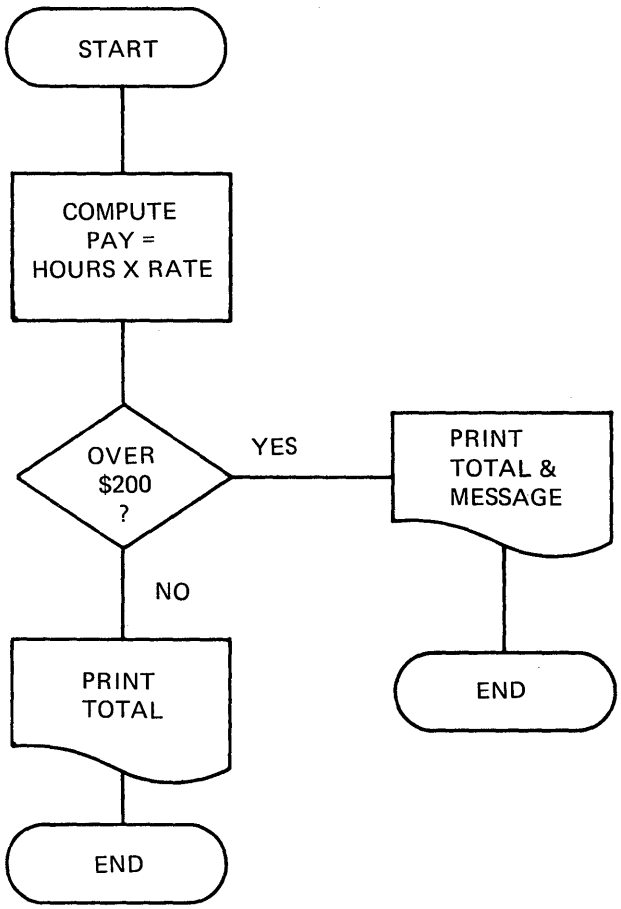
Procedure Steps

1. Compute pay amount by multiplying hours worked by rate of pay.

2. If the amount is over $200, print a note next to it.

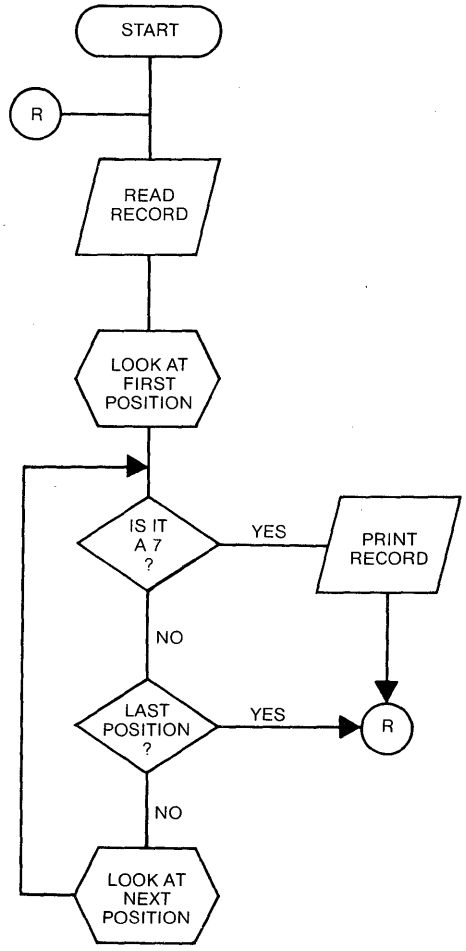
Another method used to document orderly procedures is known as flowcharting. Specially shaped symbols and connecting lines are used to draw a chart of the procedure steps, and arrows may be included to indicate the flow from step to step.

Here is a flowchart showing the procedure used to solve the same problem we just documented by the step listing method.

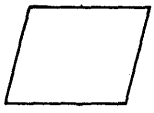


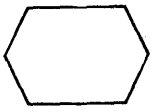
Were you able to trace through this flowchart? Notice that we used oval-shaped symbols for "start" and "end" points. We used rectangles when showing calculations, and we used a diamond to designate a branch point.

Now let's look at another flowchart. What new symbols are used in it?



This example uses three new symbols.

1. The parallelogram 

2. The hexagon 

3. The circle 

The parallelogram is used to show that the computer is to read in a record (input) or produce a new record (output) and so it is commonly known as an input/output symbol.

The hexagon is used to indicate some kind of change or modification is to occur and is known as a modification symbol.

The circle is used to indicate connecting points. That is, points on the chart to which the reader shall be directed next. Circles are called "on-page" connectors because they direct the reader to mentally connect points on the same page that is being examined.

Look at the chart to see another way of connecting points on a flowchart on the same page. What is it? (3 stars below mean "answer the question, then continue".)

* * *

A simple way of connecting points on the same page is to use flow lines with arrowheads to indicate the direction of flow. You may draw charts using either system.

RULE: Keep the flowchart simple.

If too many connecting lines are drawn they may make it difficult for the reader. Use on-page connectors to eliminate confusion, but try to use flow lines as they are easy to trace.

As you read the flowchart, what problem solution do you think is being described?

* * *

If you think something like this you are correct.

> Read all input records one at a time. Examine each position in the record for the digit 7. If a 7 is found, print the record for reference. If no 7 is found in any position in the record, read the next record and start searching in its first position.

Look at the flowchart again. What special entry is made at each branch point? The words YES and NO are shown to make certain that the person who reads the branch point question will follow the proper path. When you draw flowcharts include the words YES and NO for clarity.

In my example, YES directed you to the right. There is no special rule for this, so you can do it any way at all. The key is to label them correctly for the reader.

RULE: Include flow line arrows and words like YES and NO to help readers trace the flow properly.

Since many people draw flowcharts in a similar manner, there are acceptable conventions to simplify their use. For example:

Convention: When flow lines direct you to the right or down on the page, no arrowheads are needed.

We have illustrated the use of two methods to document orderly procedures. The flowchart is commonly used for computer applications. To get more information about flowcharting, read the booklet, "IBM Data Processing Techniques - Flowcharting Techniques" which has the form number GC20-8152.

Flowcharts may be sketched on any sheet of paper or a chalkboard, but to standardize size of drawings and symbols, some programmers make use of flowcharting worksheets and flowcharting templates. These items are illustrated in the flowcharting techniques booklet.

1. GX20-8020 IBM Flowcharting Template
2. GX20-8021 IBM Flowcharting Worksheet

Fundamental: Create and use test data.

After an orderly procedure for solving a data processing problem has been written down or flowcharted, it should be tested for accuracy. Here is how a programmer does it.

First, write down a set of values for every field of data in each type of record for every input file type that will be used in the program. Second, try out each record by going through the procedure or flowchart steps and recording field values and computed values on scratch paper as you go. After all records have been tested and their values recorded, verify that the results are as they should be in order to satisfy the requirements of the problem being solved. The second activity is known as desk checking.

RULE: Create enough test data to test every stated problem condition.

RULE: When acceptable input test data does not follow an acceptable path for calculations or desired output, re-do the procedure and test again.

Now look back at the last flowchart. How much input test data would you need in order to test every condition in the problem?

* * *

1. I need at least 2 records. One must include a digit 7 and the other cannot include a digit 7.

2. I might want a few more records. I want a record with 7 in the first position; one with a 7 in the last position, and one with a seven somewhere in the middle. Also, a record without a 7. That means I should create 4 test records.

Record	Data (assume records have 12 characters of information)	
1	720694332408	(7 in first position)
2	111111666697	(7 in last position)
3	123456789101	(7 near the middle)
4	246810121416	(no 7)

One of the most common errors made by beginning programmers is to fail to create thorough test data. Remember what we said earlier - computers do only those things they are directed to do. If your procedure is wrong or your test data does not test every condition in the problem, errors may go undetected.

Fundamental: Desk checking

As a programmer you are a data processing problem solver. First, you solve problems on paper and then you convert the paper solutions to coded computer solutions. Desk checking is an activity that occurs after you have solved the problem on paper and before you run the solution on the computer. It is that activity used to verify the accuracy of your paper solution. As we just discussed, here is what is involved in desk checking.

1. The problem solution is written or flowcharted.

2. A set of test data is created by you. Test data must include all combinations of values, fields, records and files described in the problem itself.

3. Records are traced through the procedure step-by-step to verify the procedure's accuracy.

4. A list of all input records, calculated results and output produced as a result of following the procedure during desk checking are retained for reference.

The more complex the problem, the more complex the task of desk checking. Detailed procedures are more desirable than generalized procedures because they provide for more accurate checking of each step. If errors exist, they can be pinpointed and corrected more easily in a detailed procedure.

Fundamental: Basic knowledge about a programming language

A programming language such as RPG II is a specialized set of codes to be used to translate your paper procedures into computer usable procedures called programs. To gain a working knowledge of a language, you will need to study it in a course such as this and become familiar with its use by trying to code problem solutions. You are expected to refer to other sources including books and experienced programmers whenever you are assigned the task of writing complex programs. The more you know about a particular programming language the easier it is to use. You should become familiar with the facilities of a language as well as its limitations. In some instances, you can "program around" a difficult set of steps for which no direct solution is apparent. The greatest aid in this area is experience with as many of the programming language's facilities as you can master.

Fundamental: Knowledge of available reference information

More than likely, your department will have a reference library containing up-to-date books for your use. You should always check on the changes to these books that are made when facilities are added to a language.

IBM provides updates to its publications by way of Technical Newsletters. These booklets contain replacement pages and new pages for existing manuals. A list of changes is included so that it can be filed in the updated manual, thereby giving you a source of the changes you should review from time to time. Find out where your programming reference library is located and how it is maintained.

Fundamental: Handling single or multiple files of data

A computer program controls the reading of input data, the processing of that data, and the production of corresponding output data. Some programs use data from one input file to produce output in another. More complex programs involve the reading of input data from two or more input files in order to produce output in a number of output files.

For each program that you write, you must know how many input files are to be used and you must know how many output files are to be produced. In addition to "how many files" there are, you must know what kinds of records are contained in each file, what kinds of data groups (called fields) are contained in each type of record, and how the files are organized.

Example:

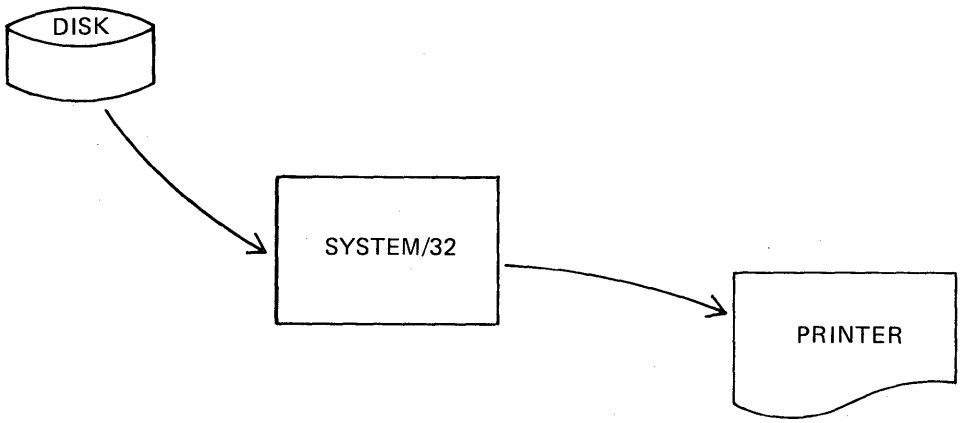
> A file of student records has been created in a school system. All records are similar. Each contains a name field, a student identification field, a class rank field, and two entrance test score fields. Print a list of all of the student records.

How many input files exist? How many output files will be created? How many different types of input records are to be processed in this job?

* * *

There is one input file of student records. One output file is to be created (the printed list of student records is the output file). Since all input records contain the same kinds of data in the same fields, there is only 1 input record type to be processed.

A form of flowchart, sometimes referred to as a system flowchart, is a convenience in picturing what files are involved in the problem to be solved. The system flowchart below describes the problem.



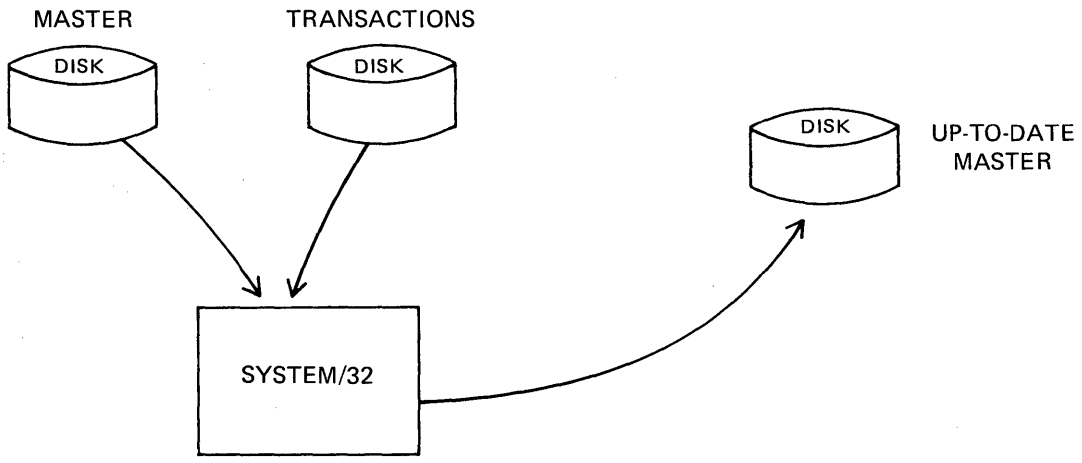
Here is another example:

> A certain bank keeps a file of customer master records for each savings account. Each of these records contains an account number, the name of the person, the balance in the account, and the date when this record was created. As business goes on each day, it is necessary to create a second file of data. There are four types of records in this second file: new account, closed account, deposits to an existing account and withdrawals from an existing account. A program is to be written so that an up-to-date file of savings accounts customers can be created at the end of each week. All records in the second file include an account number and an investor name for reference.

Now that you have read the example, make an analysis of the desired program. How many input data files are used? How many output data files will be created? How many different input record types are used in this example. Sketch a system flowchart if you wish.

* * *

There are two input data files. One is the file of customer master records, while the other is a file of transactions for a week. One new master file is to be created as output. Five different input record types are used in this example: one type in the master file and four in the transaction file.



There are some very important rules of thumb involved in programming. Here is one you will want to remember.

RULE: Input field data is never changed during a program.

Now let us consider the processing aspects of the two examples. In the first, we were to create a file which was a printed list of students. In the second, we were to create an up-to-date file of savings accounts.

No specific processing was requested for the first example. Simply stated, the problem involved: read a record, print that record, repeat until all records were read and printed.

The second example involves a variety of processing. Consider the transaction file record types.

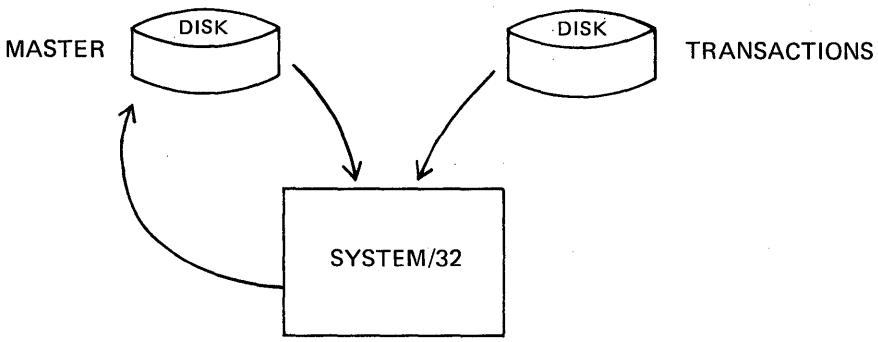
Record Type	Processing
1. New Account	None
2. Closed Account	Balance should be zero
3. Deposit	Add amount to old balance
4. Withdrawal	Subtract amount from old balance; see if new balance is less than zero (overdraw)

Not only do we need to know exactly what processing is to be done for each type of record, but we must also know something about the way in which each file is organized. Assuming that the master file is in order by savings account number, we would want the transactions file to be in order by savings account number also, so we can match them against the masters. Unfortunately, banking isn't done in account number order by customers. They do banking when they need to. Even though the problem said nothing about this condition, the transactions would probably be put into account number sequence (sorted) before the job in example two was run. More than likely, all of the records for the same account in the transaction file should be grouped in this manner for convenience. First, a new account; second, all deposits; third, all withdrawals; fourth, a closed account record.

The reason I mention this is that, though unlikely, it is possible that a person could open a new account on Monday, make a number of deposits or withdrawals during the week and close it out on Friday.

Key Point: Someone must determine the manner in which files are to be organized and determine the method of processing to be used before a program is written to solve the job. That "someone" is usually an officer in the company.

In example two, we stated that a program was to be written in which an up-to-date file could be created. Rather than creating a new file of up-to-date records each week, we might have solved this problem by "updating" the master file records.



Definition: An update file is one in which data may be changed as a result of processing.

An update file is both an input and an output file.

If we solve example two using the master file as an update file, the transactions directly affect only those accounts for which activity took place during the week. Here is how it would work if the master file was on disk and transactions are grouped by account number. All new accounts are placed last in this transaction file.

1. Read a transaction record.

2. Process that record and store the results in the central processing unit as a temporary record.

3. Read the next transaction record.

4. If it is for the same account, update the data stored in the temporary record and read another record.

5. If it is for a different account,

    a. find the master record that corresponds to the temporary record,

    b. update the master record data on disk,

    c. set the temporary record space in the central processing unit to blanks.

6. Go back to step 2.

7. When all transactions have been processed, create all new account records on disk and then end the job.

One final point regarding the handling of single or multiple files of data. The more specific information you have about a problem the better you can solve it. Before writing a program, find out all you need to know about the files, records, fields and special codes in them in order to produce the desired output. When in doubt, ask questions of the person who gave you the assignment.

When coding solutions in the RPG II language you will need to know about "counting and accumulating".

COUNTING AND ACCUMULATING

Definition: Accumulating is the addition of values from one or more data fields into a counter.

Definition: Counting is accumulating by adding 1 to a counter. The 1 is known as a literal or constant.

Definition: A counter is a temporary storage area, used for accumulating values from data fields or literals.

Problem: Read and process all of the input records in the billing file. Print a list of these records. Count each data record when you read it and print the total number of records counted when you reach the end of the job. Do not count the special-purpose last record.

NOTE: The last record does not contain data, it is an extra, special-purpose record found at the end of each input file. Its purpose is simply to identify the end of the file so that a program can end a job when it is read.

In order to use a counter properly, we must be sure of these things. Start by setting the counter to zero. Add the literal (1) to the counter each time a data record is read.

Okay, try your hand at solving this problem. You may either use a list of steps or draw a flowchart to represent your solution.

* * *

When I solved this problem, I considered these points as being significant; did you?

1. I must set the counter to zero before counting begins.

2. I must count only data records.

These are the steps I would write. I also included a flowchart for your reference if you used that approach.

1. Set the counter to zero.

2. Read a record.

3. Is it the "last record"?

4. If it is not,

   a. count it by adding 1 to the counter,

   b. process the record,

   c. print the record,

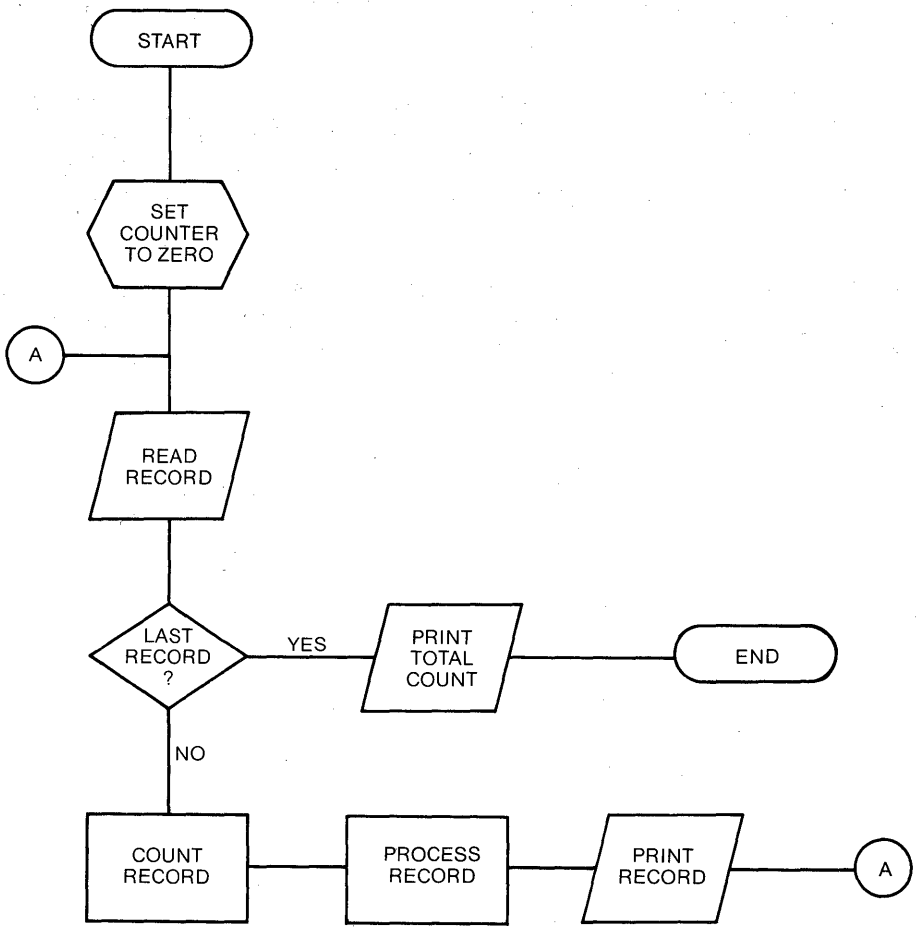
   d. repeat the activity by going back to step 2.

5. If it is the last record,

   a. print the total number of records counted,

   b. end the job.

Flowchart



Now examine my flowchart again and answer these questions.

1. Why is the last record not counted?
2. Must we count a record before it is processed?
3. Must we count a record before it is printed?

* * *

The last record does not contain data, it is an extra, special-purpose record at the end of each input file. In this example, we may count a data record before it is processed and printed, or after it is processed and before it is printed, or after it is printed.

Solutions in flowchart form are known as program flowcharts because they show the flow of program steps used in a problem solution. A system flowchart shows a flow of file information into and from the system that is to do the processing.

Many data processing problem solutions depend upon the use of tables of data. A table is simply a list of similar items sometimes called table elements. Here are 3 tables. Can you think of others?

TABLE 1	TABLE 2	TABLE 3
1	JAN	A
2	FEB	B
3	MAR	C
4	APR	D
5	MAY	E
6	JUN	F
7	JUL	G
8	AUG	
	SEP	
	OCT	
	NOV	
	DEC	

When tables are used we normally go through a process known as "table lookup".

Example: What's Ace's phone number?

The tables I use to solve this problem are found in a phone book. Yes, it really contains 2 tables! One is the alphabetical list of names, and the second is the list of corresponding phone numbers.

NAME	NUMBER
ABC	204-1451
ABX	256-1192
ACE	232-7783

The process of doing a table search is the same as you or I would use.

1. Someone asks, "What is Ace's number?"
2. You look down the name table until you find ACE.
3. You look across the page to the corresponding entry in the number table.
4. You say, "Ace's number is 232-7783."

As you can suspect, when a data processing problem solution requires the use of tables, the table data must be included as test data so that the problem solution can be verified.

What does a programmer do?

A programmer is a person who defines problems in such a way that a computer can be used to get answers each time that kind of job needs to be run. The programmer creates a procedure for doing the problem after it is solved. By that, I mean that the computer is used to repeatedly perform the same sequence of steps (called a program) for each record contained in a file.

Ah ha! The key to programming lies in writing one sequence of steps that can be stored in the computer and be used over and over at high rates of speed for many records! The computer is capable of doing what it is instructed, but the programmer must create the program.

To write a program, a programmer does these things, usually in this order.

1. Define the problem to be solved in terms of what is wanted and what data will be used.

2. Create an orderly procedure that will solve that problem.

3. Test the procedure using appropriate test data to verify the correctness of the procedure.

4. Convert the orderly procedure into a computer program using coded statements in a language such as RPG II.

5. Test the computer program by using exactly the same test data as was used to test the procedure in order to verify the correctness of the coded statements.

6. Document the program test run for future reference in case of a need for modification to the program.

RULE: If the program produces errors in the test run, recode the incorrect portion of the program.

RULE: If the procedure steps produce errors during desk checking, rewrite the procedure steps.

What does a programmer do? He is really a data processing problem solver who is capable of translating a "paper solution" into an error free computer program.

What tools does the programmer use?

A tool that is used by programmers is the flowchart. Other tools include worksheets, coding sheets and lists of test data that already exist in a given computer installation.

What sources of information are available?

A programmer uses knowledge of applications that will be done on the computer. Reference books on particular applications are a good source of information. Another source is experience in solving the same kinds of data processing problems before a computer was available.

A programmer uses knowledge of the particular coding language to be used when converting paper procedures into computer programs. A source of information is the programming language reference manual.

How long does it take to become a "good" programmer?

Well, I asked the impossible question. Let's see how much of an answer I can give you. I will describe a "good" programmer's activity and then suggest some period of time it may take you to be able to do the same activities.

A good programmer:

1. thoroughly analyzes each problem to be solved, asking questions for clarity whenever a doubt arises.
2. is familiar enough with the program coding language to be used so that efficient programs are coded.
3. tests every program condition and verifies the results before releasing the program for general use.
4. documents the solution to every program.
5. recognizes mistakes and makes corrections as needed.

How long does it take to become a good programmer? Steps 1 and 2 require experience. You may need from 6 months to a number of years to gain this kind of experience on the job. Steps 3 and 4 require self-discipline. If you already are a stickler for detail you will do them from the start of your training. If not, it may take months to force yourself to change. The last step is one that involves attitude. If you realize that the computer works at extremely high rates of speed but does exactly what it is instructed to do and nothing more, you will be more likely to follow through on this step.

If you recognize that you are the translator for this "foreign language machine", you will accept the need to re-translate programs a number of times during test runs in order to make the computer do its job as well as you do yours. This may take some time but probably is only a few months.

- -

That completes your study of some fundamentals of programming. Do not hesitate to re-read any part or all of this topic before you begin your study of RPG II programming language.

Good luck to you in your work as a programmer!

Chapter 1: Introduction to RPG II (Study time is 1 to 2 hours)

The intent of this chapter is to familiarize you with RPG II as a programming language. When you have completed the chapter, you should know what RPG II is and in general, how it is used.

The content of this chapter is presented as a series of questions and answers including two completely coded programs for your review. You will be asked to examine these programs in order to identify entries that are common to more than one sheet.

Every time that you code a solution using the RPG II language you do the steps listed below. In this self study course you will be directed to perform the first three steps only. Steps four through seven require the use of an IBM System/32.

1. Analyze a data processing problem.

   a. What is wanted as output?

   b. What is available as input?

   c. What processing needs to be done?

2. Code an RPG II solution.

   a. Describe the files (input and output).

   b. Describe the records and fields in those files.

   c. Describe the calculations needed to process the input file data.

3. "Desk check" the coded solution.

   a. Any obvious coding errors?

   b. Any omissions?

   c. All processing steps included and in the right order?

   d. Does test data "flow" through the coded solution correctly?

4. Compile the solution. This translates the RPG II statements into machine-readable statements that the System/32 can use to run the job.

5. Test the compiled program.

   a. Use the test data created for desk checking.

   b. Know in advance what the output must look like.

   c. Verify the output of the test run.

6. Re-code a solution if the test run is in error. This is normal activity of all programmers and known as "debugging" a program.

   a. Re-compile the new solution.

   b. Test the re-compiled program.

   c. Be ready to make additional corrections and repeat the process until it is a proper solution to the question, "What is the problem?"

7. Document the solution. Save all printed test results and related input data records for reference.

What is RPG II?

RPG II is really two different things:

1. It is a descriptive, programming language, and

2. It is a generative program product.

RPG II, when thought of as the generative program product, works this way:

1. A problem solution that has been coded in the RPG II language is prepared for compilation. The coded solution is known as a source program.

2. During compilation, syntax-checking (looking for coding errors) is done for each statement. If unacceptable errors are found, reprogramming is required. If no unacceptable errors are found, an object program is generated. The generated object program is then used to run the job after it has been tested with simulated job data and found to be acceptable.

How does a programmer use RPG II?

The programmer analyzes the entire problem to determine what output is desired, what input is available, and what calculations are needed to modify the available input in order to produce the desired output.

The order in which he codes a solution is not critical, but prior to compilation, he will arrange his coded statements in the order required by RPG II for the compilation process.

Each coded statement is a description of a part of his solution. For example, he will code a description of each input and output file, making an entry for each on a File Description sheet. He will describe every type of input record found in each input file by coding entries on an Input sheet. This coding will include one statement about each input record type and one or more statements about fields of input data found in the various record types. He will describe every output record type to be produced for every output file by coding entries on an Output sheet. This coding will include one statement about each output record type and one or more statements about fields of output data or the constants that are to be a part of the various output record types. For each calculation that is required to produce the output records, he will code one or more statements on a Calculation sheet.

Are reference materials available to the programmer?

There are special rules for making coded entries on the RPG II sheets just as there are rules in any language. If all coded entries are made according to the rules, there will be no syntax errors during the source program compilation process. The set of rules for using the RPG II language is found in the book, "System/32 RPG II Reference Manual", form number SC21-7595.

When you have completed this RPG II programming course, you will be familiar with the use of this reference manual. You are not expected to remember every coding rule, so get acquainted with the book as you do practice problems in this course.

What System/32 features can be described using RPG II?

The System/32 has a number of devices that may create and/or store files of data. Input files may come from the keyboard or from the disk. Output files may be found on the disk, on printed reports or temporarily on the display screen. Although information and programs may be stored on the diskette, that information cannot be processed until a separate program stores it on the System/32 disk. RPG II cannot directly process diskette data, nor can it be used to directly create an output file on a diskette.

What other items are related to the use of RPG II?

There is a System/32 program product known as SEU, the Source Entry Utility program. Its purpose is to allow the keying of an RPG II source program onto the System/32 disk after which that program may be compiled by RPG II. SEU includes the capability of performing some RPG II syntax checking which reduces the burden and total time of compilation. SEU also allows modification and addition to, or deletion of, RPG II statements for convenience in program testing.

Another System/32 program product known as DFU, the Data File Utility program, has the facility to use certain RPG II statements about input file records to do utility-type functions such as listing records.

Any more information about coding sheets?

There are other RPG II specification sheets which will be discussed during the course. Like the ones mentioned earlier, each serves a particular function related to describing files, records and fields. The entry in column 6 on each sheet is a code to remind you of its purpose.

H is the Header (control) statement

F is the File Description sheet

E is the Extension sheet

L is the Line Counter sheet

I is the Input sheet

C is the Calculation sheet

O is the Output sheet

What does a completely coded solution look like?

The following example has been coded in RPG II using File Description, Input, Calculation and Output specification sheets. This program was named EXMP01, and its purpose is to print a listing of 45-character records found on the System/32 disk. We will discuss this solution shortly, but at this time you should take a few moments to scan each entry on each sheet. Note in particular those items that are spelled alike on different specification sheets.

EXMP01

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines K	Continuation Lines Option	Continuation Lines Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	SEQDSK	I	P					45								DISK													
03	F	SDLIST	O						132				OF				PRINTER													

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence (OR AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank
01	I	SEQDSK	NS			01																								
02	I		NS																		1	5		ITEMNO						
03	I																				6	30		DESC						
04	I																				31	34	0	QTYOH						
05	I																				35	41	2	UPRICE						
06	I																				42	45		WHSE						

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators And Not	Indicators And Not	Indicators And Not	Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions	Half Adjust (H)	Resulting Indicators Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
01	C		01			COUNT	ADD	1	COUNT	5	0											COUNT RECORDS
02	C																					

RPG OUTPUT SPECIFICATIONS

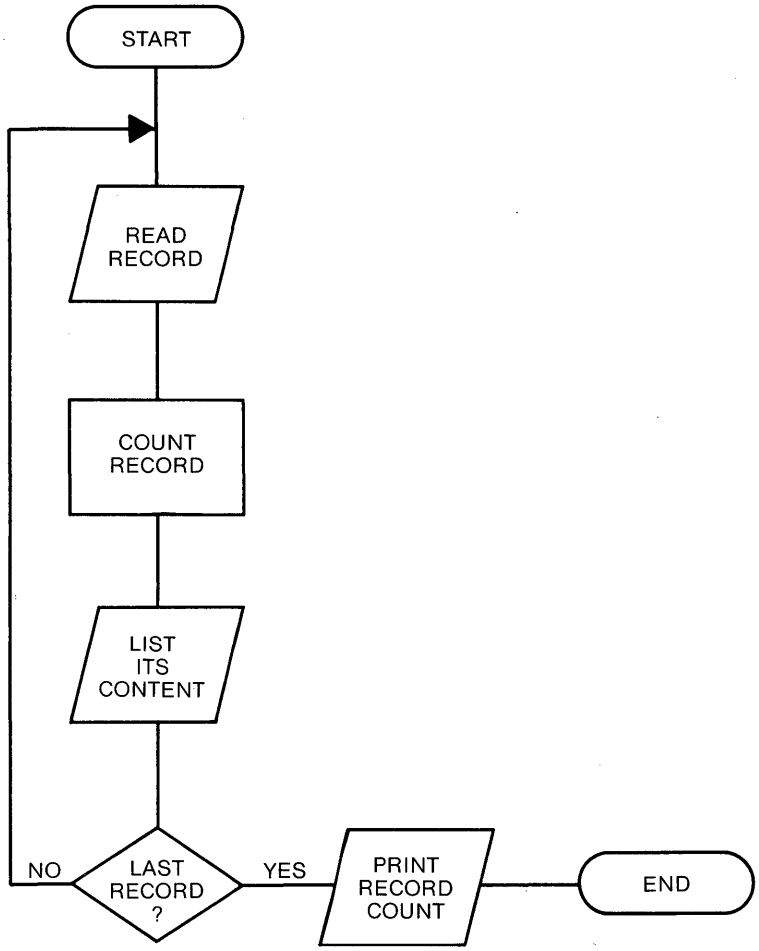
Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	QDLIST	H		2	06			1P							
02	O		OR						OF							
03	O													51		'STOCK ITEMS LIST'
04	O		H		3				1P							
05	O		OR						OF							
06	O													18		'I T E M'
07	O													50		'QUANTITY'
08	O													66		'UNIT PRICE'
09	O													77		'WAREHOUSE'
10	O		D		2				01							
11	O											ITEMNO 2		10		
12	O											DESC		40		
13	O											QTYOH 1		50		
14	O											UPRICE 1		66		
15	O											WHSE		77		
16	O		T		2				LR							
17	O											COUNT 1		21		
18	O													37		'RECORDS IN FILE'

Commas | Zero Balances to Print | No Sign | CR | -

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Here's a flowchart showing what the programmer was trying to describe as a solution:



Since the flowchart does not include specific coding information, let's examine each entry and relate it to the problem to be solved.

File Description

The input file was given the name SEQDSK by the programmer. He may choose any convenient name, but it must start with a letter and be no longer than 8 characters. The remaining characters may be either letters or numbers. The entry "I" in column 15 identifies this file as an Input file. The entry "P" in column 16 designates this file as the primary input file for the job. The remaining two entries for Record Length and Device satisfy the problem statement, "...of 45-character records found on the System/32 disk".

How would you describe, in your own words, the second file description statement?

You should be thinking something like this:

The file was named SDLIST by the programmer.
Column 15 contains an O to identify this file as an output file.
The output is to be printed on a device that can print records of up to 132 positions in length.

The other entry, OF in columns 33-34, is a selection by the programmer from a set of acceptable overflow indicator codes. Overflow as used here refers to the overflowing of heading information on printed reports. More about page overflow later.

Notice that the programmer enters only that coding needed to describe his job; he does not make entries in every position.

Input

Next, examine the Input sheet entries. The first statement describes the type of record found in the input file while the remaining entries describe the fields in each record of that type.

RULE: The input filename must be identical on the File Description sheet and on the Input sheet.

To complete a description of this first statement, the programmer coded "NS" in colunns 15-16 and "Ø1" in columns 19-20. If letters are used in columns 15-16, it means that sequence of record types is not significant to the solution of this problem. When only one record type is identified, specify any two letters in these columns. The "Record Identification Indicator" (Ø1) is a choice made by the programmer. RPG II provides a set of 99 numeric indicator codes from which the programmer may select as many as needed to solve his problem. In this case, he specified Ø1 to represent the identification of this particular type of record.

Suggestion: Draw a slash through zeros on your coding sheets to prevent the error of thinking it is the letter O.

Now examine the field descriptions on lines two through six. All have a specified "Field Location" and "Field Name". Three of them include an entry under "Decimal Positions" while two of them do not. Any idea why this is so?

If you are thinking, "Decimal positions are only used in fields of numeric data", you have the right idea. An entry of Ø in column 52 means that the numeric field has whole number data, for example, an item number. The other fields are called alphameric and may contain letters, numbers, special characters or blanks.

RULE: Field names can be from 1 to 6 characters long and must start with a letter. The remaining characters may be either letters or numbers.

Suggestion: Make up names that refer to the kind of field data contained in each record.

Here, ITEMNO is the item number field while UPRICE is the unit price.

Now look at the Field Location entries once more. How long is the field that is located from 35 to 41? The correct answer is 7 places. How long is the field named WHSE? How about DESC? WHSE is 4 positions long, while DESC is 25.

RULES: Numeric fields may be from 1 to 15 positions long with zero or up to 9 decimal positions.

Alphameric fields may be from 1 to 256 positions long.

In review, the programmer describes every input record type and every input field he will use to solve his problem. The first input statement describes the record type while the following entries on the Input sheet describe each field in that record type. Filenames and field names are made up by the programmer. He also selects the record identifying indicator.

In order to specify the fields in these disk records, someone must have provided a list of them to the programmer because they were not mentioned in the problem statement.

Next, examine the Calculation statement. In this case, the programmer included a comment for reference only, see columns 60-74. This is optional. Let's talk about counting records. The operation is ADD and that seems reasonable since counting is usually associated with adding 1 to a prior number and repeating it until you've counted as high as you need. The 1 to be added in this example is shown in column 33. The result of the addition will be stored in a new field called COUNT which is 5 positions long with no decimal positions. The entry in columns 10-11 tells the computer to count every 01 record type in the file. That takes care of every entry except COUNT in columns 18-22.

Here's how the programmer was thinking when he coded this calculation statement:

> I'm going to count a lot of records, one at a time. I'll define a numeric result field of 5 positions so I can count as high as 99999 which should be adequate for this job. The counting shall take place when indicator 01 is turned ON to indicate that the computer read another record for processing. In order to count correctly, I must add a 1 to what's in my counter from the last time I added 1. So, when 01 occurs, I want to take what's already in COUNT, add 1 to it, and store the result as before.

RULE: To do counting, specify the indicator that controls "when" counting shall take place, then add 1 to a counter you've defined. Use the counter's name in two places. Be sure the counter is large enough to hold the largest number you expect to reach in the job.

Note: The generated object program automatically sets this counter to zero before the job starts to run.

Output

As you expected, the filename is identical to the output filename found on the file description sheet.

Examine the various entries made in column 15. How many different types of output records will be printed? The code "H" refers to heading information. The code "D" refers to detail information contained in the disk input records. The code "T" refers to a record containing totals. Even though there are only three different code letters in this example, there are actually four different types of output records described on this sheet. The first heading record is a report title. The second heading record provides column headings. Both of these heading records are to be printed whenever indicator 1P is turned ON or when OF is turned ON. Indicator 1P means "first page" while overflow (OF) refers to all pages that follow the first one.

Look at the first line. The entry in columns 19-20 direct the printer to skip to line 6 on the page before printing the report title, while the entry in column 18 directs it to space 2 after printing occurs.

What do you think will happen regarding skipping and spacing for the second heading record? No skipping will occur, but the printer is directed to space three after printing the column headings, either on the first page or the overflow pages.

RULE: When no space or skip entries are coded on an OR statement, the space and skip entries in the statement above it will automatically be used for the OR statement also.

Next, examine all of the entries related to the detail record information. When is a detail record to be printed? What spacing or skipping is to be used? Are field names the same as those found on the Input sheet? How many fields have an entry in column 38?

A detail record is to be printed when indicator 01 is ON. The printer is to space two after printing the information found in the five fields. The field names are identical to those found on the Input sheet, and the three fields that have entries in column 38 (Edit Codes) are the three numeric data fields. Edit codes provide for suppression of leading zeros and the insertion of commas and decimal points where needed. A chart at the top of the Output sheet provides a brief reminder of the purpose of each code.

Using that chart as a reference, what is to be done with the numerical data when it prints?

The item number (ITEMNO) will be printed without leading zeros because code Z does only zero suppression. Both other fields (QTYOH and UPRICE) will be printed with a comma inserted between hundreds and thousands place as needed.

Note: The code 1 also provides for placement of a decimal point except for fields that have only whole numbers. It also provides for zero suppression except, as the chart states, it will print a zero when the entire field's value is zero.

The last type of record to be printed is the total record line. It shall be printed when indicator LR is turned ON. The indicator LR refers to the end of the job; that is, after the last record in the input file has been read and processed. To make this record line on the printed output stand out from the detail lines of information, the programmer coded space 2 before and included the constant, RECORDS IN FILE. What else is a part of this last output record?

The result field COUNT will be printed with zero suppression and a comma if needed as a part of this line.

Now that we've examined every entry on every sheet of this example, here are some more rules that you will be using as you do RPG II coding:

1. Filenames and field names are always entered starting at the left.

2. Record Length (on the File Description sheet), Field Location (on the Input sheet), Result Field Length (on the Calculation sheet), and End Position in Output Record (on the Output sheet) are right-justified without leading zeros.

3. On output, when two or more records of the same type are to be produced as output when the same Output Indicators are turned ON, they will be produced in the same order as they are coded from top to bottom on the sheet.

You have already been exposed to a lot of things about the RPG II language simply by looking at that first example. At this point in the course, we want to talk about two features of RPG II that were not used in the first example. They are the Interactive Data Entry feature, and the Auto Report feature.

Interactive Data Entry

This feature allows an operator to enter input file records from the keyboard while the object program is running. As each field of data is entered, it is displayed on the display screen for operator reference. When an entire record has been keyed in, the program may use it for processing. An independent storage area called a buffer holds the record until it is requested for processing by the object program.

Some general rules for using these files are:

1. File Description sheet - The device name is CONSOLE.

2. File Description sheet - Only 1 CONSOLE file is permitted in a program.

3. Input sheet - In addition to assigning a record identifying indicator, a record identification code must be specified.

4. Input sheet - Fields must be specified as being contiguous, that is, each field is next to the previous field. Example: Field LOC is in positions 2-6 and field DEP is in 7-9.

When you name input fields, select those names that are as meaningful as possible, because the object program displays this name on the screen as a prompt to the operator when the program is run.

We will look at a sample program in which a CONSOLE file is used in a few minutes.

Auto Report

This feature accepts special, simplified RPG II specifications along with other standard RPG II specifications in order to solve a data processing problem. Auto Report includes three separate functions that can be used in any combination. They are:

1. *AUTO, to simplify the specification of report page headings, column headings, positioning of data fields from left to right, and certain calculations for the accumulation of totals.

2. /COPY, to permit the use of specifications defined for one program and already cataloged for reference to be copied intact into another program without re-coding.

3. U specifications, to permit modification to an RPG II source program that is produced by Auto Report as an intermediate step during a pre-RPG II compilation run.

In the next sample program we will see the use of an Interactive Data Entry file and the use of the first Auto Report function, *AUTO.

Look at the sample program named EXMP02. You should examine it in two ways before continuing with your reading of this topic. First, as a review of those points discussed as a part of EXMP01. Try to identify all of the entries that are similar to those you saw before. Second, look for ways in which EXMP02 is different. As you look over this sample program, here's the problem that the programmer was describing:

> Print a report that shows the contents of a file of records entered from the keyboard. Accumulate totals for gross pay, federal income tax paid, social security tax paid, and net pay.

EXMP02

File Description Specification

Line	Form Type	Filename	File Type (I/O/U/C/D)	File Designation (P/S/C/R/T/D)	End of File (E)	Sequence (A/D)	File Format (F/V/S/M/D)	Block Length	Record Length	L/R	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type (A/P/I/K)	Type of File Organization or Additional Area (I/D/T or 2)	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines (K)	Option	Entry	File Addition/Unordered (A/U)	Number of Tracks for Cylinder Overflow	Number of Extents	R/U/N	Tape Rewind	File Condition U1-U8
02	F	INRECS	I	P				72	70									CONSOLE													
03	F	REPORT	O						132									PRINTER													

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence (OR/AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	INRECS	AA			01	1		CE		2		C2																	
02	I																				3	7		0MAN						
03	I																				8	16		SOCSEC						
04	I																				17	17		FINIT						
05	I																				18	18		SINIT						
06	I																				19	40		NAME						
07	I																				41	45	2	GROSS						
08	I																				46	50	2	FIT						
09	I																				51	55	2	FICA						
10	I																				56	60	2	NET						

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Output Indicators Not	Field Name *AUTO	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	REPORT	H									*AUTO				
02	O															'PAY RECORDS REPORT'
03	O		D						01			*AUTO				
04	O											FINIT				'FIRST'
05	O												C			'INITIAL'
06	O											SINIT				'SECOND'
07	O												C			'INITIAL'
08	O											NAME				'LAST NAME'
09	O											SOCSEC				'SOC SEC #'
10	O											MAN				'MAN'
11	O												C			'NUMBER'
12	O											GROSS	A			'GROSS PAY'
13	O											FIT	A			'FED INC TAX'
14	O											FICA	A			'SOC SEC WHLD'
15	O											NET	A			'NET PAY'
16	O															
17	O															

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

What new entries do you find on the File Description sheet? Naturally, you might expect to find different file names, which you did. The description of the input file includes a Block Length entry as well as a new Device entry, CONSOLE.

RULES: If CONSOLE is specified, the file is an Interactive Data Entry file. Records will be keyed in from the keyboard.

The Block Length entry for this file must be at least two more than the Record Length entry and cannot exceed 4096.

What new entry do you find on the Input sheet? There are entries under Record Identification Codes in columns 21-34. They describe the record in this way:

> The input file named INRECS contains one type of record to which record identifying indicator 01 is assigned. This record type is further described as having the character E in position 1 and the character 2 in position 2.

RULE: If a CONSOLE file is described, it must contain a record identification code in position 1. It may also include a second record identification code in position 2.

Next, examine the data fields. How many contain numeric data? How many contain alphameric data? There are 5 numeric fields and 4 alphameric fields.

RULE: Fields in console files must be contiguously defined.

Do the field descriptions satisfy the rule? It seems so, but look at the field named FINIT. How big is a field that extends from position 17 to position 17? The rule is satisfied. Both FINIT and SINIT are 1-position fields for first initial and second initial respectively.

Now look at two related entries:

1. The Record Length on the File Description sheet, and

2. the largest entry in Input for Field Location.

Though you would expect them to be the same, they do not agree. This is because fields in console files are treated in a special manner by RPG II.

RULE: The Record Length entry for a console file is calculated by adding the entry for the highest Field Location and the number of input fields and 1.

In this program, Record Length = 60 + 9 + 1, which is a total of 70.

In review, a console file is specified on the File Description sheet and the Input sheet. When such entries are made, the operator will be prompted to key in data for each specified field as the program runs. The Record Length for such a file is a computed value that includes the number of input fields to be keyed as well as the highest location, plus 1. A Block Length entry is also required. Its value must be at least 2 more than the Record Length entry.

In this example, we've used the *AUTO function of Auto Report to simplify and reduce the amount of coding required to solve the problem. Look at the Output sheet. The "H" entry in column 15 and the *AUTO entry on columns 32-36 designate this output record type to be an Auto Report heading. Using only the entries on lines 1 and 2 of this sheet, Auto Report will generate a program that includes the printing of a report title, PAY RECORDS REPORT, on the first page and overflow pages. This title will be centered on the report page. Also, a report date will be printed to its left and a page number to its right. Spacing and skipping will be included automatically.

When *AUTO is used for detail type records, a "D" in column 15 and *AUTO in columns 32-36, the fields named from top to bottom on the coding sheet will be printed in order from left to right on the report, with 2 spaces between them. Edit codes will be assigned to numeric fields as will the End Position in Output Record!

The constants specified in columns 45-70 will become the actual column headings for the corresponding fields. These headings, like the report title, will be printed on all pages along with appropriate spacing.

Examine the entries in column 39. The code "C" is used to indicate that the constant in 45-70 is to be printed as a continuation of the column heading specified on the previous line. That means that the field contents for FINIT will be printed under a column heading like this:

```
FIRST
INITIAL
```

How will the column headings for SINIT look? What about the column headings for MAN? Each of these fields will be printed under a two-line column heading similar to the one for FINIT. What about the field called SOCSEC? This field will have a single line column heading because no continuation code and constant are specified for it.

The last four entries will be printed under their respective single-line column headings. Since the code "A" was specified for these fields, Auto Report will generate a series of calculations in order to accumulate totals for each.

That's the way a programmer uses *AUTO entries to simplify his coding and reduce the time needed to code a problem solution. In this particular example, there are 27 coded entries. To code the same solution in standard RPG II statements, the programmer would need to code 53 statements.

Suggestion: Make use of the Auto Report feature whenever the opportunity presents itself, as it will cut down your programming coding time.

The purpose of this Introduction to RPG II chapter was to provide you with information about RPG II as used on the IBM System/32. You have:

1. examined two coded examples,

2. read about using the Interactive Data Entry feature, and

3. read about using the Auto Report feature.

As you recall, RPG II is a descriptive programming language. Your job as a programmer is to describe, as precisely as you can, solutions to data processing problems. In the next chapter, you will be asked to do some coding. You will need a pencil and a pad of RPG II coding sheets of each of these types: File Description, Input, Calculation and Output.

Chapter 1: Summary

You have been exposed to a number of RPG II rules that are important when coding solutions to data processing problems. Here's a list of those you should remember.

1. Each input file and each output file in a problem is described on a File Description sheet.

2. Each type of input file record and its fields is further described on an Input sheet.

3. Each type of output file record to be created is further described on an Output sheet. Output descriptions include fields of data and/or constants.

4. Each calculation is described on a Calculation sheet.

5. Numeric fields may be from 1 to 15 positions long with up to 9 decimal positions.

6. Alphameric fields may be from 1 to 256 positions long.

7. Filenames and Field names must start with a letter.

A number of definitions were also presented. You should remember these.

1. An alphameric field is a field that may contain either letters, numbers, special characters or blanks in any combination.

2. A numeric field is a field that may contain only numbers.

3. Overflow pages are all printed pages that follow the first page.

Chapter 2: Creating Disk File Records (Study time is 3 to 5 hours)

To "create" disk file records, information from source documents such as salesmen's order pads is stored onto the surface of the magnetic disk in an area set aside for that information. The usual procedure used to create disk file records on a System/32 disk is to key these records in one at a time from the console keyboard.

CHAPTER OBJECTIVES

When you have finished your study of this chapter you should be able to code a solution to a problem in which disk file records are to be created. You should also be able to code a program that causes these newly created records to be printed for reference.

Six problems will be presented in this chapter.

1. Create a file of disk records. The disk file is to have sequential organization.
2. Print a list of the file of disk records created in problem 1.
3. Create an indexed file of disk records.
4. Print a list of the file of disk records created in problem 3.
5. Print a list of the file of disk records created in problem 3, using the Auto Report feature of RPG II.
6. Create an indexed file of disk records and print a report of the keyed records.

In addition to this text you will need about 6 sheets of each of the following specification sheets.

GX21-9092	Control and File Description
GX21-9094	Input
GX21-9093	Calculation
GX21-9090	Output

A self test is included at the end of this chapter so that you can measure your progress.

Problem 1: Create a file of disk records. These disk file records are to be organized sequentially. The disk records shall each be 67 positions long and contain the following fields.

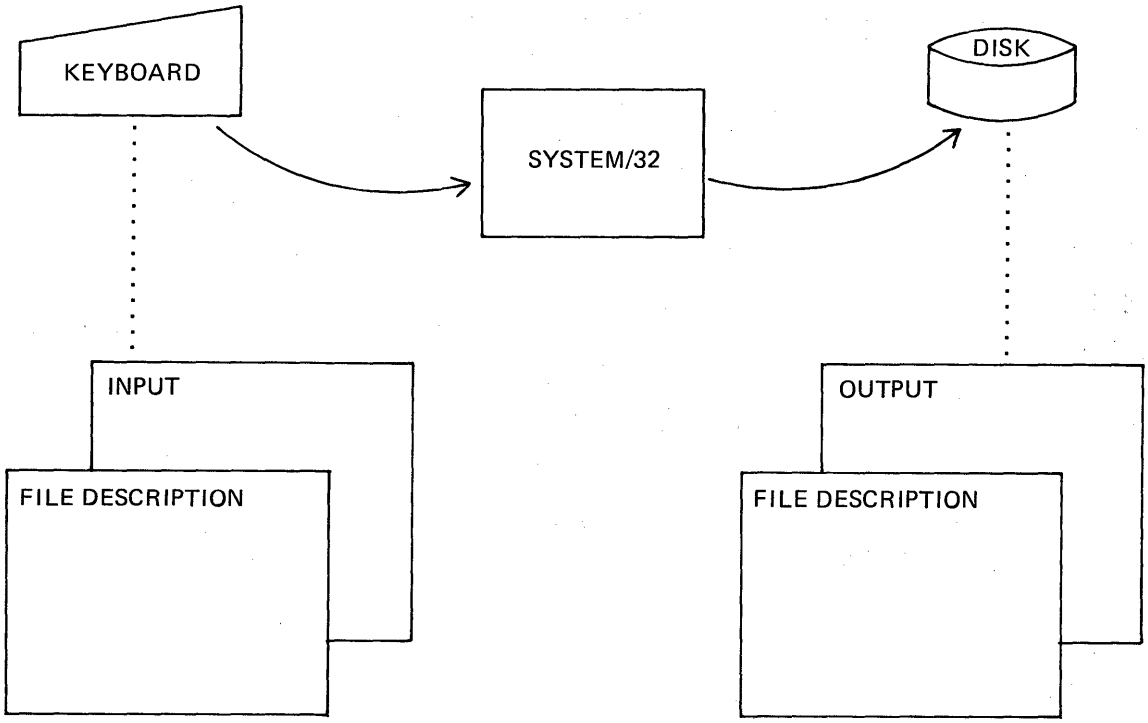
Disk Positions	Output Field
1	Code "M"
2- 7	Customer Number
8-27	Customer Name
28-47	Street Address
48-67	City/State Address

The information used to create these disk records is to be keyed in from a handwritten list. The fields of input data are identical in name but re-arranged as follows.

Keyed Positions	Input Field
1	Code "M"
2-21	Customer Name
22-41	Street Address
42-61	City/State Address
62-67	Customer Number

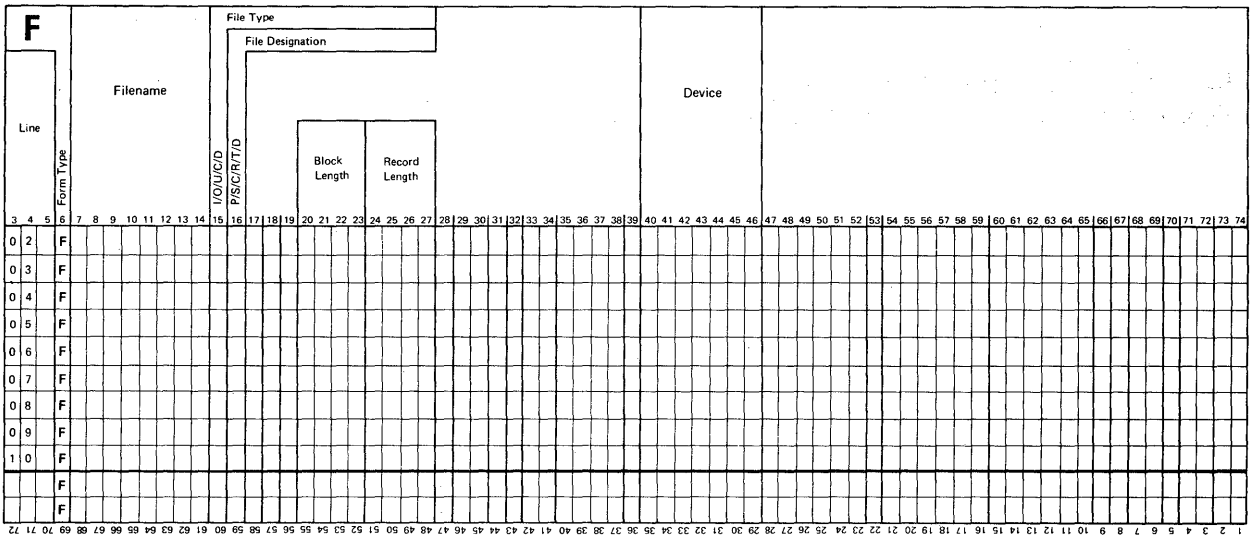
In order to code a solution to this problem we need to review some points. Look at a system flowchart to see the relationship between the problem and the RPG II specification sheets we will need to use in order to describe a solution.

Problem 1: Create a sequentially organized disk file.



Coding solutions in the RPG II language is done by describing files, records and fields. First we will describe each input and output file on a File Description specification sheet. This figure highlights the kinds of entries we need to make.

File Description Specification



Here is a list of rules for coding entries in the highlighted columns of the File Description Sheet.

1. Filename (7-14) - Make up a name of from 1 to 8 alphameric characters.

   a. The first character must be alphabetic.

   b. The remaining characters may be any combination of alphabetic characters and/or numbers.

   c. In RPG II, alphabetic characters include the letters A through Z, the # sign, the $ sign and the @ sign.

Examples:

```
X
T5
#628
INPUTFIL
MASTER
ART$$6
@M
```

2. File Type (15) - Use code I for input files or code O for output files.

3. File Designation (16) - Use code P for primary input files.

   a. No entry is made for output files.

   b. One and only one input file may be designated as "primary".

      NOTE: File designation entries are used to further identify the use of input files.

4. Block Length (20-23) - Enter a number that:

   a. describes the length of a block of disk records, or

   b. describes a block of space to be used for temporarily storing keyed data.

      NOTE: Enter numbers without leading zeros.

      NOTE: If no entry is made for a disk file, the block length is assumed to be equal to the record length entry.

5. Record Length (24-27) - Enter a number (without leading zeros) that:

   a. is the length of a disk record, or

   b. is the computed length needed to store one keyed record.

6. Device (40-46) - Enter the code name that identifies the input or output device that is associated with the particular file being described.

   a. Acceptable device names include:

      1) DISK for disk files.

      2) CONSOLE for keyed input files.

   b. The device name always starts in position 40.

You may refer to that set of rules as we code entries for our problem. Once again, the problem is:

> "Create a file of disk records. The disk file records are to be organized sequentially. The disk records shall each be 67 characters long and contain ..."

The remaining problem information indicated by the three dots will be used to describe records and fields later on.

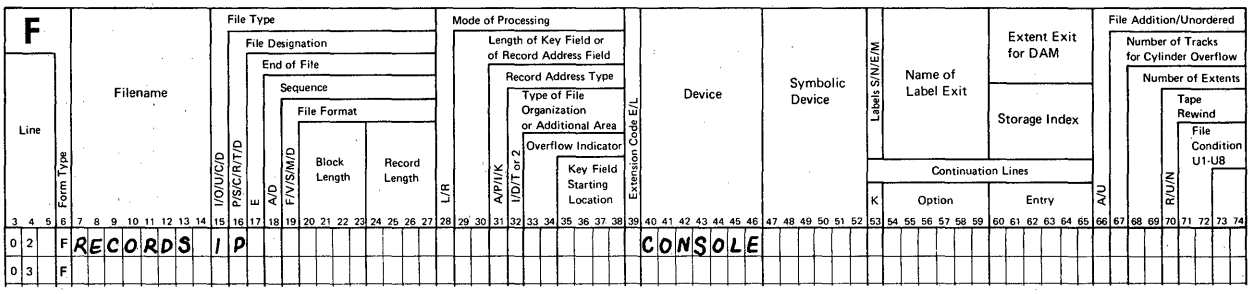
O.K. Let's make entries for the input file. Take a blank File Description Sheet and fill in entries for Filename, File Type, File Designation and Device. Look at the rules again if you wish.

* * *

The three asterisks mean that you are to interrupt your reading to do an activity such as coding. I did not ask you to fill in block length or record length entries as yet because they depend upon input sheet entries when input is keyed.

Here are my entries. Yours should be like mine except for filename which is your choice.

File Description Specification



We will make entries for block length and record length later on as I said. Now let's consider the output file. In any program, each filename must be different. Now, refer to the rules and the problem statement and then describe the output file on the second coding line.

* * *

Check your entries with these questions.

1. Did you make up a different filename?
2. Is your file type O for output?
3. Did you include a record length entry?
4. Is your device name DISK?

Now check your entries against mine.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/L/N	File Condition U1-U8
02	F	RECORDS	I	P											CONSOLE														
03	F	MASTER	O						67						DISK														
04	F																												
05	F																												

How's it going? If you have entries like these (except filename which is your choice) you are doing fine. If not, why not review the rules for describing files before we move along.

We'll add "Block Length" (20-23) and "Record Length" (24-27) for the CONSOLE file later on.

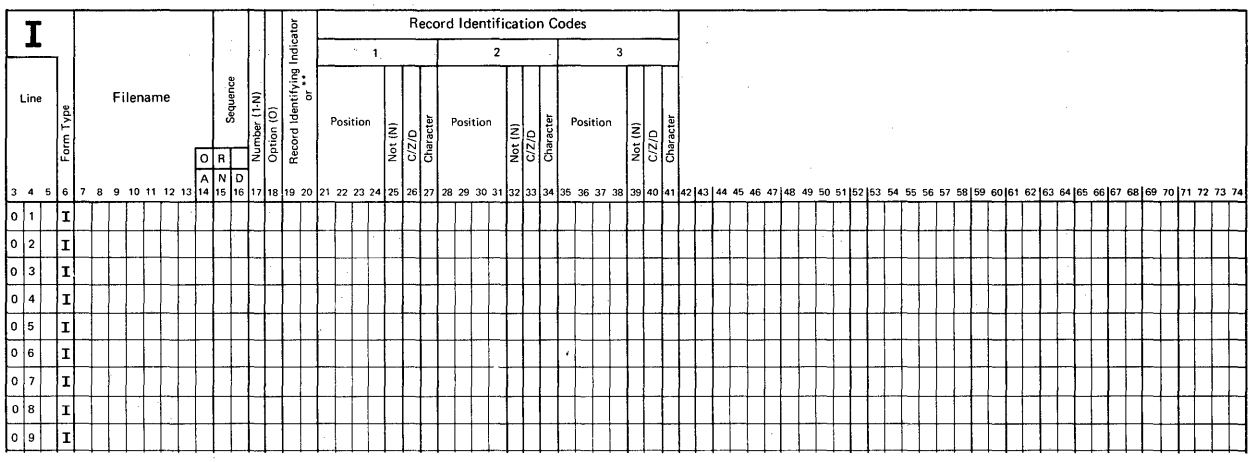
* * *

Now we are ready to describe the records and fields in the input file. Get a blank Input Sheet so that you will be ready to code some entries. This form is used for two purposes: use positions 7-41 to describe the record, and use positions 43-70 to describe the fields in the record. Position 42 is not used for System/32.

These two figures highlight the kinds of entries we need to make on an Input Sheet.

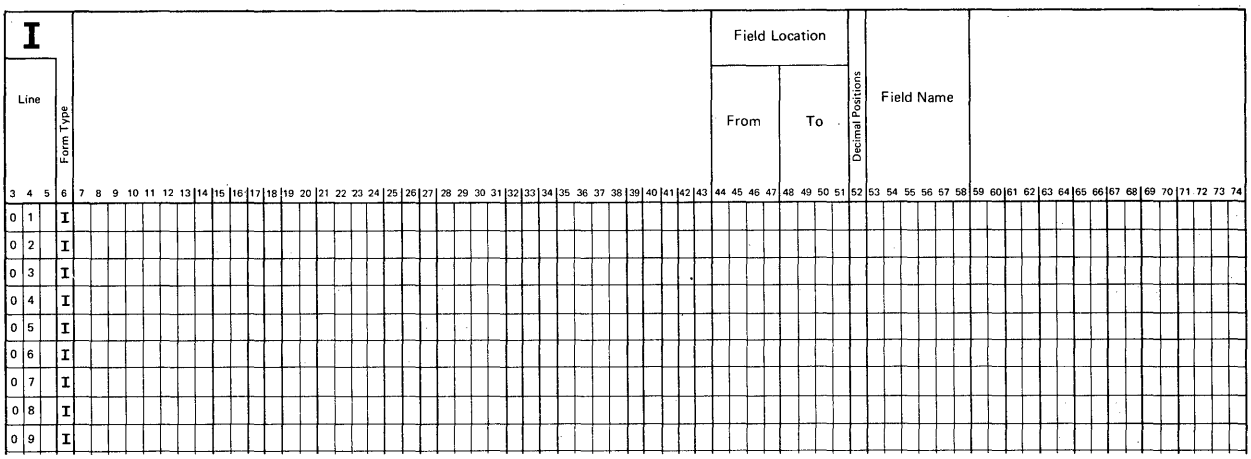
For Describing Input Records (Positions 7-41)

RPG INPUT SPECIFICATIONS



For Describing Input Fields (Positions 43-70)

RPG INPUT SPECIFICATIONS



Here are the rules for coding Input Sheet entries.

Rules: RECORD

1. Filename (7-14) - Copy the input filename from your File Description Sheet.

2. Sequence (15-16) - If all records in the file have the same kind of information, that is, there is really only one type of record in the file, enter any two letters in these positions.

3. Record Identifying Indicator (19-20) - Select any unused indicator from 01-10 for this entry. An indicator is a program activity controller and we will go into more detail about indicators throughout the course.

4. Record Identification Codes (21-41) - Enter one, two or three specifications to describe record identifying codes. In our problem one code (the letter M in position 1) identifies every record in the file.

   a. Position (21-24) - Enter a number without leading zeros to designate the position of the code character in the record.

   b. C/Z/D (26) - If the identifying code is any specific character, enter the letter C.

   c. Character (27) - Enter the actual code character.

   d. Make similar entries in the remaining positions if needed.

At this time code entries to describe the records of the Input File. Use the first coding line.

* * *

RPG INPUT SPECIFICATIONS

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	Record Identification Codes 1: Position (21-24)	1: Not (N) (25)	1: C/Z/D (26)	1: Character (27)	2: Position (28-31)	2: Not (N) (32)	2: C/Z/D (33)	2: Character (34)	3: Position (35-38)	3: Not (N) (39)	3: C/Z/D (40)	3: Character (41)	Stacker Select (42)	P/B/L/R (43)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Field Indicators Minus (67-68)	Field Indicators Zero or Blank (69-70)	(71-74)
01	I	RECORDS	NS			Ø1	1		C	M																					
02	I																														
03	I																														
04	I																														

Your entries should be similar to mine. Be sure that:

1. Your filename exactly matches the one you specified on the File Description Sheet.

2. Your sequence entry can be any two letters.

3. Your record identifying indicator can be any two-digit number from 01-10 since we haven't specified any of them up to this point.

   NOTE: We normally put a / through zeros on coding sheets. That prevents a mix-up between zeros and the letter O.

4. Your record identification code entries must be exactly as you see them here.

So far we've described the record, now let's describe the fields in that record.

Rules: FIELDS

1. Field Location (44-51) - Specify the leftmost (From) and rightmost (To) positions of the field.

   a. Leading zeros are not shown.

   b. If a field is only one position long, enter the same number in both places.

2. Decimal Positions (52) - Use only for a numeric field.

   a. For whole number fields, enter zero.

   b. For other numeric fields, enter the number of decimal positions.

   Example: For the number 2.657 enter a 3.

3. Field Name (53-58) - Invent a name having from 1 to 6 alphameric characters.

   a. The first character must be alphabetic.

   b. The remaining characters may be any combination of alphabetic characters and/or numbers.

   c. Alphabetic characters include A through Z, #, & and @ in RPG II.

   d. No special characters or embedded blanks may be used.

   e. Invent names that relate to the kind of data that is stored in the field. For example, use CUSNO for Customer Number.

4. Each field to be defined requires one coding line.

5. No field specifications may be entered on the record specification line.

Starting on the second coding line of your Input Sheet make entries for each of the following input fields.

Position	Field
1	Record Code
2-21	Customer Name
22-41	Street Address
42-61	City/State
62-67	Customer Number

* * *

Your entries should have been made on lines 2 through 6. The field locations must be the same as the book solution entries, but the names on your sheet are probably different and that's expected. Did you remember (or even think about) making an entry in position 52 for zero decimal positions in the Customer Number field?

RPG INPUT SPECIFICATIONS

I	Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
	01	I	RECORDS	NS			Ø1	1		C	M																				
	02	I																				1	1		CODE						
	03	I																				2	21		NAME						
	04	I																				22	41		STREET						
	05	I																				42	61		CITY						
	06	I																				62	67	Ø	CUSNO						
	07	I																													
	08	I																													

Earlier I said that we would make entries on the File Description Sheet for Block Length (20-23) and Record Length (24-27) for CONSOLE files. Now we have enough information to do this. It applies only to the Input File entry.

<u>Record Length - File Description</u> (CONSOLE files)

1. Look at the Input Sheet in positions 48 through 51 for the highest number.

2. Add a 1 to that number for each field you specified, except for a field that is the same as the record identifying code field.

3. Add 1 more for its record identifying code.

What's your answer? I calculated 72.

67 + 4 (don't count CODE) + 1 = 72

Block Length

1. Enter a number that is at least 2 more than the record length.

What's your answer? I got 74.

Now, complete your File Description Sheet entries for this console input file.

* * *

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	K	Option	Entry	A/U	R/U/N
02	F	RECORDS	I	P				74	72						CONSOLE							
03	F	MASTER	O						67						DISK							
04	F																					
05	F																					

RPG INPUT SPECIFICATIONS

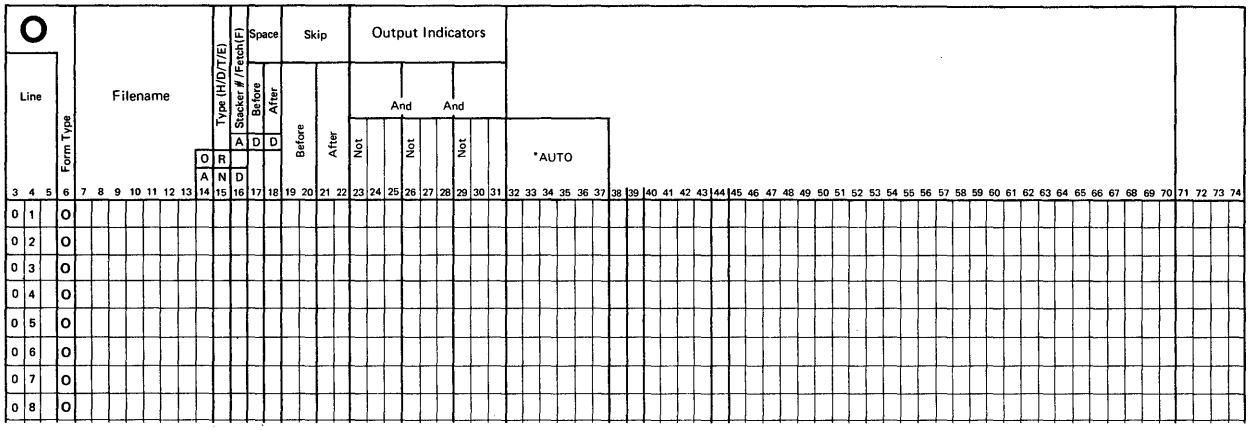
Line	Form Type	Filename	Sequence (AND/OR)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	RECORDS	NS			01		1		C																				
02	I																				1	1		CODE						
03	I																				2	21		NAME						
04	I																				22	41		STREET						
05	I																				42	61		CITY						
06	I																				62	67		0CUSNO						
07	I																													
08	I																													
09	I																													

We have described an input file, its records and fields. We have also described an output file. You should have entries for all of these items and be ready to describe the output records and fields next.

These next two figures highlight the two kinds of entries we need to make on an Output Sheet in order to describe output records and output fields.

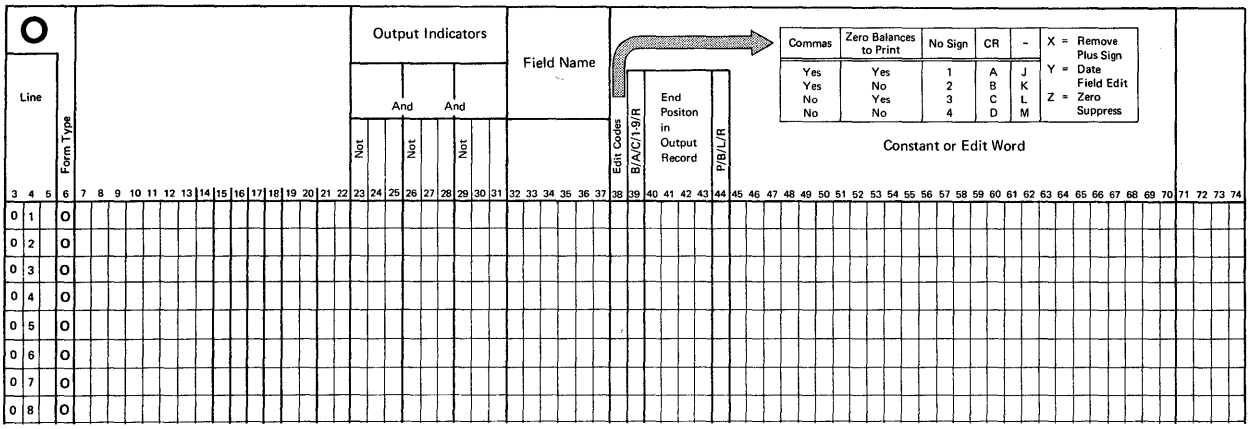
For Describing Output Records (Positions 7-37)

RPG OUTPUT SPECIFICATIONS



For Describing Output Fields (Positions 23-70)

RPG OUTPUT SPECIFICATIONS



Here are the rules for coding Output sheet entries.

Rules: RECORD

1. Filename (7-14) - Copy the output filename from your File Description Sheet.

2. Type (15) - Specify the type of output record desired. In our example, enter D for "detail" type because we want to create an output record for every input record.

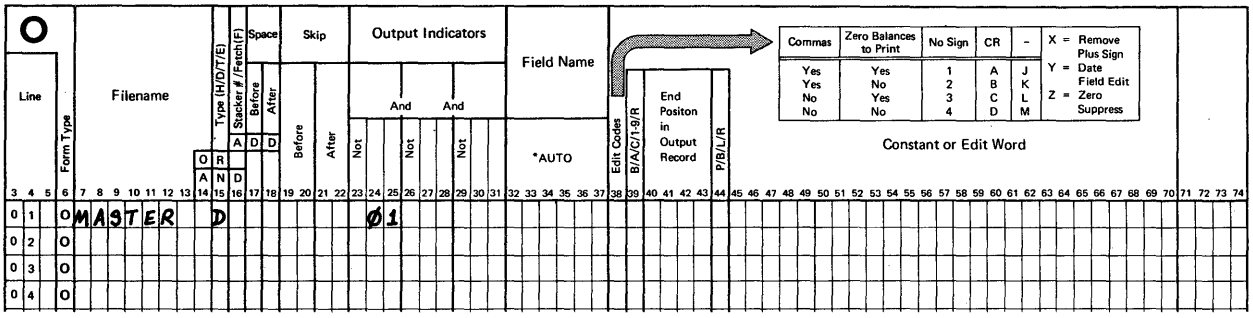
3. Output Indicators (23-31) - For detail output records copy the number of the indicator assigned on the Input Sheet. If additional conditions must be met in order to create this output record, specify the indicator(s).

4. When printing reports, make appropriate Space (17-18) and/or Skip (19-22) entries. None are needed in this particular example since no report is desired.

Specify your Output record description at this time.

* * *

RPG OUTPUT SPECIFICATIONS



Again, your entries should be similar to mine. Be sure that your filename and indicator are exactly like those you assigned on the other sheets.

The final step in coding a solution to this problem is describing the fields that make up the output record.

Rules: FIELDS - Specify one field per coding line.

1. Field Name (32-37) - Copy the names of each field needed for the output record from those you entered on the Input Sheet.

   a. Names start in position 32.

   b. For convenience, specify the fields in the order they are to appear in the output record.

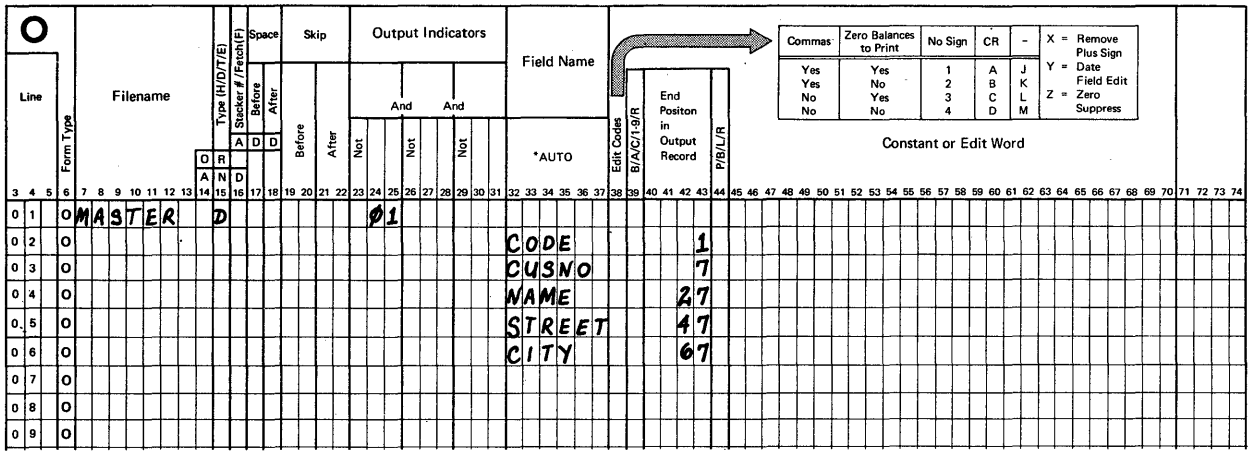
2. End Position in Output Record (40-43) - Specify the rightmost character's position for the field. For example, if a field named AMZ is to be located in positions 30-40 of the output record, specify 40 as the End Position. Omit leading zeros.

Here is the list of output field locations for our problem. Make all of your entries at this time.

Disk Positions	Output Field
1	Code "M"
2- 7	Customer Number
8-27	Customer Name
28-47	Street Address
48-67	City/State Address
*	* *

Your entries should be similar to those shown. Naturally, the field names must be exactly like those you specified as input fields.

RPG OUTPUT SPECIFICATIONS



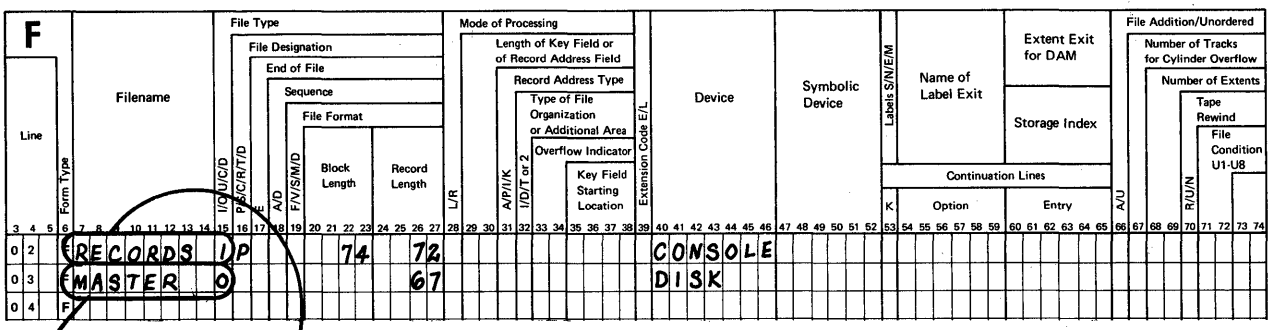
Now that we have coded all of the entries needed to describe the files, records and fields, I want to show you the relationships that these entries have to each other. Here's the problem statement. The total solution is shown on the next page. Take some time to note the relationships as shown by the connecting lines.

Create a file of disk records. These disk file records are to be organized sequentially. The disk records shall each be 67 positions long and contain the following fields. Input records contain this data in a different order, also shown.

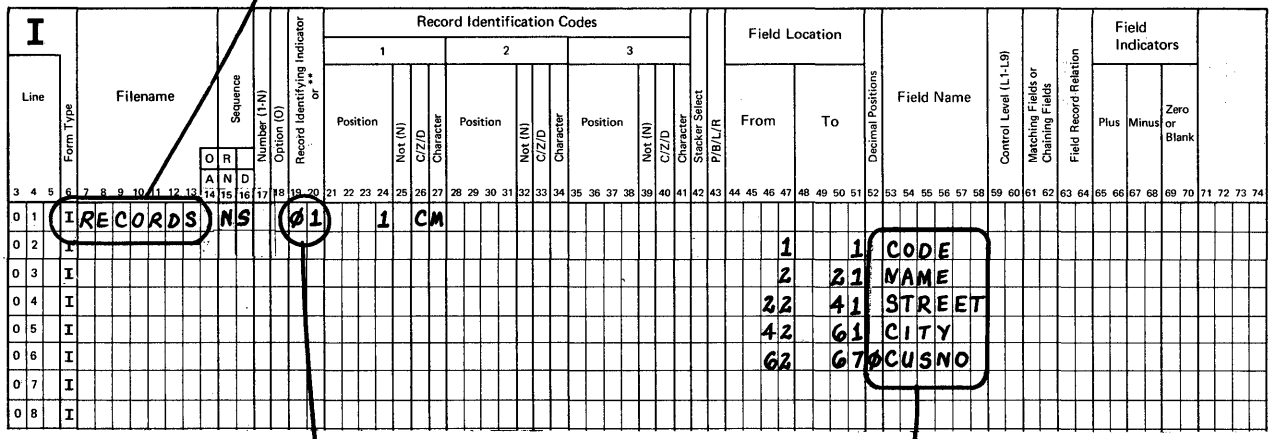
Disk Output Fields		Keyed Input Fields	
1	Code "M"	1	Code "M"
2- 7	Customer Number	2-21	Customer Name
8-27	Customer Name	22-41	Street Address
28-47	Street Address	42-61	City/State Address
48-67	City/State Address	62-67	Customer Number

Problem 1: Book Solution

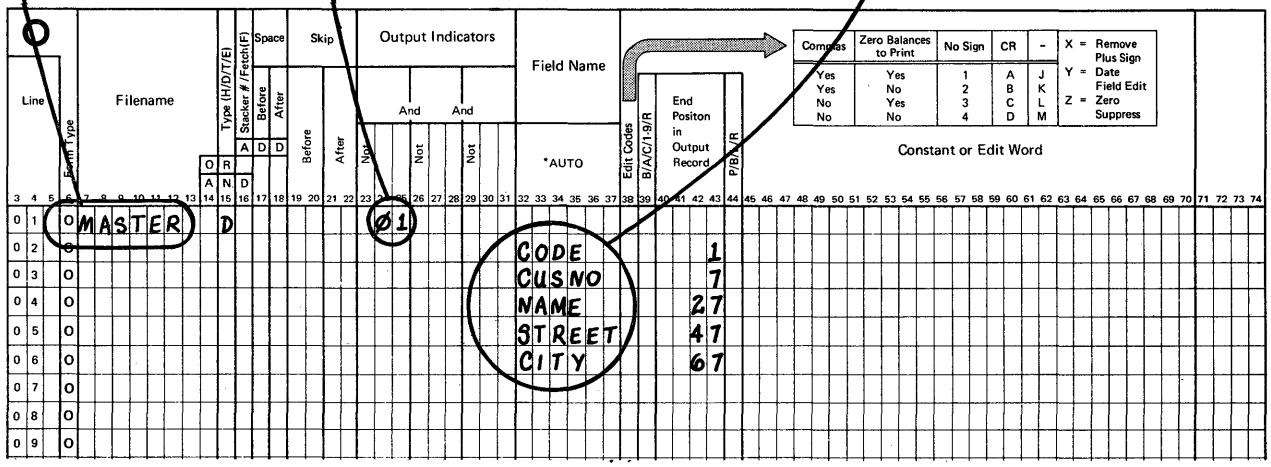
File Description Specification



RPG INPUT SPECIFICATIONS



RPG OUTPUT SPECIFICATIONS



Problem 1: Summary

Coding solutions to data processing problems using the RPG II language involves "describing" everything to be done.

1. Describe each file on the File Description Sheet.

2. Describe each input record type and its fields on the Input Sheet.

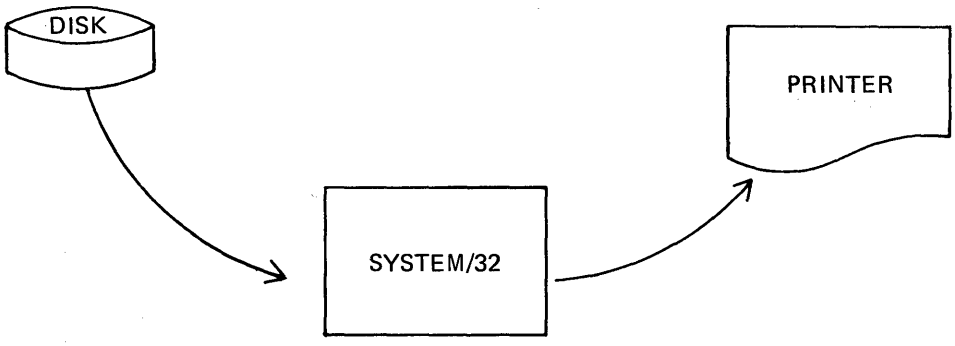
3. Describe each output record type and its fields on the Output Sheet.

4. Examine every entry on every sheet when you've finished all of the descriptions to assure yourself that all interrelated entries are spelled alike (or for indicators, exactly alike).

We will be doing 5 more problems in this chapter. The rules you have learned apply to each one. You will also learn new rules when needed to describe new situations.

Set aside your solution to problem 1 and get some blank forms so we can code a solution to this one.

Problem 2: Print a listing of the records in the sequential disk file created in problem 1.



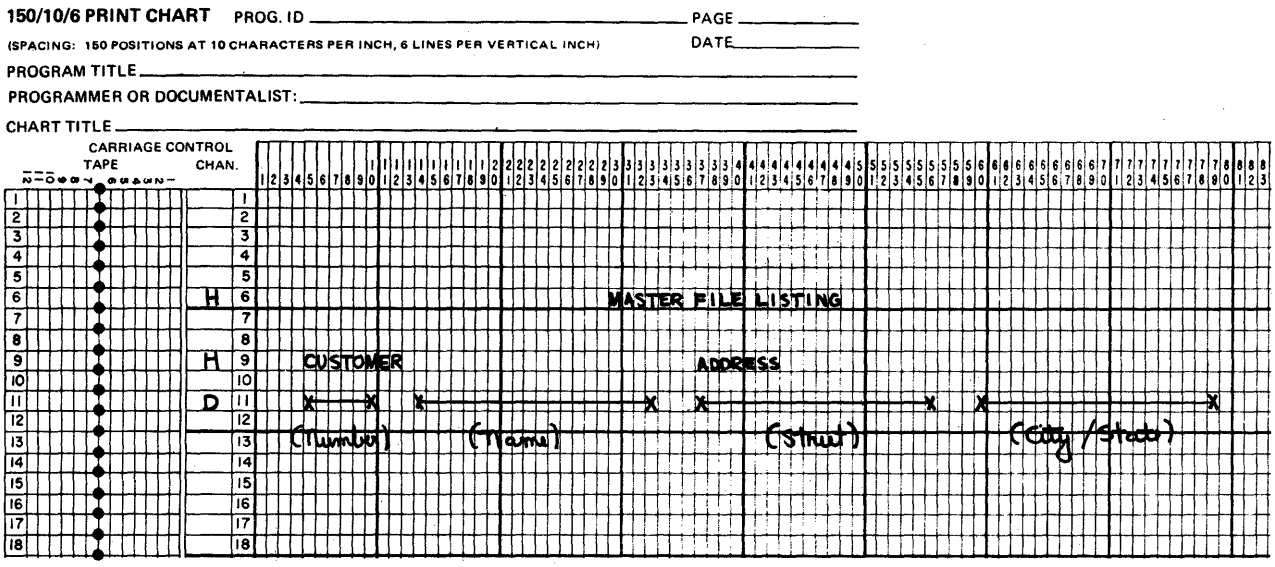
The disk file has 67-character records including a code "M" in position 1, customer number in 2-7, customer name in 8-27, street address in 28-47 and city/state address in 48-67.

The report that lists the records is to include headings as shown here.

MASTER FILE LISTING

CUSTOMER		ADDRESS		
234567	THE RIBBON COMPANY	12 POST OFFICE DRIVE	SHIPER, AX	10110
240011	OILEY BURD	HIGHWAY 97	WALON, OJ	77774
252578	HOT SUMMER'S DAE	120 SHADEE LANE	TUNDRA, KL	90903
378941	SELLFAS PIPE CLEANSO	BOX 1012023035647883	LKJHGFDSA, CM	00001

A print chart showing the desired record and field layouts is also included. Look at it briefly and then continue. We will refer to it again when coding output sheet entries.



At this time code all of the File Description entries and the Input entries needed to describe the input disk file and its records. Refer to the rules presented with the first problem if necessary. Use MASTER as the input filename for this example.

* * *

The input file and its fields should be specified as shown here. You may have different field names and a different record identifying indicator and you may have assigned two different letters for sequence, but all other entries must match mine.

RULE: Indicators 01-99 may be assigned to identify DISK file input records.

RULE: Indicators 01-10 may be assigned to identify CONSOLE file input records.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	MASTER	I	P					67							DISK													
03	F																												
04	F																												

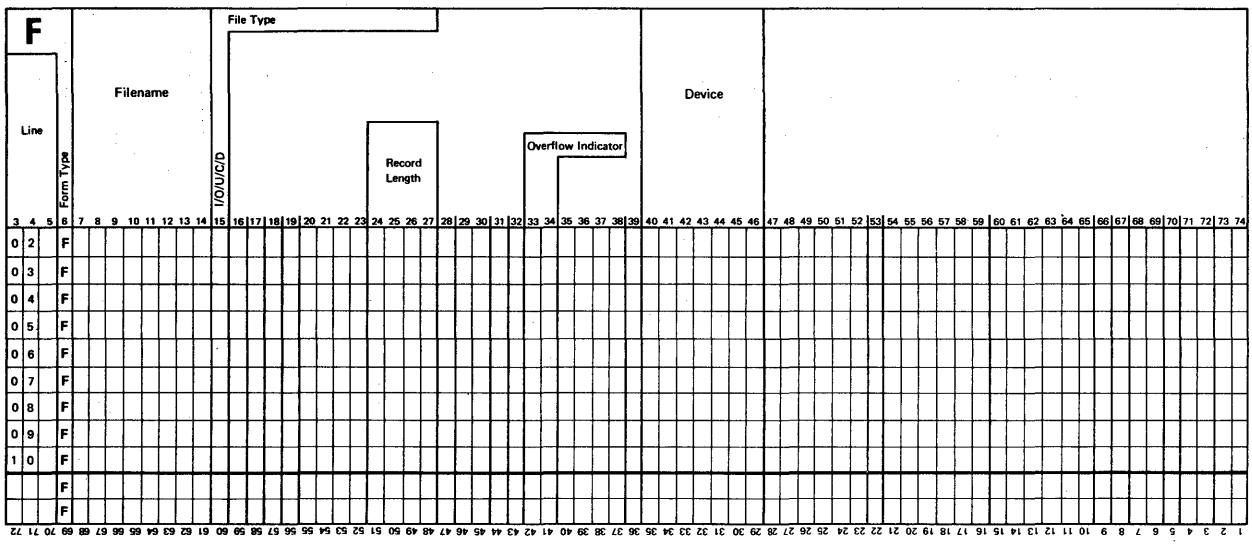
RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	MASTER	AA			Ø1	1		CM																					
02	I																				2	7		ØC#						
03	I																				8	27		CNAME						
04	I																				28	47		ST						
05	I																				48	67		CS						
06	I																													
07	I																													
08	I																													

Now we are ready to describe the output file. In order to describe a printed report as an output file, follow these rules. When you have read them, make the File Description entries that describe the report file.

Here's a figure highlighting the entries needed to describe the output print file.

File Description Specification



Rules: Print Files

1. Filename (7-14) - Invent a name having from 1 to 8 alphameric characters.

2. File Type (15) - Enter O for output.

3. Record Length (24-27) - Enter 132. On System/32, print files contain lines (records) of 132 positions.

4. Overflow Indicator (33-34) - Enter one of the following codes: OA, OB, OC, OD, OE, OF, OG or OV. It is used to control printing when a page is full.

5. Device (40-46) - Enter PRINTER as the device name.

Now specify the print output file.

* * *

I used LIST as a filename and selected the code OF for an overflow indicator. Here are my entries.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
	F																									
02	F	MASTER	I	P					67							DISK										
03	F	LIST	O						132				OF			PRINTER										
04	F																									
05	F																									

Continuation Lines: K | Option | Entry | A/U

Next we will examine the report to determine what "types" of output records are shown. A report may contain 3 types:

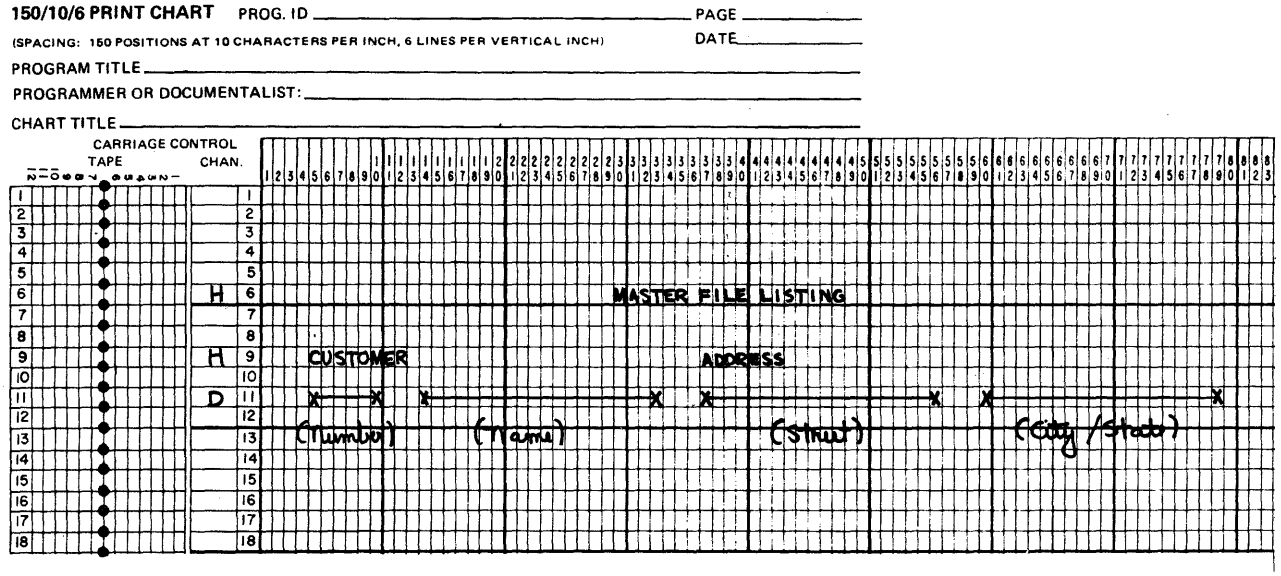
1. Headings - Report title or column headings, and/or
2. Details - Lines of input record data, and/or
3. Totals - Lines that contain some totals from information in detail records.

As you examine the sample report try to determine how many different types of output records are shown.

MASTER FILE LISTING

CUSTOMER		ADDRESS		
234567	THE RIBBON COMPANY	12 POST OFFICE DRIVE	SHIPER, AX	10110
240011	OILEY BURD	HIGHWAY 97	WALON, OJ	77774
252578	HOT SUMMER'S DAE	120 SHADEE LANE	TUNDRA, KL	90903
378941	SELLFAS PIPE CLEANSO	BOX 1012023035647883	LKJHGFDSA, CM	00001

I noted 2 heading record types and 1 detail record type. There are no totals on this particular example. A print chart is used to show both the type of line (record type) and the fields or constants such as titles they contain.



You noticed that we represent fields of data with an X at either end connected by a straight line. Also, we show the heading information exactly as it shall be printed.

We are ready to describe these record types, their fields and their constants on an output sheet. Here are the rules you need to refer to.

Rules: RECORD

1. Filename (7-14) - Copy the output filename from the File Description sheet.

   a. When two or more record types are described in the same file, the filename is only required for the first record type.

2. Type (15) - Specify code H for headings or code D for details.

3. Space (17-18) - Specify the number of lines to be spaced either before or after printing.

   a. Acceptable entries are 0, 1, 2 and 3. A 1 produces single spacing; 2 produces double spacing and 3 produces triple spacing. When zero is specified, no spacing occurs.

   b. An entry may be made in both positions 17 and 18.

4. Skip (19-22) - Specify the line number to which paper shall be skipped either before or after printing occurs.

   a. Acceptable entries are 01-84.

   b. An entry may be made both for skipping "before" printing and skipping "after" printing.

   NOTE: If no entry is made in columns 17-18 or 19-22, printing will automatically be single spaced.

   NOTE: If entries are made in all four places for one line, the action will occur in this order:

   First - Skip Before
   Second - Space Before
   Third - Skip After
   Fourth - Space After

5. Output Indicators (23-31) - Specify the indicator that shall control "when" printing, spacing and/or skipping shall take place.

a. To print a heading on the first report page made up of constants but no fields, specify indicator 1P (first page).

At this time code the entries needed to describe the first heading record type for the report.

* * *

Here's my solution that provides for controlling the printing of the H line.

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Field Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	LIST	H			3	Ø6		1P					
02	O													
03	O													
04	O													
05	O													

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Where did I get the entries "3" for space after (18) and "Ø6" for skip before? If you said, "From the print chart", that's exactly right. The chart shows that the report line shall print on line 6 and the next line of printing is to be line 9.

The coding just shown provides for printing a title on the first page only. To print the same title on all other pages, I need to control it by using the overflow indicator assigned in positions 33-34 on the File Description sheet. I use a special RPG II output statement called an "OR" line. Here are some rules you should know when coding such a statement.

Rules: <u>OR output line</u>

1. OR (14-15) - Specify the word OR in these positions on the next coding line when an output record is to be created if either of 2 conditions exists.

2. Space and Skip (17-22) - When special spacing and/or skipping is to be associated with the OR line, specify it. If spacing and skipping is to be exactly the same as already specified on the previous coding line, do not make entries here.

3. Output Indicators (23-31) - Specify the indicator that shall control the OR line.

At this time specify an "OR" line to be printed when your overflow indicator is used.

* * *

Here's the coding that will print a title on the first page (1P control) and on all additional pages (OF overflow control) of the report. If you used a different overflow indicator it must be the one you specified on the File Description sheet earlier.

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Output Indicators Not	Field Name	Edit Codes	End Positon in Output Record	P/B/L/R	Constant or Edit Word
01	O	LIST	H			3	06				1P					
02	O		OR								OF					
03	O															
04	O															

Commas: Yes, Yes, No, No — Zero Balances to Print: Yes, No, Yes, No — No Sign: 1, 2, 3, 4 — CR: A, B, C, D — -: J, K, L, M

X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress

Well, we now have specified two lines of code to describe the record type (H), the desired lines (skip before to Ø6, space 3 after), and when this line shall be printed (1P or OF). What's next? Now we need to specify the actual report title. This is done by describing a constant on the next coding line. Here are the rules to follow.

Rules: Output Constant

1. End Position in Output Record (40-43) - Specify the location of the rightmost character in the constant.

   a. Refer to a print chart.

   b. Omit leading zeros.

2. Constant (45-70) - There are 5 points to remember.

   a. Specify a single quote mark in position 45.

   b. Specify the exact characters starting in position 46.

   c. Specify a single quote mark in the position after the last character of the constant.

   d. Use one space for each character.

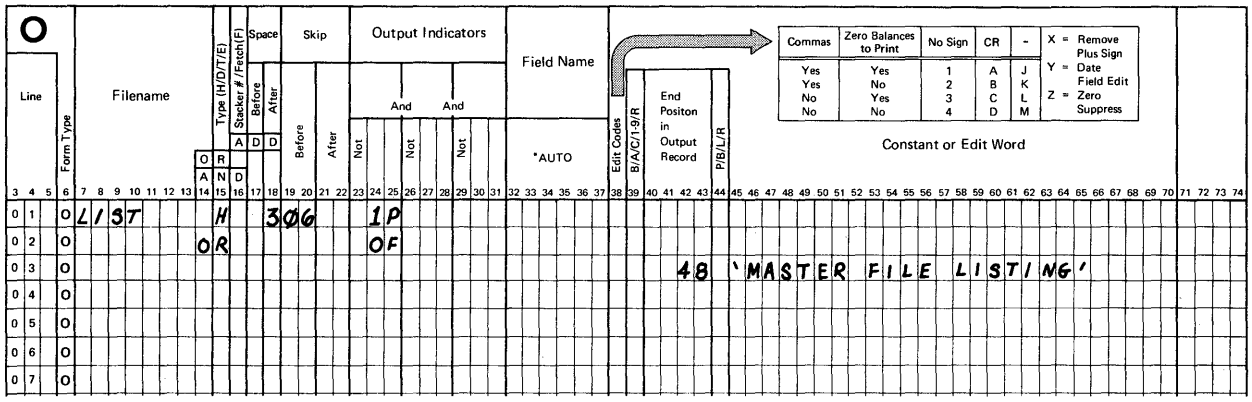
   e. Any character may be part of a constant.

At this time specify the report title as a constant on your third coding line. Refer to the print chart to determine the End Position in Output Record entry.

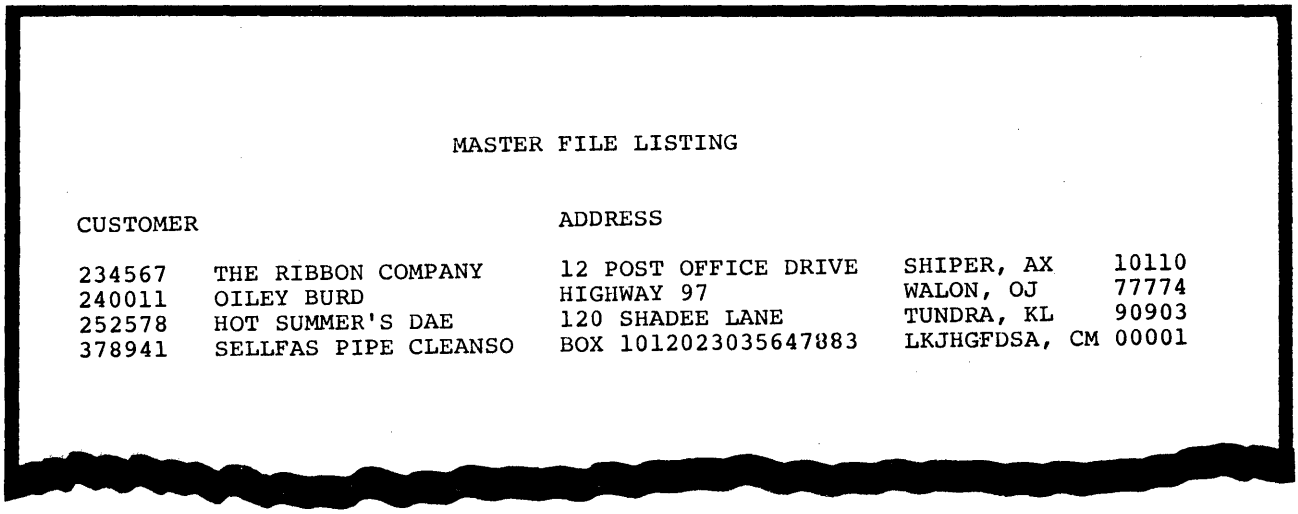
* * *

Here's my solution. Is yours the same? If not, re-examine the print chart before continuing.

RPG OUTPUT SPECIFICATIONS



Well, that satisfies our requirement to print the report title on every page of the report. What's next?



MASTER FILE LISTING

CUSTOMER		ADDRESS		
234567	THE RIBBON COMPANY	12 POST OFFICE DRIVE	SHIPER, AX	10110
240011	OILEY BURD	HIGHWAY 97	WALON, OJ	77774
252578	HOT SUMMER'S DAE	120 SHADEE LANE	TUNDRA, KL	90903
378941	SELLFAS PIPE CLEANSO	BOX 1012023035647883	LKJHGFDSA, CM	00001

There are two constants (CUSTOMER and ADDRESS) to be printed as a second heading record.

1. Print this type of record on every page also.
2. Double space after printing it.
3. Specify each constant as a part of this record.
4. Refer to a print chart to determine End Position.

Try to code all of the entries needed to describe this heading line.

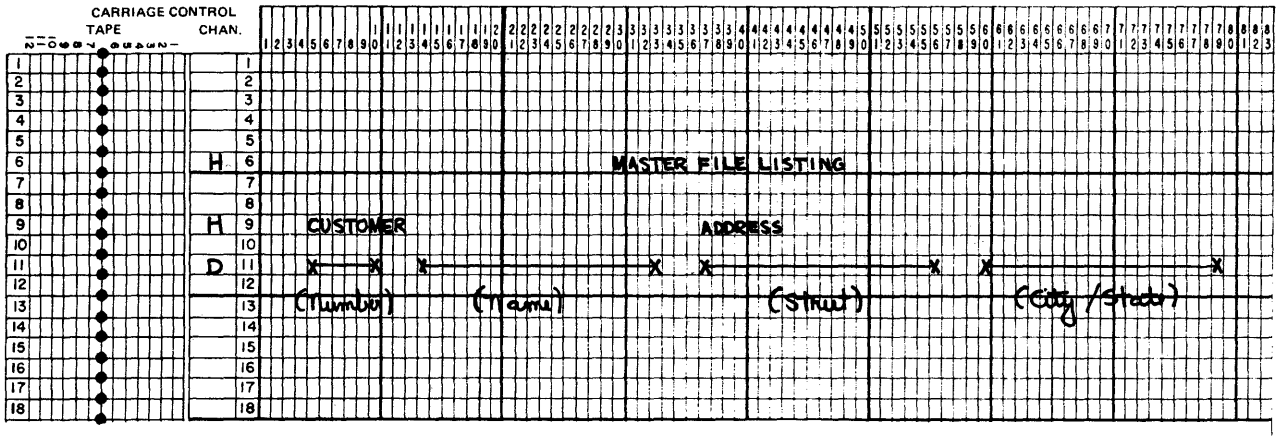
150/10/6 PRINT CHART PROG. ID _____ PAGE _____

(SPACING: 150 POSITIONS AT 10 CHARACTERS PER INCH, 6 LINES PER VERTICAL INCH) DATE _____

PROGRAM TITLE _____

PROGRAMMER OR DOCUMENTALIST: _____

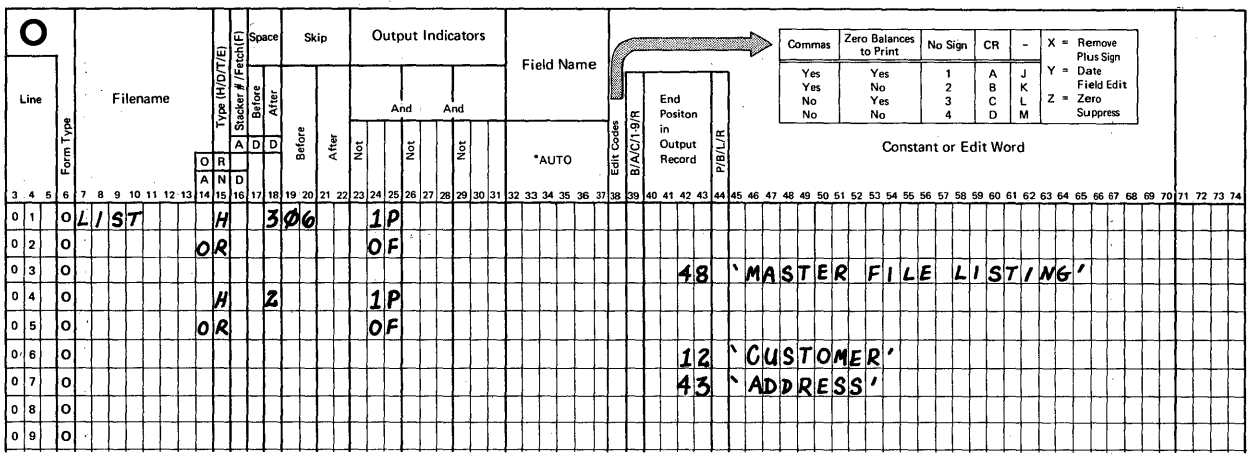
CHART TITLE _____



* * *

Check your entries against the book solution. Did you recall that you don't need to specify the filename since it was already specified for the first heading record?

RPG OUTPUT SPECIFICATIONS



In order to describe the remainder of this report we need to examine both the print chart and the sample report page.

1. What type of record is to be described?
2. What spacing is required?
3. How many fields or constants are a part of this record?
4. "When" shall it be printed?

From the print chart we see that the type of record to be described is detail (D on line 11). A detail output record is produced for every detail input record.

The report shows that detail lines are single spaced.

The print chart shows 4 fields, no constants.

A detail line is to be printed for each input record, that means control printing by using the record identifying indicator assigned on the Input sheet as your output indicator on the Output sheet.

At this time code all of the entries you think are needed to describe the detail line.

* * *

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	LIST	H		3	0	6		1P							
02	O		OR						OF							
03	O													48		'MASTER FILE LISTING'
04	O		H		2				1P							
05	O		OR						OF							
06	O													12		'CUSTOMER'
07	O													43		'ADDRESS'
08	O		D		1				01							
09	O										C#			10		
10	O										CNAME			33		
11	O										ST			56		
12	O										CS			79		
13	O															
14	O															

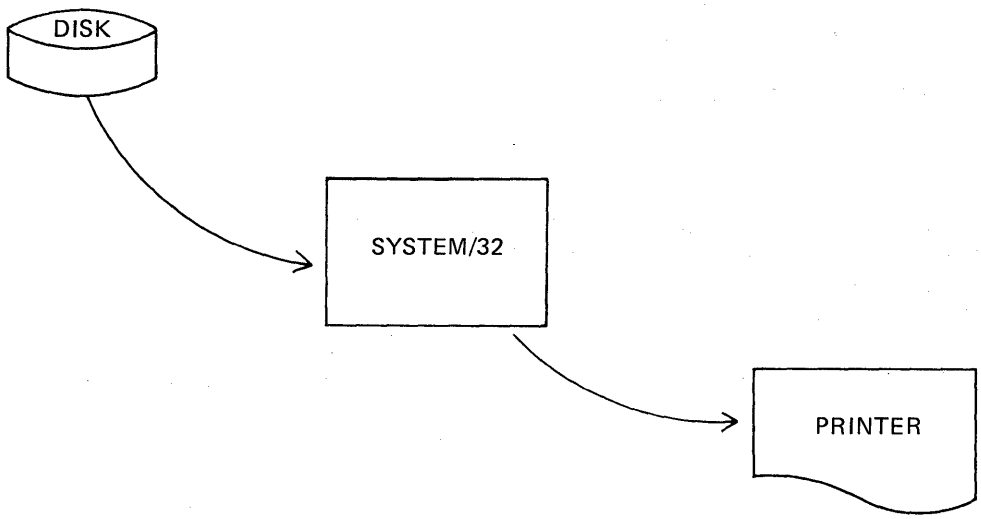
Commas: Yes, Yes, No, No | Zero Balances to Print: Yes, No, Yes, No | No Sign: 1, 2, 3, 4 | CR: A, B, C, D | -: J, K, L, M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Your entries should be similar to mine. If you intentionally left position 18 (Space After) blank, that's fine because there are no other entries for positions 17-22. Therefore, the detail lines will automatically be single spaced.

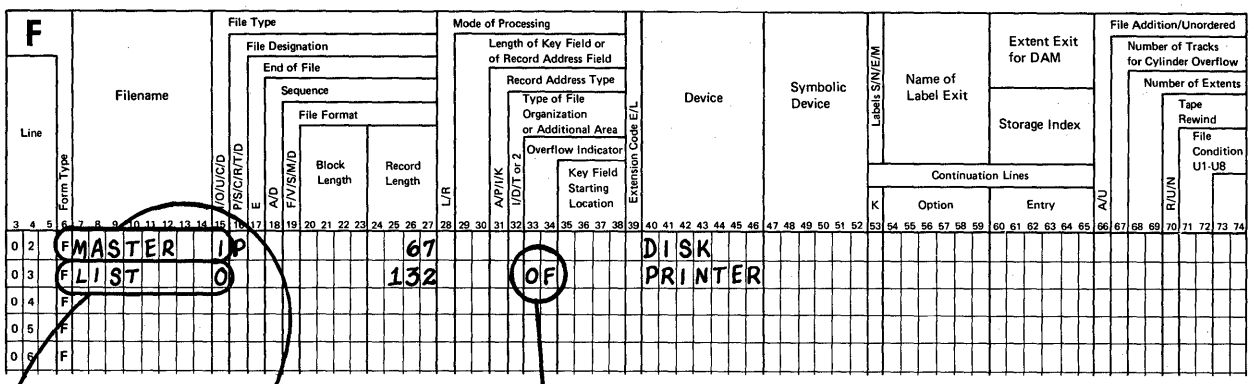
For a review of problem 2, here's a system flowchart and the problem statement. A completely coded solution is shown on the next page. Note again the interrelated entries.

Problem 2: Print a listing of the records in the sequential file created in problem 1.

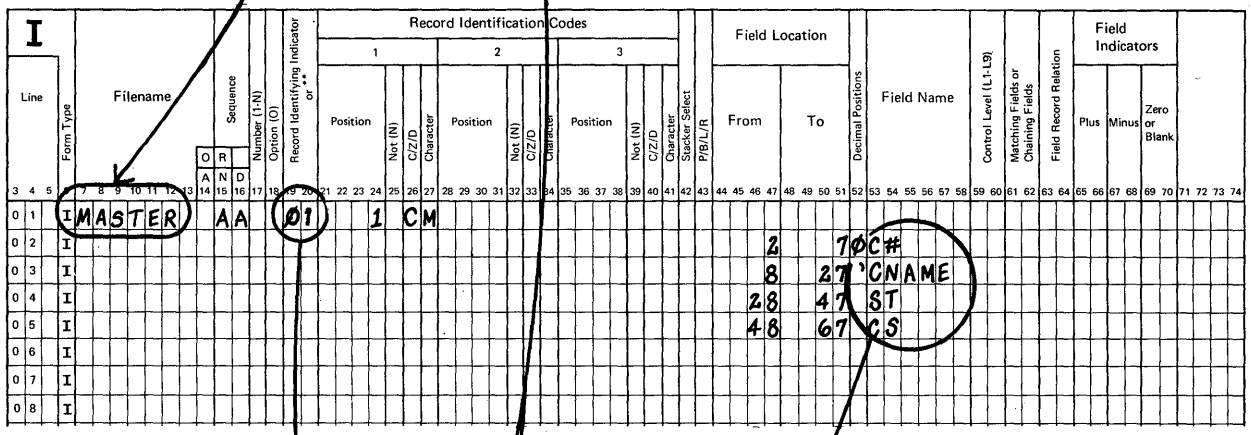


Problem 2: Book Solution

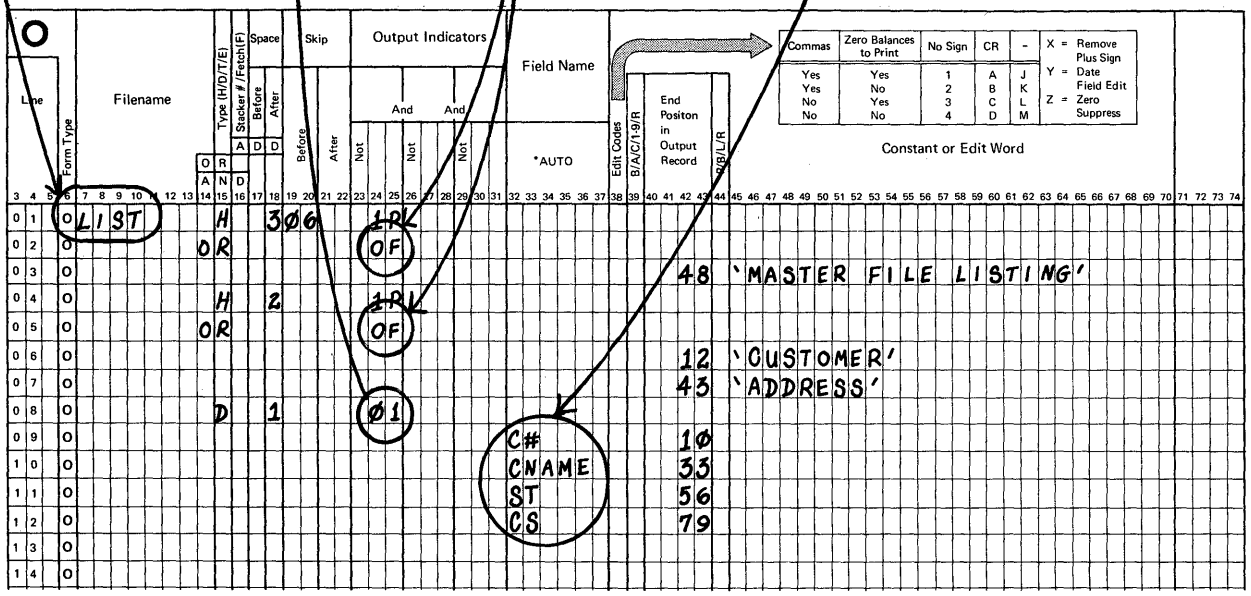
File Description Specification



RPG INPUT SPECIFICATIONS



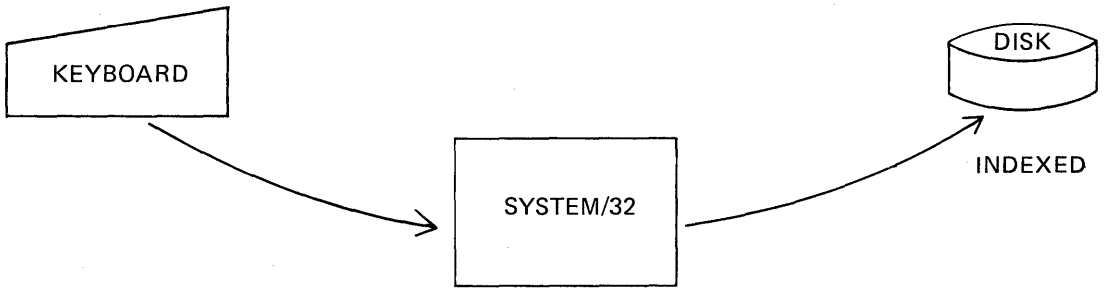
RPG OUTPUT SPECIFICATIONS



Problem 3:

Problem number one involved the creation of a file of disk records organized sequentially. For problem number two we coded a solution that involved printing the contents of the sequential disk file. Here's problem three.

> "Create a file of indexed disk records. Input records are to be keyed in from the console".



The input file contains two kinds of records. The very first record is a file creation date record. After it has been keyed in, all the regular transaction records are keyed. A "key field" is found in the transaction records. It will be used by RPG II to build the index when the disk file is created.

KEYED RECORDS - 2 types

1. first record

Position	Field
1	Code letter "D"
2 - 7	Creation date

2. transactions

Position	Field
1 - 2	Code characters "X6"
3 - 6	Account # (key field)
7 - 25	Name
26 - 31	Beginning balance, two decimals

DISK RECORDS

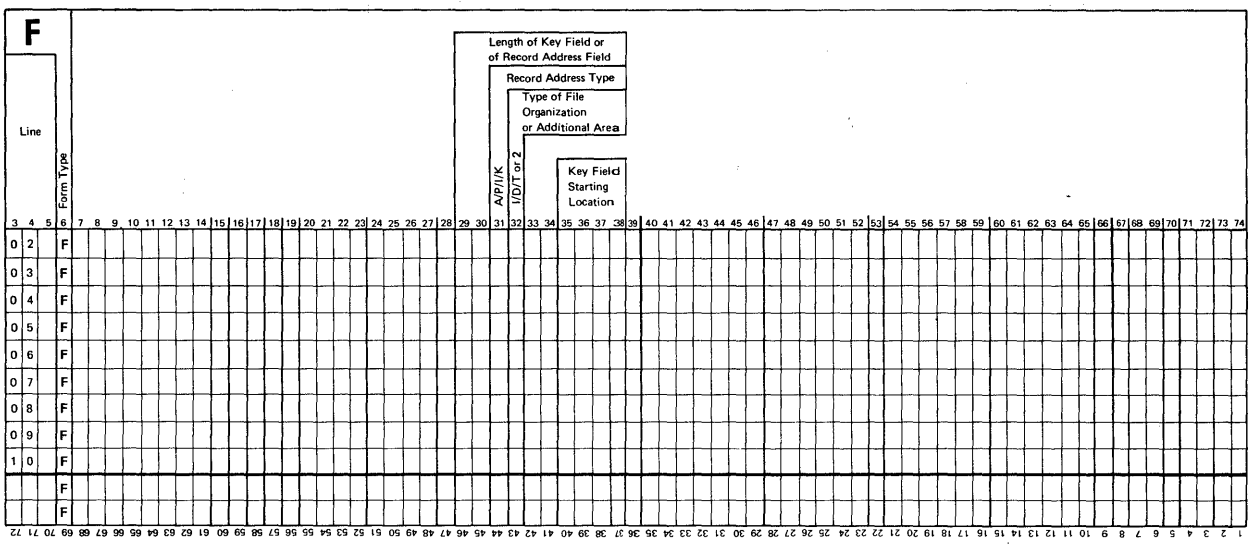
The output file records are to contain these fields.

Position	Field or Constant
1	Character "B"
2 - 5	Account number (key field)
6 - 11	Beginning balance
12 - 17	Creation date
31 - 49	Name

Each output record is to be 75 positions long.

When an indexed file is described on the File Description, we need to include information shown in the highlighted areas.

File Description Specification



Your first step is to describe the input file and the output file on a File Description sheet.

Here are some new rules for reference when describing an indexed file.

Rules: INDEXED FILES

1. Length of Key Field (29-30) - Specify the length of the field that is to be used as the key field.

   a. Omit the leading zero.

   b. The maximum length of a key field is 29 characters if the data is alphameric.

   c. The maximum length is 15 characters if it is a numeric field.

2. Record Address Type (31) - Enter code "A". This code is used when processing or creating (loading) indexed files.

3. Type of File Organization (32) - Enter code "I" to designate Indexed file organization.

4. Key Field Starting Location (35-38) - Specify the starting location of the key field in the disk record.

   a. Starting location is the leftmost position.

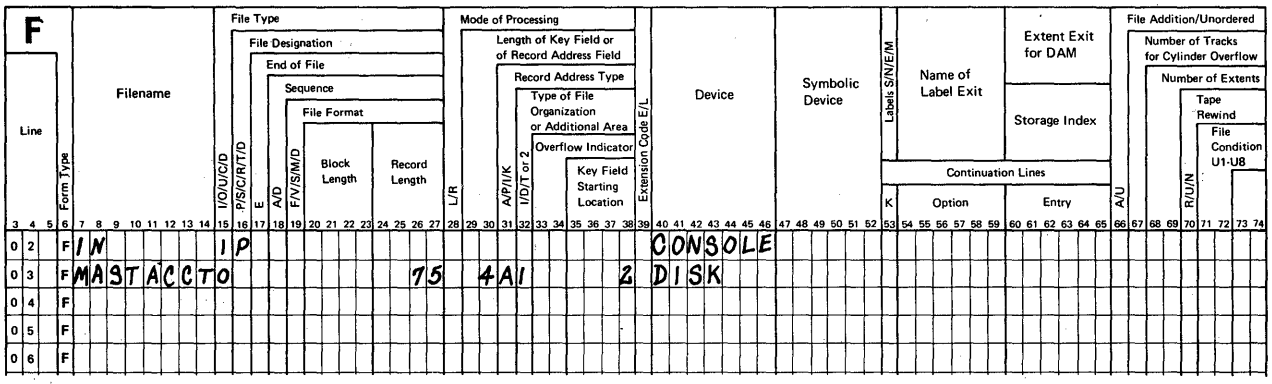
   b. Omit leading zeros.

Describe both the input and the output files at this time. For the moment, ignore the block length and record length entries for the keyed input records.

* * *

Your entries should be similar to the ones shown. If you had an error in columns 29-32 or 35-38 re-read the coding rules and make corrections before continuing.

File Description Specification

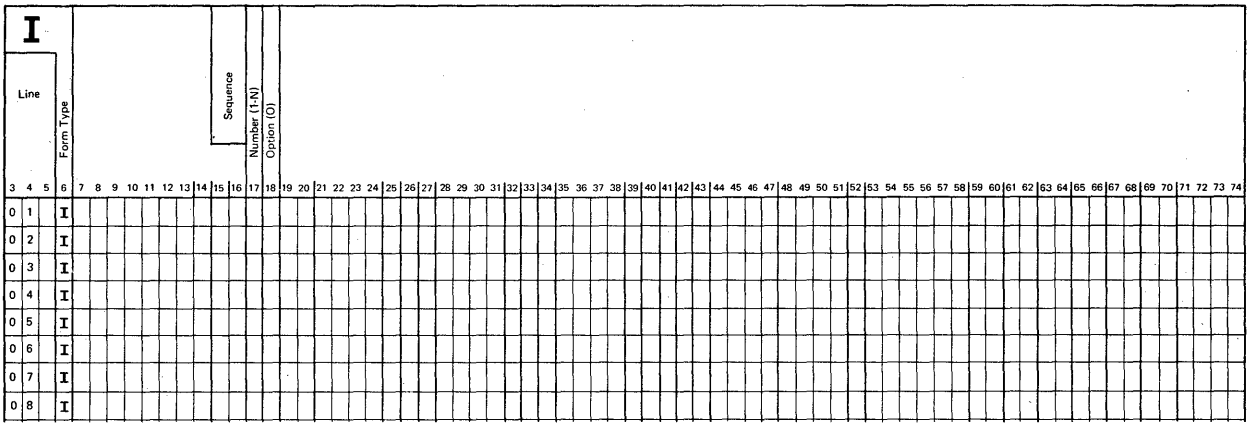


Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	INP	I	P													CONSOLE														
03	F	MASTACCTO							75		4	A	I			2	DISK														
04	F																														
05	F																														
06	F																														

In this particular problem, the very first keyed input record is different from all remaining keyed records. The first keyed record is the file creation date record. RPG II permits the use of multiple record types in any input file.

Before we describe each of these input record types consider this point. If the file creation date record is not keyed in ahead of all transactions used to create the file, the output records will not include this field.

RPG II can assist the programmer in making sure that the type of record that must be keyed in ahead of all others is actually keyed in. A numeric entry under "Sequence" in position 15-16 of the Input sheet takes care of that requirement. To further describe the conditions we will be making entries in the highlighted positions shown.



Now for the rules. To identify two or more input record types that must be keyed in (or read in from disk) in a particular order, make these entries.

1. Sequence (15-16) - Enter "Ø1" to identify the first record type.

   a. Enter Ø2 for the second record type. Enter Ø3 for the third record type, and so on.

   b. Gaps in sequence numbers are allowed, but the numbers used must be kept in ascending order. The first number must be Ø1.

2. Number (17)

   a. Enter a "1" if one, and only one record of this type is to be keyed in or read in as input.

   b. Enter the letter "N" if any number of records of this type will be keyed in or read in as input.

   c. When there may be one record or more than one record of a particular type, use "N" as the entry.

3. Option (18) - Enter the letter O if this record type may or may not be present.

   a. If position 18 is left blank, at least one record of the type being described must be keyed in or read in as input.

I've repeated information about the input records for problem 3 so that you can refer to it as you describe each type. Remember to describe a record type on one line of the Input sheet and then use the next line to describe a field. At this time code all of the Input entries you feel are needed to complete the description of the input file.

NOTE: Filename is only entered for the first record type in the file.

Input Record	Positions	Field
First	1	Letter "D"
	2 - 7	Creation date
All Others	1 - 2	Code characters "X6"
	3 - 6	Account # (key field)
	7 - 25	Name
	26 - 31	Beginning balance, 2 decimals

* * *

Examine your entries.

1. The first line should describe the first record type.
2. The second line should describe the field in that record.
3. The third line should describe the second record type.
4. Lines 4, 5 and 6 should describe the fields in the transaction records.

Now look at the book solution. If your answers are not the same in positions 15-18, re-read the rules and correct your entries.

RPG INPUT SPECIFICATIONS

I

Line	Form Type	Filename	Sequence (O A / R N / D)	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	IN	Ø1	1		Ø1		1	C	D																				
02	I																				2	7	Ø	CRDATE						
03	I		ØZN			Ø2		1	C	X		2	C	6																
04	I																				3	6	Ø	ACCT						
05	I																				7	25		NAME						
06	I																				26	31	2	BEGBAL						
07	I																													
08	I																													

As coded here, the description of the input file's records means:

1. first, the operator must key in a creation date record, and

2. next, the operator must key in <u>at least one</u> transaction record.

If this is not done, the program will automatically stop to prevent the creation of disk records without a real creation date.

You should now finish the entries needed to describe the console file - record length and block length. Do you recall the rules? If so, good. If not, they are repeated for your convenience.

Rules: <u>Console File</u>

1. Record Length (24-27) - Take the value from the record type with the highest end position specified on the Input sheet, and add one. Add one for each field specified in that record type except the record identification field (if it is specified as a field).

2. Block Length (20-23) - Enter a number that is at least two more than the calculated record length.

* * *

Here is a File Description with all desired entries for your reference.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	R/U/N	Tape Rewind	File Condition U1-U8
02	F	IN	I	P				37	35						CONSOLE												
03	F	MASTACCTO							75		4A1		2		DISK												
04	F																										
05	F																										
06	F																										

There are no new entries needed on the Output sheet in order to describe the disk records. Here's a quick review just to refresh your mind.

Rules: Output

1. Use the first line to describe an output record.

   a. Specify filename, file type and output indicators.

2. Starting on line 2, use one line to describe each field or constant that is a part of the output record.

In our problem you are to specify one type of output file record. Each record shall contain:

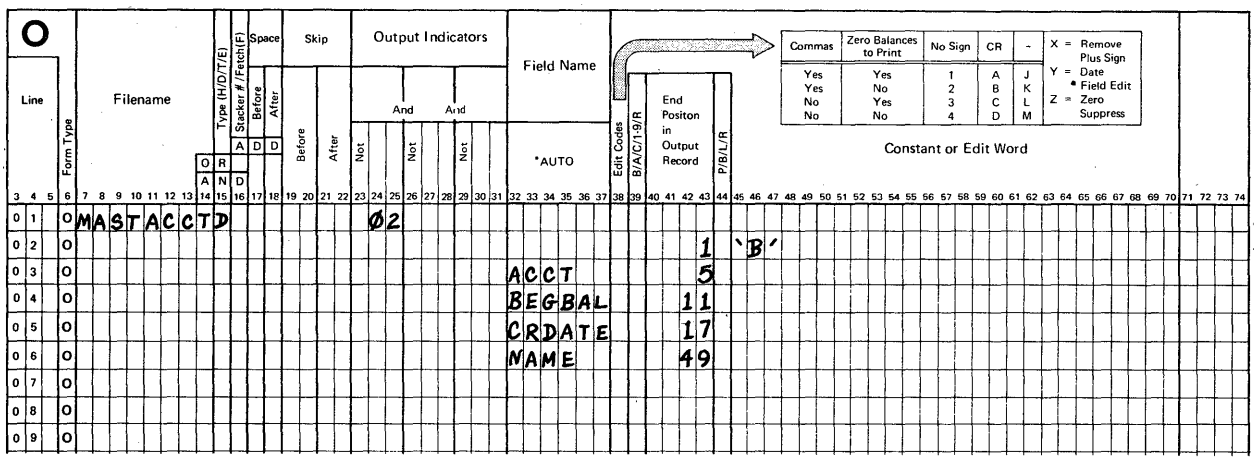
1. Code "B" in position 1,
2. Account number in 2-5,
3. A beginning balance in 6-11,
4. The creation date in 12-17, and
5. The customer's name in 31-49.

Keep in mind one more item. A disk record is to be created for every transaction as it is keyed in. Now, describe the output record and its fields or constants.

* * *

Compare your coding to the book solution.

RPG OUTPUT SPECIFICATIONS



A key question is, "Why is the correct indicator Ø2"? The answer is, "Because Ø2 is the indicator assigned to identify transactions, and we are creating an output disk record for every transaction record that is keyed". I assume that you used a constant in order to specify that code "B" is to be in position 1 of each output record. Remember to start and end every constant with a single quote mark.

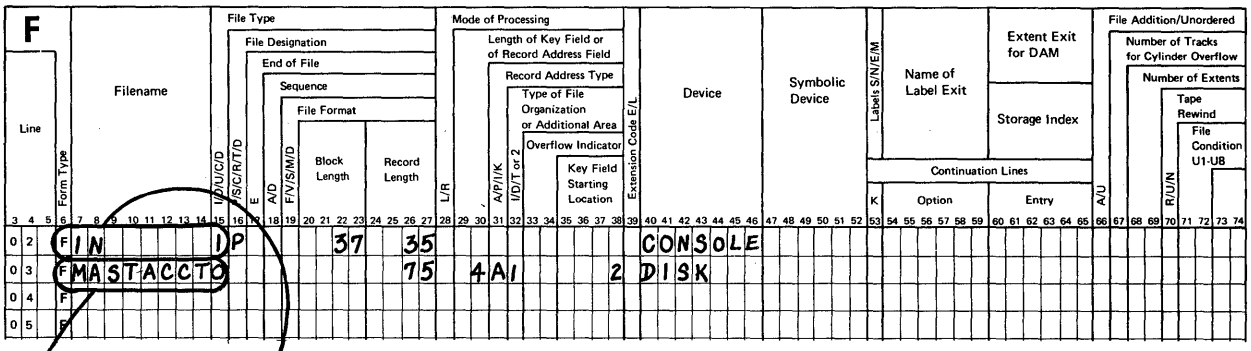
Problem 3: Summary

> "Create a file of indexed disk records from transactions keyed into the computer. A special record that contains a creation date field must be the very first record keyed in. The key field in the indexed file is account number in each transaction record".

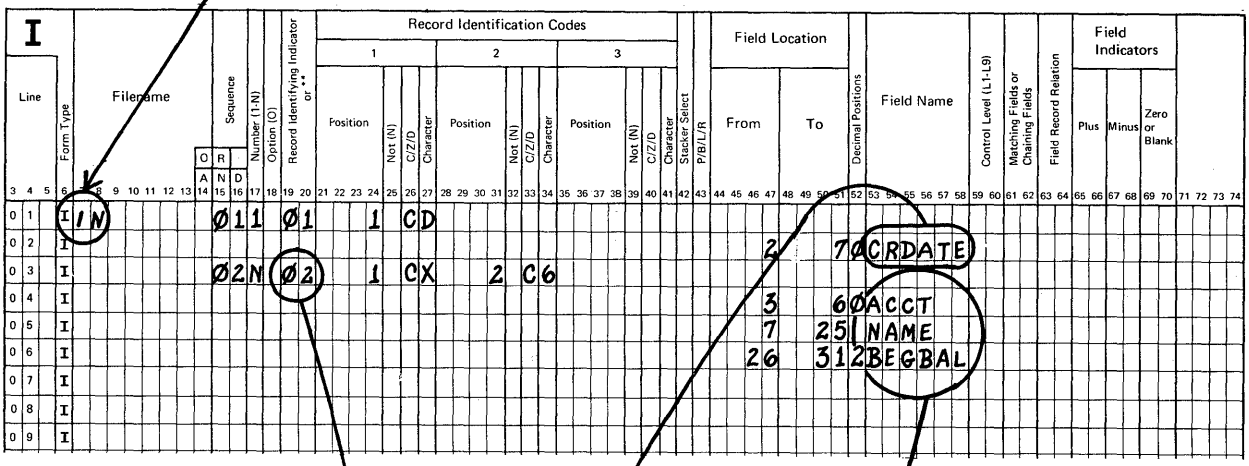
Again, the interrelated entries are connected.

Problem 3: Book Solution

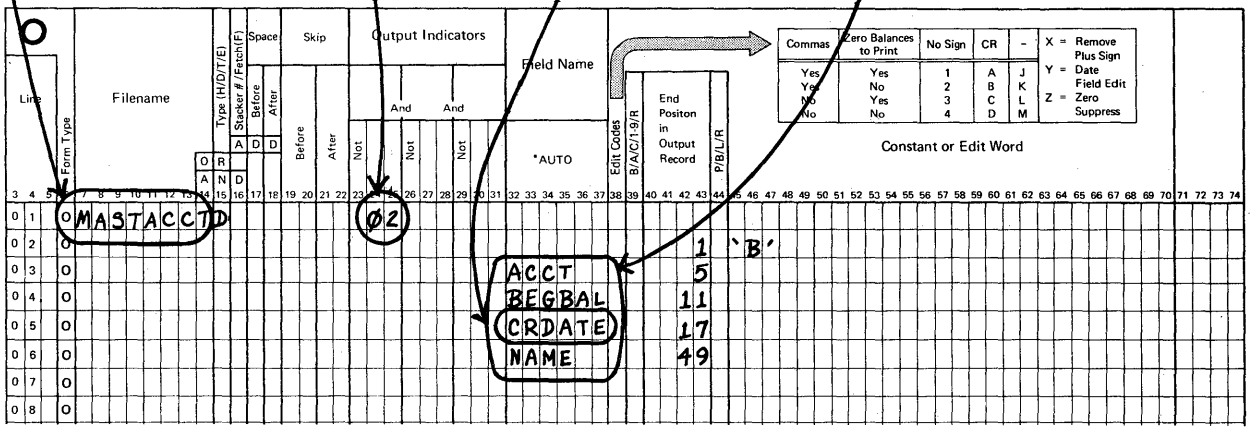
File Description Specification



RPG INPUT SPECIFICATIONS



RPG OUTPUT SPECIFICATIONS



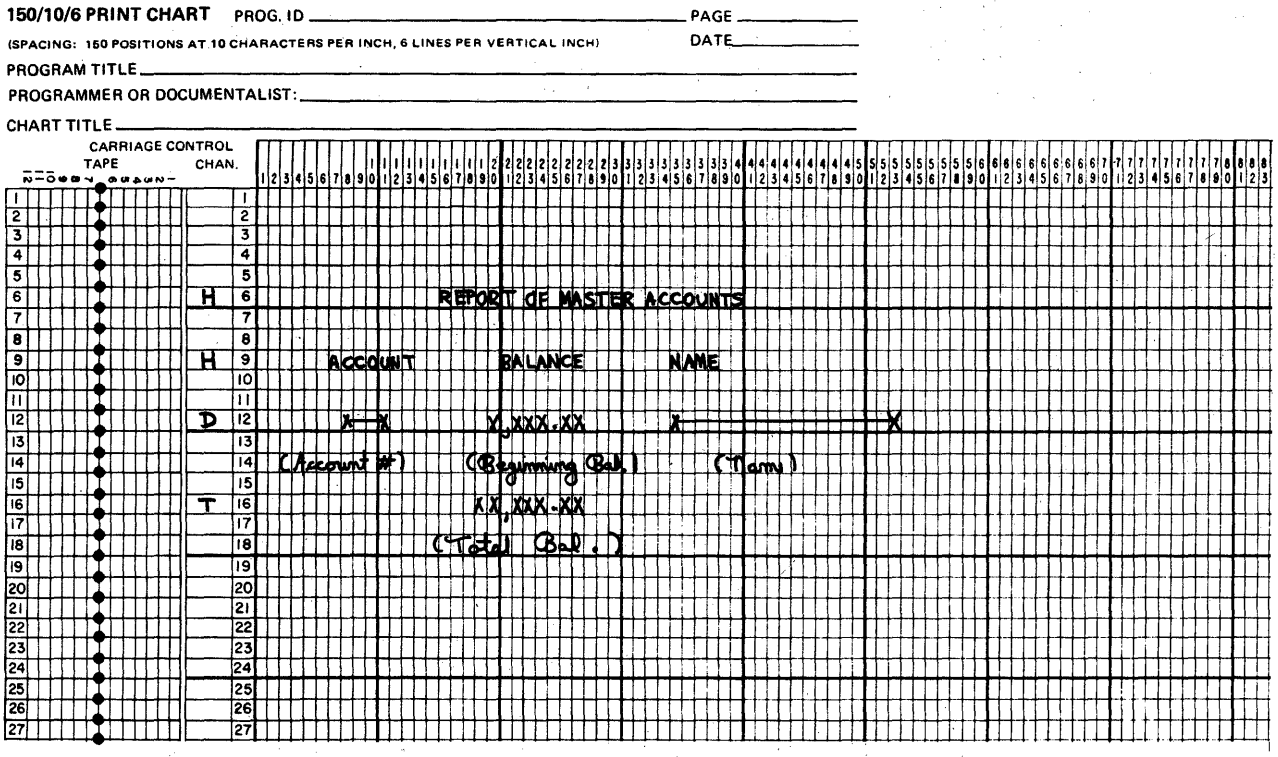
Problem 4: "Print the contents of the indexed file created in problem three. As a part of the report, print the total amount of all the "beginning balance" fields in the entire file".

Sample Report Page

REPORT OF MASTER ACCOUNTS

ACCOUNT	BALANCE	NAME
1002	100.00	MONEMATERS
2116	2,160.00	CASHINFLOW
2249	1,728.50	BANKCREDIT
2367	750.60	CASHIN
	4,739.10	

Sample Print Chart



Disk Input Records - Each is 75 positions long.

Position	Field
1	Code "B"
2 - 5	Account Number (Key Field)
6 - 11	Beginning Balance, 2 decimals
31 - 49	Name

In addition to the information just provided, you need to know how to code the calculation used to compute the total amount of all the beginning balances. Refer to a blank Calculation sheet as you read through these rules.

Rules: Computing a total from data in every record.

1. Indicators (9-17) - Copy the Record Identifying Indicator assigned to the record on the Input sheet.

2. Factor 1 (18-27) - Invent a name to hold the total values to be accumulated from the Input field. Start in position 18.

3. Operation (28-33) - Specify ADD in positions 28-30.

4. Factor 2 (33-42) - Copy the input field name whose values are to be accumulated. Start in position 33.

5. Result Field Name (43-48) - Copy the name you entered as Factor 1. Start in position 43.

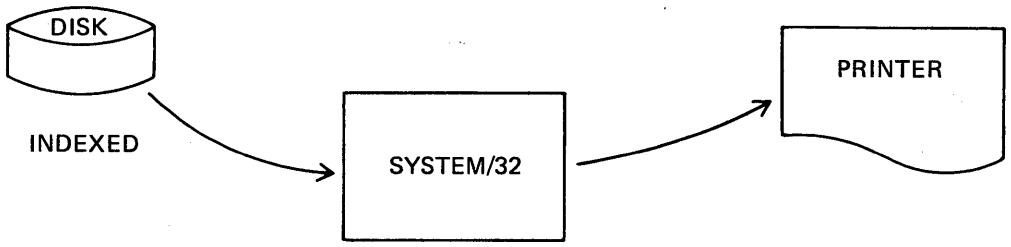
6. Length (49-51) - Specify a length at least equal to the length of the field named as Factor 2.

   a. A better length may be one that is larger because we will be adding to the total over and over.

   b. Omit leading zeros.

7. Decimal Positions (52) - Specify the number of decimal positions that shall be in the final total.

The system flowchart is also provided for reference.



When you examined the printed report page and the print chart you may have noticed that decimal points and commas appeared in the amount fields. In RPG II we say that these numeric fields are to be "edited". To do this we will need to include an edit code in positions 38 on the same line as the numeric field that is to be edited. A list of codes is shown on the Output sheet for reference. When any edit codes except X, Y or Z are used a decimal point will be included as needed. One question, "When should the final total be printed?" After the last record has been processed. Use the last record indicator (LR) to control this printing.

I think you are ready to code the entire solution to this problem since it is similar to problem 2. Review all information about the problem before you start the coding. You will need to describe the files, records, fields, constants and calculations. Take your time and code carefully. When you have finished, return to this point in the text.

* * *

Let's review all entries, one at a time. I assume that yours are different from mine, so let's review by asking questions about your coding.

File Description

1. What filenames did you choose, and does it matter?

2. How long is a printed record?

3. Did you assign an overflow indicator?

4. How did you describe the disk file records as having indexed organization?

If your solution is correct your answers to the 4 questions should be:

1. It doesn't matter provided each name is unique.
2. On System/32 printers, records are 132 positions long.
3. Yes (and you should have one of the acceptable codes).
4. By describing:
   a. a key field length,
   b. a code "A" record address type,
   c. an "I" (indexed) file organization, and
   d. a key field starting location.

Input

1. Did you copy the input filename?
2. Did you specify two letters for sequence?
3. Did you specify a record identifying indicator?
4. Did you specify the character B as the record identifying code?
5. How many fields did you describe? How many were numeric?

Your answers should be "Yes" for items 1-4 and these records have 3 fields, one of which is numeric.

NOTE: If you specified the creation date field it's O.K., but it is not needed for this report.

Calculation

I assume that you followed the coding rules since this sheet is new. Look at your entry under "Result Field Length" (48-51). If you specified "7" as your entry, fine. If not, did you examine the print chart to see how many X's were planned for this field? Also, since the beginning balance field on the input records is from 6-11 which is 6 positions, it's a good practice to specify the total field to be at least one position larger.

Output

1. The report title and the three column headings are to print on every page of the report. Will yours?

2. The print chart shows headings on lines 6 and 9, a detail line on 12 and a total line four lines below that. Do your entries for spacing and skipping provide for this correctly? If not, what will you change? Also, the printed report shows that detail lines are double spaced. How about yours?

3. The report title is 25 characters long. The longest constant I can specify on one line of coding is 24 characters because I must start and end with a single quote mark. How did you handle this problem?

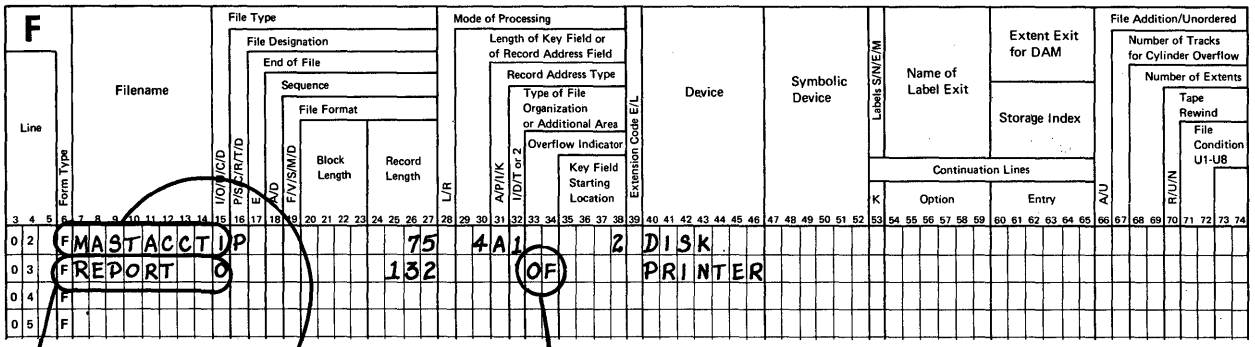
4. Did you include an edit code for the balance field and for the final total? If not, specify one now.

5. Did you remember to control the printing of the total line by specifying the last record indicator, LR? If not make an entry for this now.

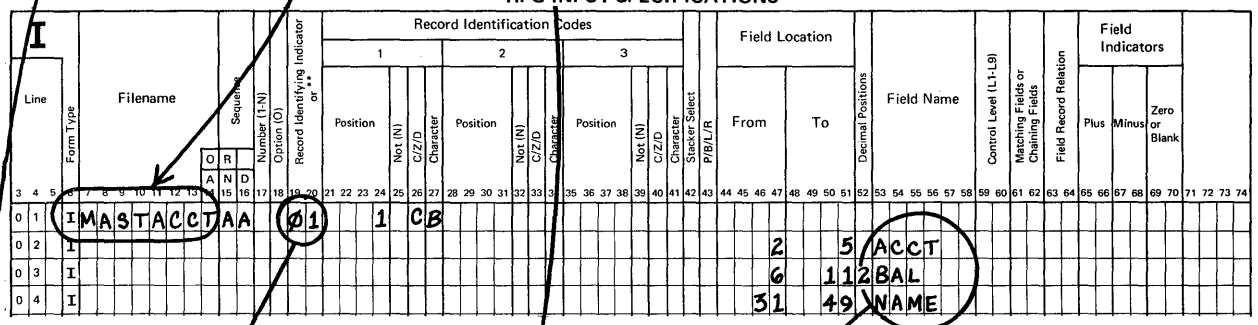
The book solution is shown for reference. Your entries should be similar. If you disagree with these entries and you don't know why they are correct, re-read the problem and look at the samples.

Problem 4: Book Solution

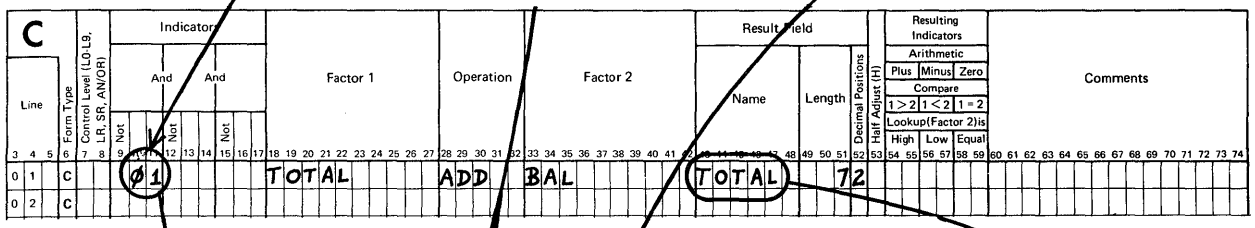
File Description Specification



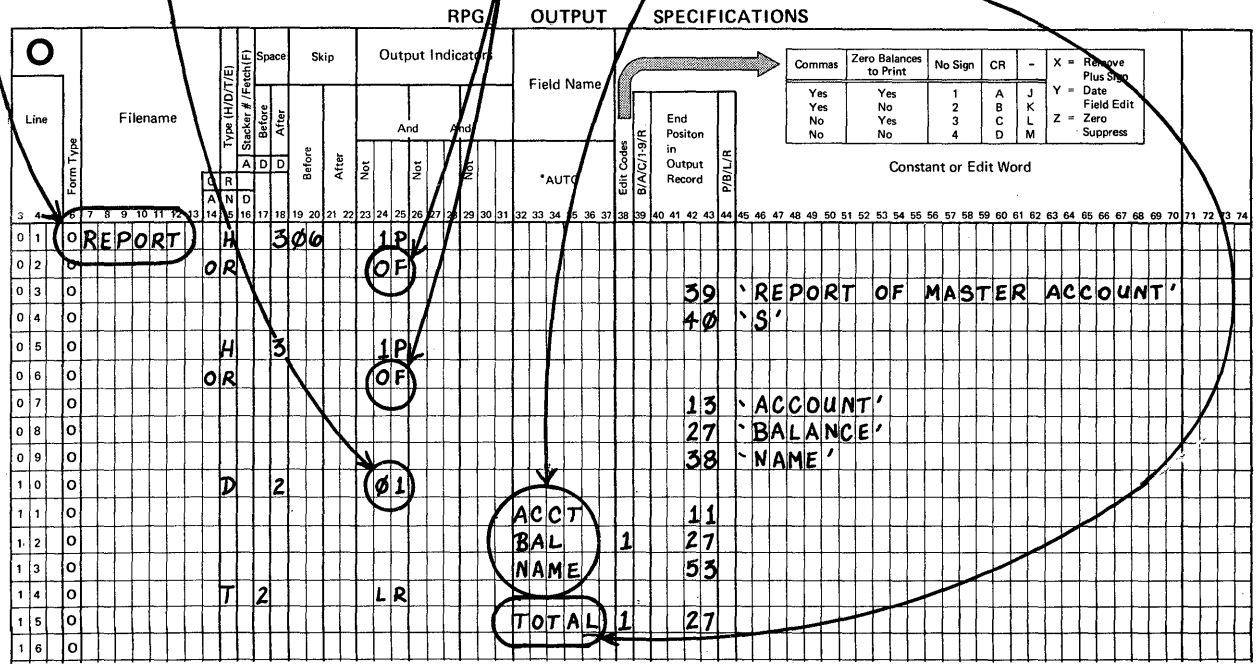
RPG INPUT SPECIFICATIONS



RPG CALCULATION SPECIFICATIONS



RPG OUTPUT SPECIFICATIONS



Problem 5 sounds just like problem four.

Problem 5: "Print the contents of the indexed file created in problem three. As a part of the report, print the total amount of all the "beginning balance" fields in the entire file".

Sample Report Page

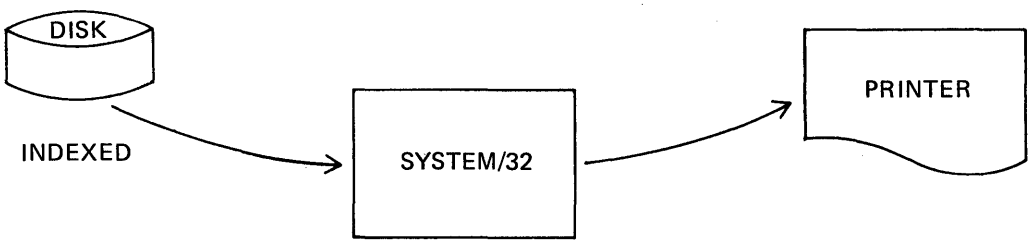
REPORT OF MASTER ACCOUNTS

ACCOUNT	BALANCE	NAME
1002	100.00	MONEMATERS
2116	2,160.00	CASHINFLOW
2249	1,728.50	BANKCREDIT
2367	750.60	CASHIN
	4,739.10	

Input Records

Position	Field
1	Code "B"
2 - 5	Account Number (key field)
6 - 11	Beginning Balance, 2 decimals
31 - 49	Name

System Flow



Problem five is the same as problem four, but we are going to make use of the Auto Report function of the RPG II language. In particular, you will learn to use the *AUTO entry on the Output specification sheet.

If you were to describe the files and input records and fields for this problem you would make only one change: omit the entry for "Overflow Indicator" on the File Description sheet. The reason for this is that Auto Report will select and insert an overflow indicator for you automatically.

In this particular example you are required to specify the coding needed to accumulate a total of the balance in each record. For problem four we did that by specifying the ADD operation on the Calculation sheet. In problem five we will not need to specify any calculations because, you guessed it, Auto Report can provide for this kind of a calculation with one extra entry on the Output sheet.

What else does Auto Report do for me? Let's investigate it in more detail.

Using the *AUTO entry on the Output sheet provides for reports by including:

1. automatic selection and insertion of an overflow indicator to control the printing of overflow lines.

2. automatic assignment of indicator 1P to control the printing of headings on the first page of reports.

   NOTE: A report may include a date and page numbering. *AUTO inserts a field called UDATE for the date and a field called PAGE for the page number.

3. calculations for certain types of accumulations with minimum coding.

4. edit codes for numeric data fields.

5. automatically centered report page titles.

6. automatically positioned column headings for fields.

7. predetermined spacing and skipping between lines.

NOTE: Items 5, 6 and 7 eliminate the need to design print charts.

Are you kidding? What's left for me, the programmer, to describe?

As I said, you describe the files on the File Description sheet, omitting the entry for Overflow Indicator. You also describe the Input fields to be used. It may not be necessary to describe any calculations. The Auto Report feature *AUTO is specified on the Output sheet following these rules.

1. Describe the report title heading line including:

   a. filename (7-14)

   b. type (15) - Enter an H

   c. *AUTO (32-36)

2. On the next line or lines specify the actual report title as one or more constants.

   a. single quote to start

   b. constant, up to 24 characters

   c. single quote to end

3. Next, for our particular problem, describe:

   a. Type (15) - Enter a D for detail line.

   b. Specify a spacing entry if you wish.

   c. Specify the indicator assigned on Input.

   d. Specify *AUTO in 32-36.

4. On the next line or lines specify a field name and a constant. The constant will become your heading for that field! Use one coding line for each field and its constant. A printed field does not require a column heading in order to make use of Auto Report.

5. If the value of any field is to be accumulated for every record, include the code letter "A" in position 39 on that same line. This entry automatically generates the necessary calculations.

When a problem lends itself to the use of Auto Report you should try to use it to reduce your coding effort.

Use the File Descriptions and Input Specifications shown and then try to code a solution to problem five using the *AUTO feature of Auto Report on a blank Output sheet. Re-read the rules for its use as you work out your solution.

REPORT OF MASTER ACCOUNTS

ACCOUNT	BALANCE	NAME
1002	100.00	MONEMATERS
2116	2,160.00	CASHINFLOW
2249	1,728.50	BANKCREDIT
2367	750.60	CASHIN
	4,739.10	

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	Length of Key Field or of Record Address Field	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	K	Name of Label Exit	Option	Extent Exit for DAM	Storage Index	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	R/L/N	Tape Rewind	File Condition U1-U8
02	F	MASTACCT	I	P					75		4	A	1		2	DISK													
03	F	REPORT	O						132							PRINTER													
04	F																												
05	F																												
06	F																												

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	MASTACCTAA				01	1		C	B																				
02	I																				2	5		ACCT						
03	I																				6	11	2	BAL						
04	I																				31	49		NAME						
05	I																													
06	I																													

Here's the total solution for problem 5 using the *AUTO feature of Auto Report. How's yours?

File Description Specification

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	L/R	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	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines K	Continuation Lines Option	Continuation Lines Entry	File Addition/Unordered A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind R/U/N	File Condition U1-U8
			I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D				A/P/I/K			I/D/T or 2																
02	F	MASTACCT	I	P					75		4	A	1				2	DISK												
03	F	REPORT	O						132									PRINTER												
04	F																													
05	F																													

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank
			O/A/R/N/D																											
01	I	MASTACCT	AA			01		1	C	B																				
02	I																				2	5		ACCT						
03	I																				6	11	2	BAL						
04	I																				31	49		NAME						
05	I																													
06	I																													

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Field Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
			O/A/R/N/D	A/D	D				Not	Not / Not	*AUTO	B/A/C/1-9/R			
01	O	REPORT	H								*AUTO				
02	O														'REPORT OF MASTER ACCOUNT'
03	O														'S'
04	O		D		2				01		*AUTO				
05	O										ACCT				'ACCOUNT'
06	O										BAL	A			'BALANCE'
07	O										NAME				'NAME'
08	O														
09	O														

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Problem 6: Create an indexed file of personnel records. These output records are 57 positions long. The key field is the "name" field which is in positions 17-40 of each disk record. The personnel records are keyed in from the console. Here is a list of the input and output record fields that you are to describe.

INPUT RECORDS

Position	Keyed Data Field
1	Code P
2-10	Social Security Number
11-16	Date of Hire
17-40	Name
41-45	Man Number
46-48	Location
49-51	Department

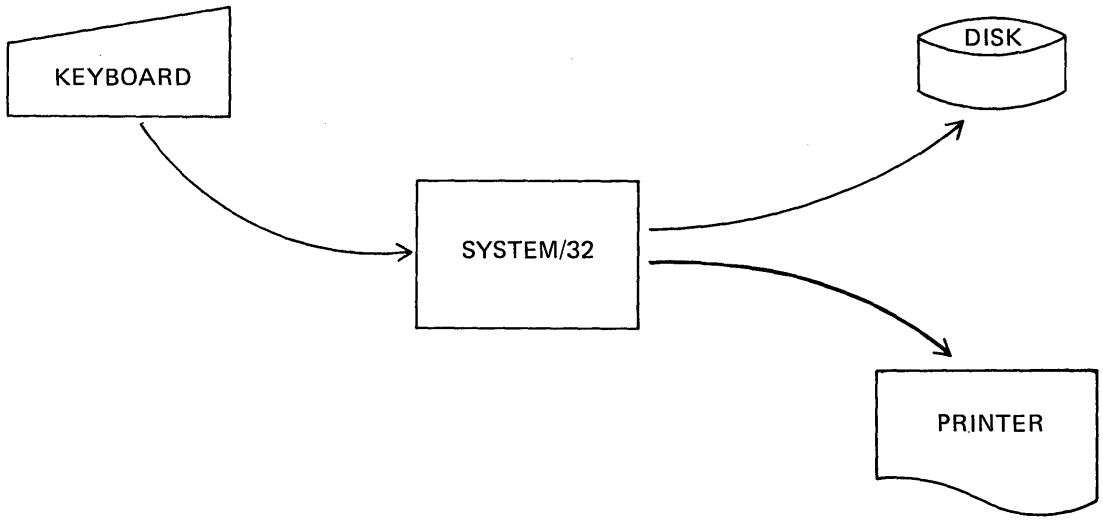
OUTPUT RECORDS

Position	Disk Data Field
1	Code X
2-10	Social Security Number
11-16	Date of Hire
17-40	Name
41-45	Man Number
46-48	Location
49-51	Department
52-57	Record Creation Date

Use the special RPG II field called UDATE as an output field to provide the Record Creation Date for the disk records. By specifying UDATE, today's date as keyed into the system when it is turned on will be stored in your records.

In order to keep track of the keyed records, include the coding needed to print a list of those records. Make use of Auto Report to prepare the report.

System Flowchart



Keyed Records - Sample Data

RECORDS LIST

S S #	HIRED	NUMBER	NAME	LOC	DEPT
410225619	6/27/39	12345	ADAMS, H L	123	16A
399650002	10/23/58	56562	BERWYNN, M M	705	95C
311114655	4/30/60	70711	DOMINIACE, T R	T66	11A
158097433	7/31/66	89887	FISCHER, A L	6X8	44B

Code a solution at this time.

* * *

Problem 6: Book Solution

Notice the use of the edit code "Y" on the Output sheet, line 14 position 38. This code is used to edit date fields such as date of hire (DOH).

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	File Addition/Unordered A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind R/U/N	File Condition U1-U8
02	F	PERSIN	I	P				60	58								CONSOLE													
03	F	PERSOUT	O						57		24	A	I		17		DISK													
04	F	RECLIST	O						132								PRINTER													
05	F																													

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	PERSIN	AA			01		1		CP																			
02	I																			2	10		SS						
03	I																			11	160		DOH						
04	I																			17	40		NAME						
05	I																			41	45		MNO						
06	I																			46	48		LOC						
07	I																			49	51		DEPT						
08	I																												

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators: And Not	Output Indicators: And Not	Output Indicators: And Not	Field Name *AUTO	Edit Codes B/A/C/1/9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	PERSOUT	D						01							
02	O													1		'X'
03	O											SS		10		
04	O											DOH		16		
05	O											NAME		40		
06	O											MNO		45		
07	O											LOC		48		
08	O											DEPT		51		
09	O											UDATE		57		
10	O	RECLIST	H									*AUTO				
11	O															'RECORDS LIST'
12	O		D		2				01			*AUTO				
13	O											SS				'S S #'
14	O											DOH	Y			'HIRED'
15	O											MNO				'NUMBER'
16	O											NAME				'NAME'
17	O											LOC				'LOC'
18	O											DEPT				'DEPT'
19	O															

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Chapter 2: Summary

Disk files may be created by keying in data from the console keyboard. In this kind of problem, the disk file being created is an output file. You have seen examples in which the created files were organized sequentially and indexed.

At this point you should be able to code a solution that requires the creation of disk file records. Also, you should be able to make use of the Auto Report feature to print a listing of disk file records. It is assumed that you would refer to the System/32 RPG II Language manual as needed.

The RPG II coding entries covered in this chapter are grouped by specification sheet for your reference.

FILE DESCRIPTION

7 - 14	Filename
15	File Type (Input, Output)
16	File Designation (Primary input)
20 - 23	Block Length (Console file only)
24 - 27	Record Length
29 - 30	Length of Key Field (Indexed only)
31	Record Address Type (Indexed only)
32	Type of File Organization (Indexed only)
33 - 34	Overflow Indicator (Printer only)
35 - 38	Key Field Starting Location (Indexed only)
40 - 46	Device (CONSOLE, DISK, PRINTER)

INPUT

7 - 14	Filename
15 - 16	Sequence
17	Number (when 15-16 entry is numeric)
19 - 20	Record Identifying Indicator
21 - 41	Record Identification Codes
44 - 51	Field Location
52	Decimal Positions (numeric fields only)
53 - 58	Field Name

CALCULATION

9 - 17	Indicators
18 - 27	Factor 1
28 - 32	Operation (ADD)
33 - 42	Factor 2
43 - 52	Result Field Name, Length and Decimal Positions

OUTPUT

7 - 14	Filename
14 - 15	OR, for headings
15	Type (Heading, Detail, Total)
17 - 18	Space
19 - 20	Skip Before
23 - 31	Output Indicators (1P, OF, Ø1, Ø2, LR)
32 - 37	Field Name, including special-purpose fields UDATE & PAGE
32 - 37	*AUTO (for Auto Report only)
38	Edit Codes (1, to include commas and zero balances)
38	Edit Codes (Y, for date fields)
39	A (to accumulate with Auto Report)
40 - 43	End Position in Output Record
45 - 70	Constants (on heading and detail lines)

SELF TEST - Chapter 2

Try to answer these questions from memory. When you finish, turn the page to check your answers. Use a piece of scratch paper rather than writing in this book.

1. What is the longest allowable length for a key field in an indexed disk file?

2. For what two kinds of descriptions is the Input specification sheet used?

3. For what two kinds of descriptions is the Output specification sheet used?

4. What special character is needed to describe a constant on the Output sheet?

5. What special reserved RPG II field supplies the date keyed into the system when needed on a report or output record?

6. True or False?

   Each file must have a unique name in a program.

7. What special entry is used to signify the use of Auto Report output records?

ANSWERS - Chapter 2 Self Test

Compare your answers to the ones below. If you missed more than 2, you might wish to re-read the chapter before continuing.

1. Longest key field is 29 positions.

2. Input sheet - first, to describe input record types and second, to describe fields in each of those record types.

3. Output sheet - first, to describe output record types and second, to describe fields or constants that are part of those record types.

4. A single quote mark must be coded ahead of and another must be coded following each constant.

5. The special date field is named UDATE.

6. True. Each file must have a unique name in a program.

7. The special entry, *AUTO, is used to signify the use of Auto Report output records.

If you had 7 correct answers, congratulations! A score of 6 is also very good. Don't forget, if you made a number of errors, you should probably re-read this chapter.

Chapter 3: Processing and Maintaining Disk Files (2 to 3.5 hours)

After a disk file has been created, that is, stored on the disk, its records may be processed in a number of ways. The file may be used as input as we used them in the previous chapter when we listed the disk file records on the printer.

In this chapter we will be solving problems that involve what is known as "file maintenance". What we can do in order to maintain a file of disk records includes:

1. adding new records to it,
2. changing data in one or more existing records in the file in order to "update" the file, and
3. identifying selected records to be deleted from the file at a later point in time.

CHAPTER OBJECTIVES

When you have finished studying this chapter, "Processing Disk File Records", you should be able to code solutions to problems in which any of these conditions (adding records, changing data, and selecting records for future deletion) exist.

Again, a series of data processing problems will be presented.

1. Add records to an existing sequential file.
2. Add records to an existing indexed file.
3. Use the chaining technique to inquire about selected records in an indexed file.
4. Use the chaining technique to update information in certain records in an indexed file, and flag certain other records for deletion.
5. Use the matching records technique to update records in a sequentially organized file.

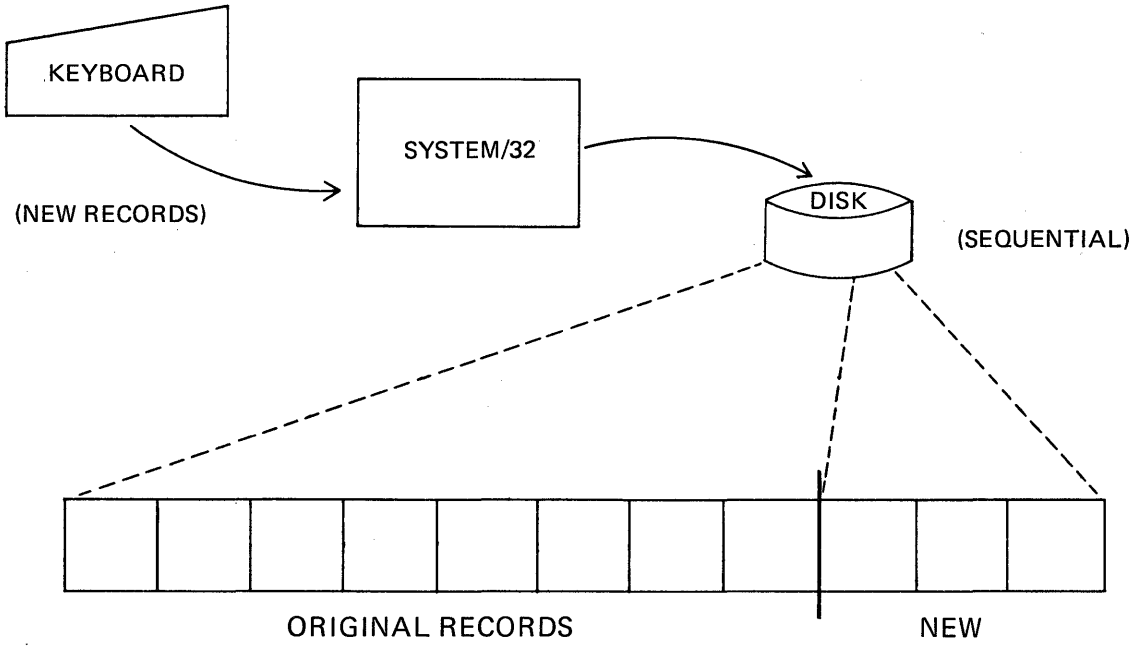
NOTE: You may wish to refer to the optional topic in the front of this book in order to review file processing methods before you continue your study of this chapter.

In addition to this text you should have a number of each of the following RPG II specification sheets.

GX21-9092	Control and File Description
GX21-9094	Input
GX21-9093	Calculation
GX21-9090	Output

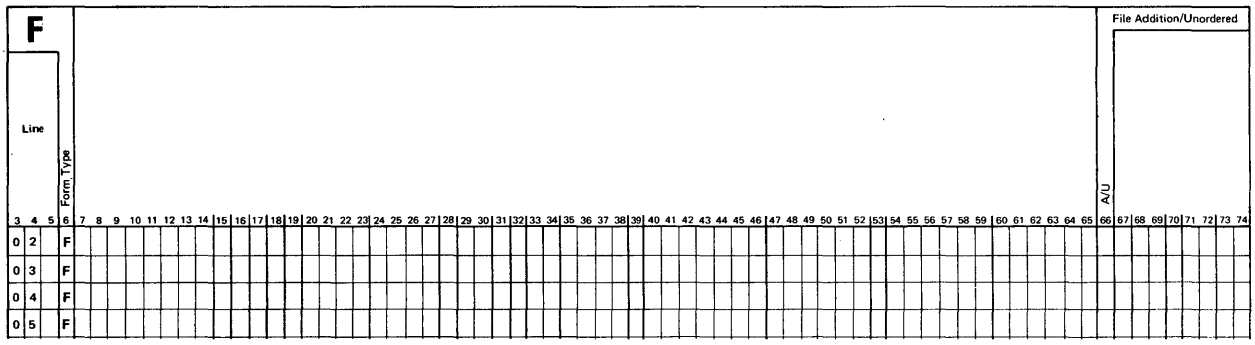
A self test is included at the end of the chapter. Use it to measure your understanding of the topics in this chapter.

Problem 1: Add records to an existing sequential disk file.

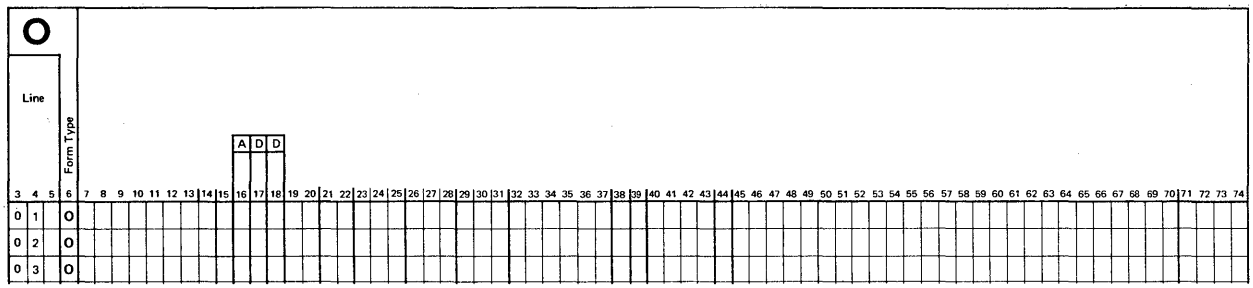


When records are to be added to existing records in a sequential file you need to specify two new entries. These figures highlight the new entries on the File Description and Output sheets.

File Description Specification



RPG OUTPUT SPECIFICATIONS



The entries to be made are:

FILE DESCRIPTION

1. File Addition (66) - Enter code "A" to indicate that records are to be added to the file.

OUTPUT

1. ADD (16-18) - Enter "ADD" when describing the type of record to be added.

In this problem you are to code the entries needed to add records to a sequential file named INTER. Records in this disk file are 80 characters long and have code I in position 1. The keyed input records (new) contain these fields:

Keyed Input Records

Position	Input Field
1	Code N
2 - 7	Item Number
8 - 28	Item Description
29 - 31	Unit Code
32 - 36	Unit Cost, 2 decimals
37 - 40	Units in stock

The output records contain the same fields in different positions. Also, today's date (UDATE) is to be included so we know the creation date. This special field is provided for your use in any RPG II program where today's date, as keyed into the System/32 when the system is turned on, is needed.

Output Records

Position	Output Field
1	Code I
2 - 7	Today's Date
8 - 13	Item Number
14 - 34	Item Description
35 - 37	Unit Code
38 - 42	Unit Cost
43 - 46	Units in Stock

One more point before you code a solution.

Rule: The disk file to which records shall be added is to be described as an output file.

Code your solution now.

* * *

Problem 1: Book Solution

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	NEW	I	P				48	46								CONSOLE													
03	F	INTER	O						80								DISK									A				
04	F																													
05	F																													
06	F																													
07	F																													

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	NEW	AA			01	1		C	N																				
02	I																				2	7		ITEM#						
03	I																				8	28		DESC						
04	I																				29	31		UCODE						
05	I																				32	36		UCOST						
06	I																				37	40		USTOCK						
07	I																													
08	I																													
09	I																													
10	I																													

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Output Indicators Not	Output Indicators Not	Output Indicators Not	Field Name *AUTO	Edit Codes B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	INTER	D	ADD						01							
02	O														1		'1'
03	O												UDATE		7		
04	O												ITEM#		13		
05	O												DESC		34		
06	O												UCODE		37		
07	O												UCOST		42		
08	O												USTOCK		46		
09	O																
10	O																

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

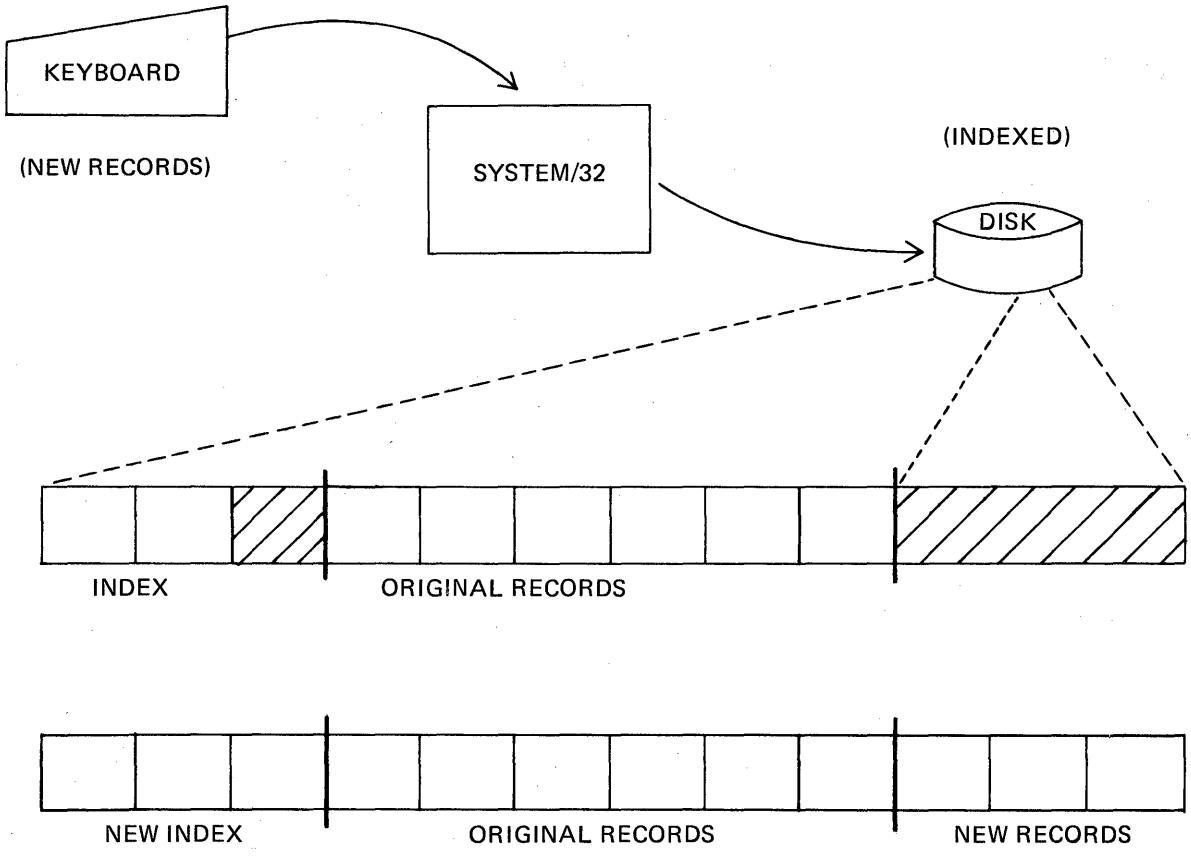
Look again at my entry for "unit cost" on the Input sheet. Even though it was stated that this field contains 2 decimal positions, I did not specify it in position 52 for that field. My reason for not specifying it is this.

Decimal positions are needed for calculations and for edited (commas and decimal points) output. Unit cost is not used for either calculations or edited output so I do not need to use the entry. If you included a 2, your answer is also correct.

That was an example in which a file of sequential disk records was increased in size by the addition of new records.

Our second problem requires the same kind of coding needed to solve problem one. As you recall, in the second problem we are to add records to an existing indexed file.

Problem 2: Add records to an existing indexed file.



This illustration looks far more complicated than the one before it, but the actual coding for the addition of records is the same! RPG II automatically arranges the index entries after you describe the file as being indexed with key fields.

OK, code a solution for this problem.

Problem 2: Add records to an existing indexed file. The existing records are 60 characters long with a key field in positions 2-8. The code character "$" is in position 1 of each disk record. The keyed-in records are to contain:

Position	Input Field
1	Code "$"
2 - 8	Identification (key field)
9 - 21	Transaction Data
22 - 26	Start Value
27 - 31	Last Value

Disk record data is to be arranged like this:

Position	Output Field
1	Code "$"
2 - 8	Identification (key field)
28 - 32	Start Value
33 - 37	Last Value
48 - 60	Transaction Data

* * *

Problem 2: Book Solution

NOTE: I used a field named CODE in order to place a $ in the output record. You could have specified a constant if you wished, as it is also correct.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8	R/U/N
02	F	EXTRA	I	P					36								CONSOLE														
03	F	START	O						60		7	A	I			2	DISK									A					
04	F																														
05	F																														
06	F																														

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	EXTRA	AB			01		1	C	$																				
02	I																				1	1		CODE						
03	I																				2	8		ID						
04	I																				9	21		TDATA						
05	I																				22	26		SVAL						
06	I																				27	31		LVAL						
07	I																													
08	I																													

RPG OUTPUT SPECIFICATIONS

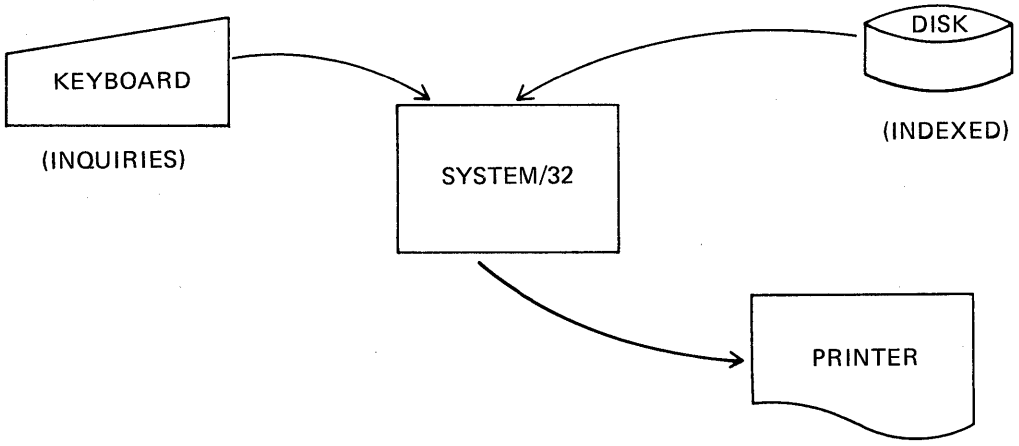
Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Output Indicators And	Field Name	Edit Codes	End Position in Output Record	P/B/L/R
01	O	START	D	ADD					01						
02	O											CODE		1	
03	O											ID		8	
04	O											SVAL		32	
05	O											LVAL		37	
06	O											TDATA		60	
07	O														
08	O														

Constant or Edit Word

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Problem 3: Use the chaining technique to inquire about selected records in an indexed file. Print the desired information when found.



When the chaining technique is used to search a file, you need to specify a number of new entries. These figures highlight the new entries on the File Description and Calculation sheets.

File Description Specification

F

Line	Form Type	P/S/C/R/T/D	L/R	File Designation	Mode of Processing
02	F				
03	F				

RPG CALCULATION SPECIFICATIONS

C

Line	Form Type	Factor 1	Operation	Factor 2	Resulting Indicators
01	C				
02	C				
03	C				

The entries to be made are:

FILE DESCRIPTION

1. File Designation (16) - Enter code "C" for the input disk file to designate it as the chained file.

2. Mode of Processing (28) - Enter code "R" for the input disk file to denote that its records are to be processed randomly.

CALCULATION

1. Factor 1 (18-27) - Enter the name of the field in the console file that contains the search field.

2. Operation (28-32) - Enter CHAIN to do the search.

3. Factor 2 (33-42) - Enter the name of the chained file.

4. Resulting Indicators (54-55) - Assign an unused indicator.

   a. If no corresponding record is found, this indicator turns ON.

   b. If a desired record is found, this indicator stays OFF.

OUTPUT

1. Output Indicators (23-31) - Include the letter N (for Not) along with the Resulting Indicator assigned to the CHAIN operation in 54-55 on the Calculation sheet, when the search is successful.

In this problem you are to code the entries needed to search an indexed file and print a reference list of that activity. The indexed file records are 65 positions long. The input files contain:

DISK FIELDS		KEYED FIELDS	
12-26	Customer Name (Key Field)	1	Code X
43-49	Customer Balance, 2 decimals	2-16	Customer Name (Search Field)

The output list is to include a message, NAME NOT FOUND IN FILE, if the search is not successful. Here's a sample of the desired reference listing.

NAME	ACCOUNT BALANCE
SMITH	307.16
LAWRENCE	1,000.00
JONS	NAME NOT FOUND IN FILE
MILLER	500.00
ADDAMS	17,560.60

Refer to the print chart for end positions when coding output sheet entries.

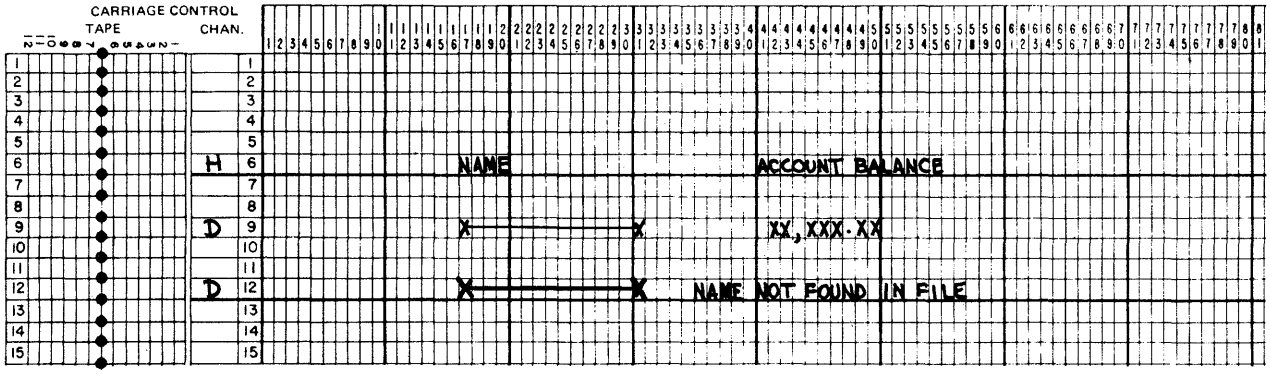
150/10/6 PRINT CHART PROG. ID _____ PAGE _____

(SPACING: 150 POSITIONS AT 10 CHARACTERS PER INCH, 6 LINES PER VERTICAL INCH) DATE _____

PROGRAM TITLE _____

PROGRAMMER OR DOCUMENTALIST: _____

CHART TITLE _____

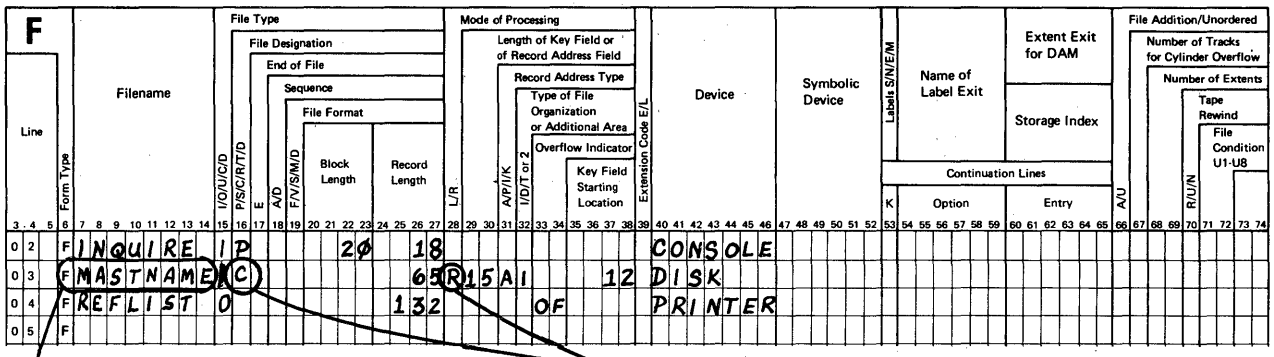


Code your solution at this time.

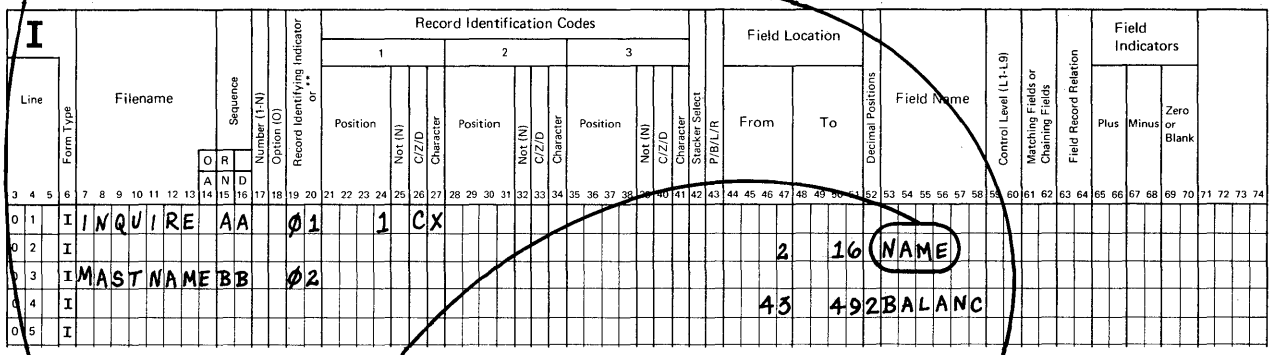
* * *

Problem 3: Book Solution

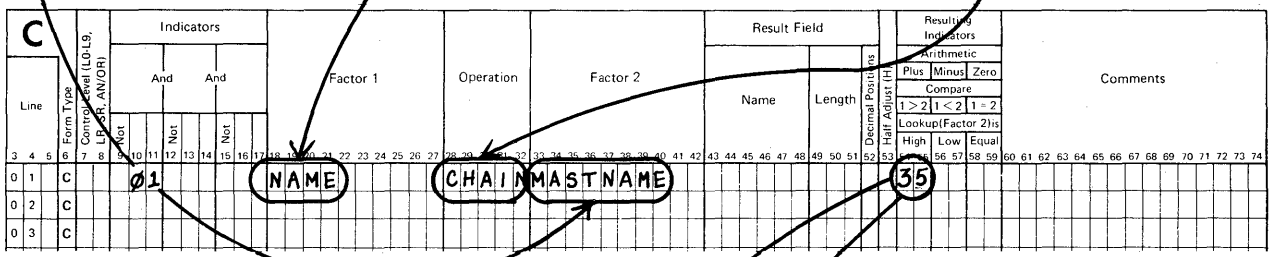
File Description Specification



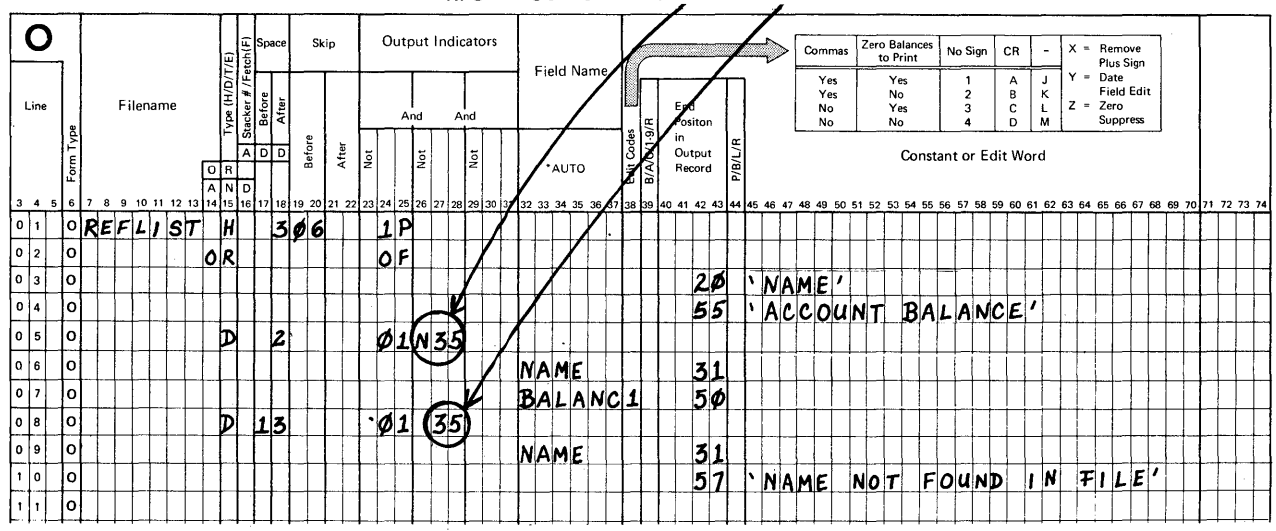
RPG INPUT SPECIFICATIONS



RPG CALCULATION SPECIFICATIONS

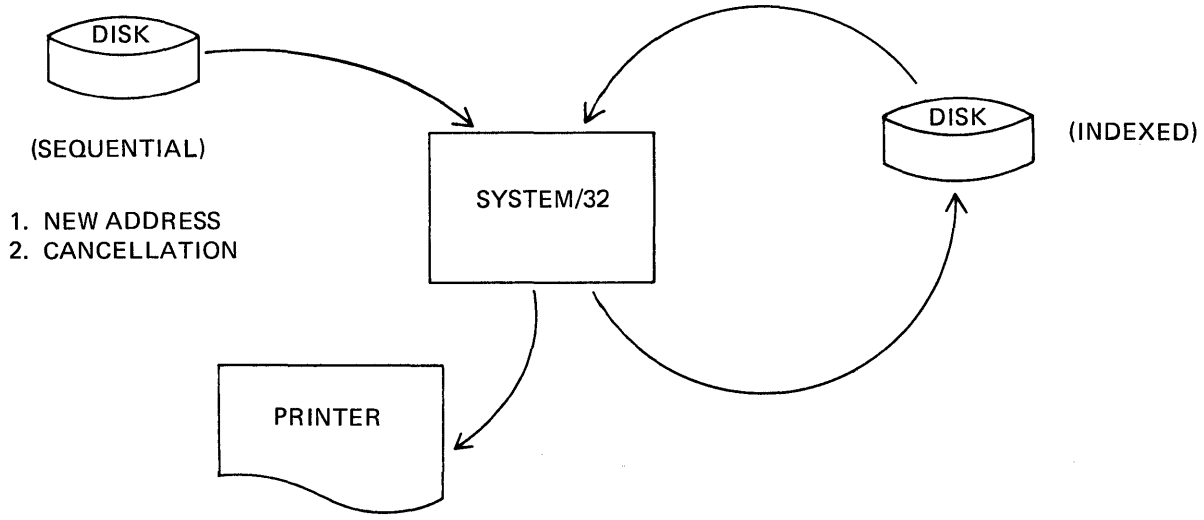


RPG OUTPUT SPECIFICATIONS



Problem 4: Use the chaining technique to update information in certain records in an indexed file and "flag" other selected records for deletion.

This problem is an example of file maintenance. A master indexed file contains information about subscribers. The update activity involves change of address notifications while flagging is to be done for cancellations. Both types of change are found in a file of sequentially organized disk records created during a prior run.



There are two types of records in the master file. Active records contain the code "M" in position 1. Records that were flagged for deletion on a prior run contain the code "D" in position 1. The reason for including printed output is to list any "not found" records following the CHAIN operation. A sample report is shown for your reference.

```
                RECORDS NOT FOUND

SUBSCRIBER

235476        ADDRESS CHANGE

100114        ADDRESS CHANGE

767676                              DELETION
```

Also for your reference is this list of input fields from both disk files.

File	Record Type	Field(s)	
Changes	Address Change	1	Code "A"
		2- 7	Subscriber Number
		8-47	New Address
	Deletion	1	Code "D"
		2- 7	Subscriber Number
Master		1	Code "M" or "D"
		2- 7	Subscriber Number (Key Field)
		8-47	Address
		48-53	Expiration Date

Before you code a solution let's consider some key points about updating records and flagging for deletion.

UPDATE

1. The record to be updated must be found in the file.
2. The file to be updated acts as both an input and an output file.
3. The file is described on a File Description, Input and Output sheet.
4. Only the field or fields to be updated are described, not the entire record.
5. If a field is to be replaced by substituting a new field value or constant, the field does not need to be described on the Input sheet.

For our problem you need to learn one new File Description entry. To describe an update file, specify "U" as the File Type in position 15.

FLAGGING FOR DELETION

1. The record to be flagged must be found in the file.

2. A special deletion code replaces an existing code or blank to mark the record for deletion.

At this time you should describe both disk files on a File Description sheet and an Input sheet. Re-read the problem if necessary. We will randomly process the master records using the chaining technique.

* * *

My descriptions are shown for reference. Do you have similar entries? If not, examine the entire problem again before continuing.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	File Addition/Unordered A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind R/U/N	File Condition U1-U8
02	F	CHANGES	I	P					47								DISK													
03	F	MASTER	U	C					53R	GA	I				2		DISK													
04	F																													
05	F																													
06	F																													

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence (OR AND)	Number (1-N)	Option (O)	Record Identifying Indicator or .	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	CHANGES	AA			01		1		CA																				
02	I																				2	7		SUBSNO						
03	I																				8	47		NEWADR						
04	I		BB			02		1		CD																				
05	I																				2	7		SUBSNO						
06	I	MASTER	CC			03		1		CM																				
07	I		OR			04		1		CD																				
08	I																													
09	I																													

Next, consider the calculations. What operation(s), under what conditions, need to be specified?

Since we will use the chaining technique, the operation will be "chain". Do you remember how to describe it? Here are some rules for reference.

CALCULATION - Chaining

1. Enter the field name used to search as Factor 1.

2. Enter CHAIN as the Operation.

3. Enter the file name to be searched as Factor 2.

4. Assign an unused indicator in positions 54-55.

Since we have two types of input changes, we need to describe two chaining operations. Include an Indicator assignment in 9-17 for each and use a unique Resulting Indicator in 54-55 for each. Specify the chaining operations at this time.

* * *

I've included both my Input sheet and my Calculation sheet for your reference so that you can see which indicators (in 9-17) control each operation. Also, I inserted a comment on the Calculation sheet (60-74) for convenience in remembering which indicator is assigned to which record search. You may include comments on your sheet if you wish. Comments may include letters, numbers or special characters.

RPG INPUT SPECIFICATIONS

I

Line	Form Type	Filename	Sequence (OR / AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	CHANGES	AA			01	1		CA																					
02	I																				2	7		SUBSNO						
03	I																				8	47		NEWADR						
04	I		BB			02	1		CD																					
05	I																				2	7		SUBSNO						
06	I	MASTER	CC			05	1		CM																					
07	I		OR			04	1		CD																					
08	I																													
09	I																													

RPG CALCULATION SPECIFICATIONS

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And/Not	Indicators: And/Not	Indicators: And/Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Arithmetic: Minus	Arithmetic: Zero	Compare: 1>2	Compare: 1<2	Compare: 1=2	Lookup(Factor 2)is: High	Lookup(Factor 2)is: Low	Lookup(Factor 2)is: Equal	Comments
01	C		01			SUBSNO	CHAINMASTER						1	1								ADDRESS CHANGE
02	C		02			SUBSNO	CHAINMASTER						2	2								DELETION
03	C																					
04	C																					

Remember this about the Resulting Indicators assigned to a CHAIN operation:

> When a record is found, the indicator remains OFF. If no record is found, the indicator turns ON.

To complete the actual updating and flagging activities we need to describe the desired output. Here are the requirements of this job.

1. For a change of address replace the old address with the new one.

2. To flag a record for deletion, replace the code M with a code D.

Code these requirements for maintenance of the master disk file at this time, but ignore the printed output for the moment.

* * *

If you described these output records correctly, the indexed master file is properly updated - some addresses are changed and particular subscriptions cancelled. Your entries for Output indicators (23-31) must include an input indicator and a calculation resulting indicator that is not on.

Rule: To describe some desired activity that is to be done when an indicator is turned off, specify the letter N ahead of that indicator.

Here's my solution up to this point. Notice the use of the "N" with its indicator on the Output sheet.

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	CHANGES	AA			Ø1			1	CA																				
02	I																				2	7		SUBSNO						
03	I																				8	47		NEWADR						
04	I		BB			Ø2			1	CD																				
05	I																				2	7		SUBSNO						
06	I	MASTER	CC			Ø3			1	CM																				
07	I		OR			Ø4			1	CD																				
08	I																													

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (LO L9, LR, SR, AN/OR)	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Arithmetic: Minus	Arithmetic: Zero	Compare: 1>2	Compare: 1<2	Compare: 1=2	Lookup(Factor 2)is: High	Lookup(Factor 2)is: Low	Lookup(Factor 2)is: Equal	Comments
01	C		Ø1		SUBSNO	CHAINMASTER												11			ADDRESS CHANGE
02	C		Ø2		SUBSNO	CHAINMASTER												22			DELETION
03	C																				
04	C																				

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And	Output Indicators: And	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Commas	Zero Balances to Print	No Sign	CR	-
01	O	MASTER	D						Ø1 N11							Yes	Yes	1	A	J
02	O										NEWADR			47		Yes	No	2	B	K
03	O		D						Ø2 N22							No	Yes	3	C	L
04	O													1	'D'	No	No	4	D	M
05	O																			
06	O																			

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Constant or Edit Word

To complete this problem we need to list any records that are to be updated or flagged but could not be found. The list is to look like this.

```
                RECORDS NOT FOUND

SUBSCRIBER

235476      ADDRESS CHANGE

100114      ADDRESS CHANGE

767676                          DELETION
```

When should we print these items? Since the list shows records not found, we know that they are to be printed when chaining failed to find them. Will the Resulting Indicators assigned to the chain operations be on or off when a record is not found?

If a desired record is not found during a chain operation, the resulting indicator is turned ON. Keep this in mind so that we can describe the output report.

The desired listing lends itself to the use of Auto Report and you should make use of it here. I've listed rules for you to follow as you describe this output.

1. Report Title

    a. Specify the output filename (7-14).

    b. Specify H (heading) as the file type (15).

    c. Specify *AUTO as the field name (32-36).

    d. On the next coding line specify the actual title as a constant (45-70).

        1) Enter a single quote mark in 45.

        2) Enter the title.

        3) Enter a single quote mark behind the title.

2. Column Heading, Data Fields and Constants

   a. Specify D (detail) as the file type.

   b. Specify "2" for space after (18) to provide double spacing.

   c. Specify one set of indicators in 23-31 (use an input indicator and a resulting indicator).

   d. Specify *AUTO as the field name (32-36).

   e. Since we want to list 2 kinds of errors, use the next coding line and

      1) enter OR in 14-15

      2) enter the second set of indicators in 23-31.

   f. On the next coding line

      1) secify the subscriber number field

      2) specify the column heading as a constant

   g. On the next coding line

      1) specify the set of indicators (23-31) that correspond to the desired message

      2) specify the message as a constant.

   h. Finally, on the next coding line

      1) specify the other set of indicators

      2) specify the second message as a constant.

At this time refer to these rules and your coding sheets in order to complete your coding.

* * *

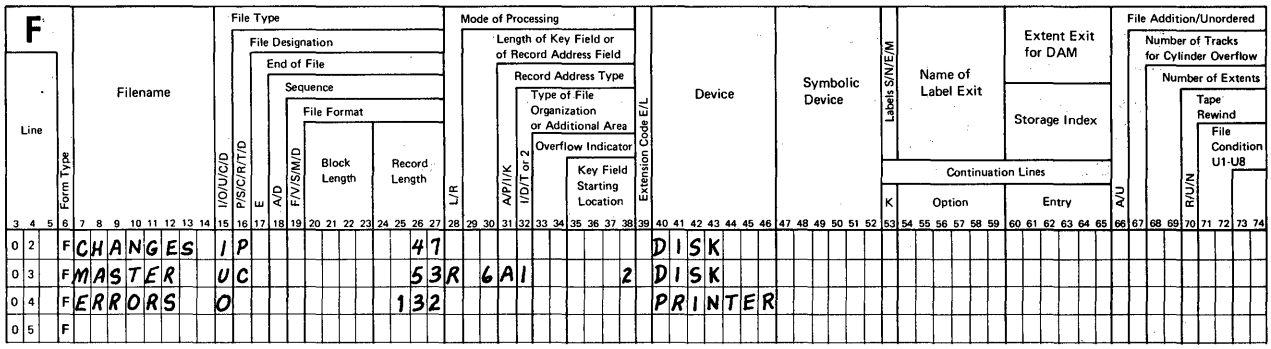
Do you know why no indicators need to be specified for printing the subscriber number field? The reason is that both sets already describe the entire record and subscriber number is found in every printed detail line.

The only time you need to specify indicators for specific fields or constants within a record is when they are to be included for specific conditions that are different from the conditions used to control the production of the entire record.

Now take a look at the entire solution.

Problem 4: Book Solution

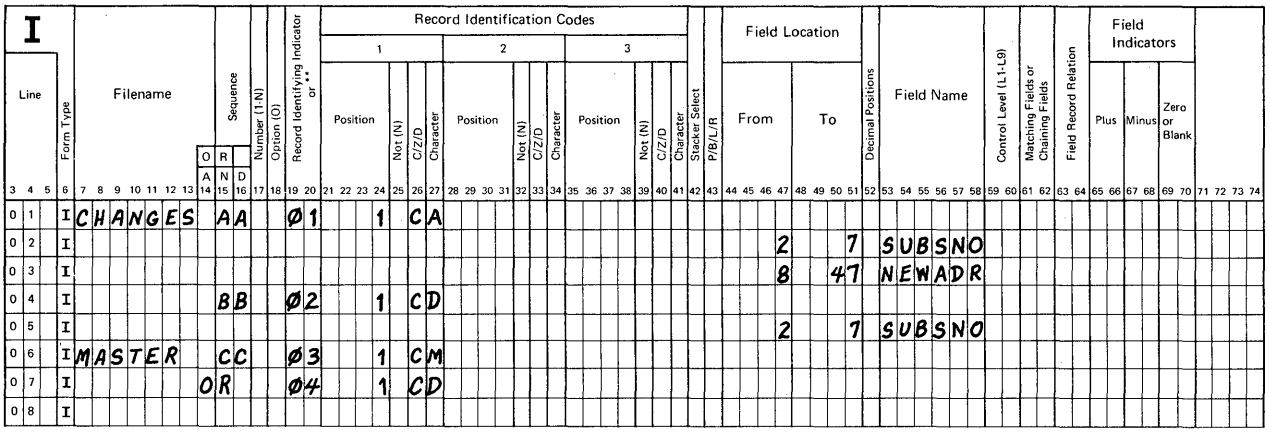
File Description Specification



Line	Form Type	Filename	File Type (I/O/U/C/D)	File Designation (P/S/C/R/T/D)	End of File (E)	Sequence (A/D)	File Format (F/V/S/M/D)	Block Length	Record Length	Mode of Processing (L/R)	Length of Key Field or of Record Address Field	Record Address Type (A/P/I/K)	Type of File Organization or Additional Area (I/D/T or 2)	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	File Addition/Unordered (A/U)	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind (R/U/N)	File Condition U1-U8
02	F	CHANGES	I	P					47								DISK										
03	F	MASTER	U	C					53	R	6	A	I			2	DISK										
04	F	ERRORS	O						132								PRINTER										
05	F																										

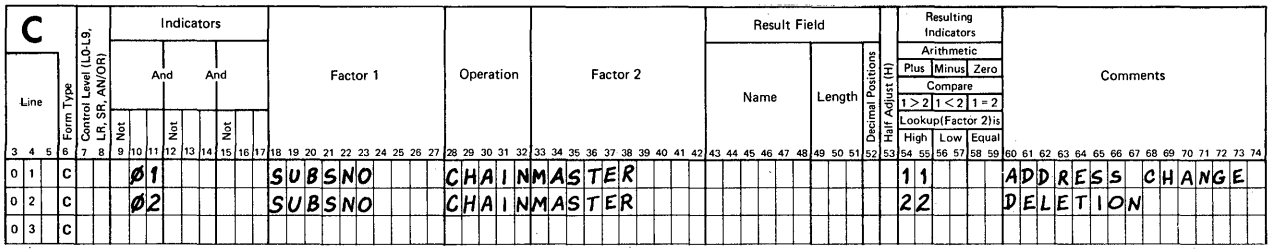
Continuation Lines: K | Option | Entry

RPG INPUT SPECIFICATIONS



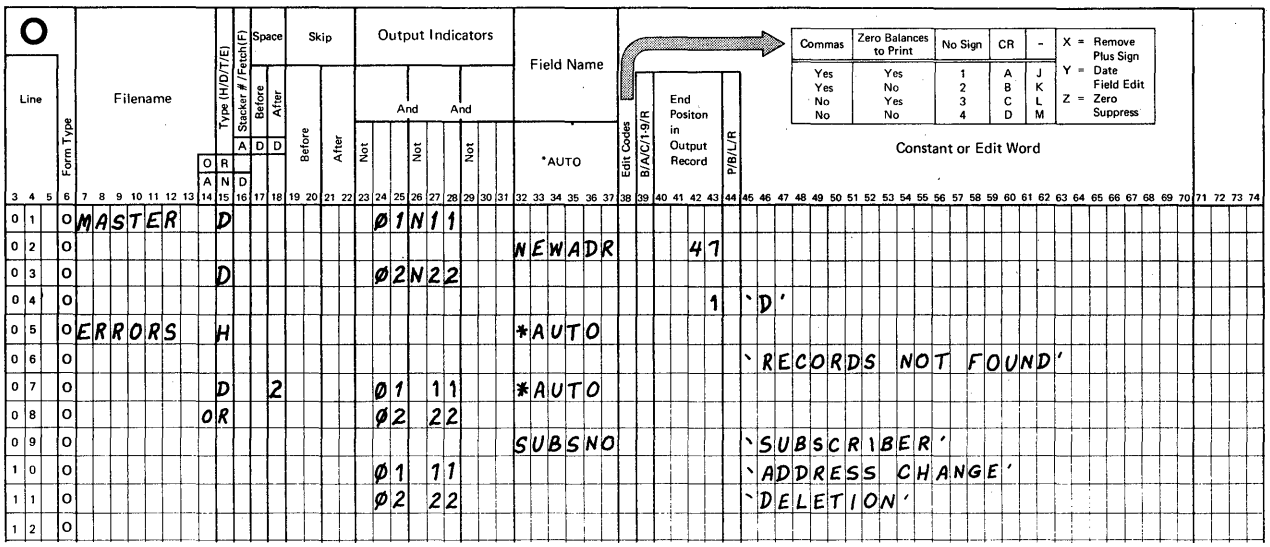
Line	Form Type	Filename	Sequence (OR / AND)	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	CHANGES	AA			01		1	C	A																				
02	I																				2	7		SUBSNO						
03	I																				8	47		NEWADR						
04	I		BB			02		1	C	D																				
05	I																				2	7		SUBSNO						
06	I	MASTER	CC			03		1	C	M																				
07	I		OR			04		1	C	D																				
08	I																													

RPG CALCULATION SPECIFICATIONS



Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Arithmetic: Minus	Arithmetic: Zero	Compare: 1>2	Compare: 1<2	Compare: 1=2	Lookup(Factor 2)is: High	Lookup(Factor 2)is: Low	Lookup(Factor 2)is: Equal	Comments
01	C		01		SUBSNO	CHAINMASTER												11			ADDRESS CHANGE
02	C		02		SUBSNO	CHAINMASTER												22			DELETION
03	C																				

RPG OUTPUT SPECIFICATIONS



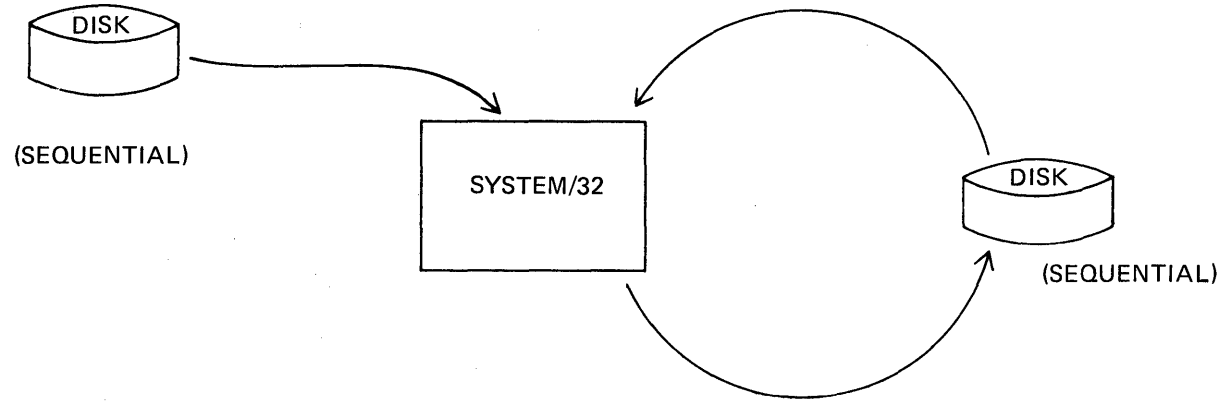
Line	Form Type	Filename	Type (H/D/T/E) (OR / AND)	Stacker #/Fetch(F): Before	Stacker #/Fetch(F): After	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And	Output Indicators: And	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	MASTER	D							01	N11						
02	O											NEWADR			47		
03	O		D							02	N22						
04	O														1		'D'
05	O	ERRORS	H									*AUTO					
06	O																'RECORDS NOT FOUND'
07	O		D		2					01	11	*AUTO					
08	O		OR							02	22						
09	O											SUBSNO					'SUBSCRIBER'
10	O									01	11						'ADDRESS CHANGE'
11	O									02	22						'DELETION'
12	O																

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Problem 5: Use the matching records technique to update records in a sequentially organized disk file.

This problem is another example of file maintenance. It is different from problem four in that we will update values in a field by adding values to it or subtracting values from it. Also, the activity will be done with two sequentially organized disk files.



Because the matching records technique is applied, the following coding relationships need to be remembered.

FILE DESCRIPTION

1. File Designation (16) - Specify one file as primary (P) and the other file or files as secondary (S).

2. Sequence (18) - Specify the sequence of the records; use either A for ascending sequence or D for descending sequence.

INPUT

1. Describe the field or fields that are to be assigned for matching purposes. Do this for each file.

2. Matching Fields (61-62) - If one field is used to control the matching of records, specify M1 for that field in each file; if two or more fields are used, specify appropriate entries (use M2-M9).

RULE: The more significant field is assigned the larger number.

You need to know the following information about calculations and output when the matching records technique is used to solve a problem. First, RPG II programs process one record at a time even though two records are read and compared for matching purposes. Second, a primary record is processed before a secondary record that matches it. Third, when a match does occur, the matching records indicator (MR) is turned ON so that all processing and output controlled by it may be produced.

If at any time a record checked for matching is out of sequence, the program stops running and the operator sees an error message on the display screen.

CALCULATION

1. Indicators (9-17) - Calculations to be performed when a match occurs should include the matching records indicator (MR) along with the desired record identifying indicator.

   Calculations may be desired when a match does not occur. To describe this condition include the letter N (for Not) in front of MR.

OUTPUT

1. Output Indicators (23-31) - Include the matching records indicator (MR) with the appropriate record identifying indicator.

   Again, if output is desired when a match does not occur, include the letter N (for Not).

Refer to all of these coding rules in order to code a solution to problem five.

Problem 5: Update customer records in a sequentially organized disk file by calculating a new account balance for each matched record. Transaction records are in a sequentially organized disk file. Both files are in ascending sequence by account number. To calculate the new balance, add the deposit value and subtract the withdrawal value.

RULE: To describe subtraction,

1. Specify SUB as the operation.

2. Specify the value to be subtracted as Factor 2.

3. Specify the field from which you are to subtract as Factor 1.

4. Specify a Result Field.

When you update a field by subtraction, specify the update field as Factor 1 and as the Result Field.

Before you specify file description entries, you need to consider the question, "Which file should be designated as the primary file?"

Keep in mind that records in the primary file are processed before matching secondaries. For our problem, designate the transaction file as primary.

The records in each file contain the following fields of data.

Transaction File - Each record has 18 characters.

Position	Field
1	Code T
2 - 6	Account Number
7 - 12	Deposit Amount, 2 decimals
13 - 18	Withdrawal Amount, 2 decimals

Customer Accounts File - Each record has 75 characters.

Position	Field
1	Code A
2 - 6	Account Number
53 - 59	Account Balance, 2 decimals

At this time code all of the entries you think are needed to describe a solution to the problem of updating a customer accounts file using the matching records technique.

* * *

Take some time to review your solution before looking at the book solution. Here are some guidance questions to help you.

FILE DESCRIPTION

1. Which file is the input file and which one is the update?

2. Which file is primary and which is secondary?

3. What is the sequence of records in each file?

4. How large are the records in each file?

INPUT

1. Which field is the match field for each file?

CALCULATIONS

1. What combination of indicators is needed to control each operation?

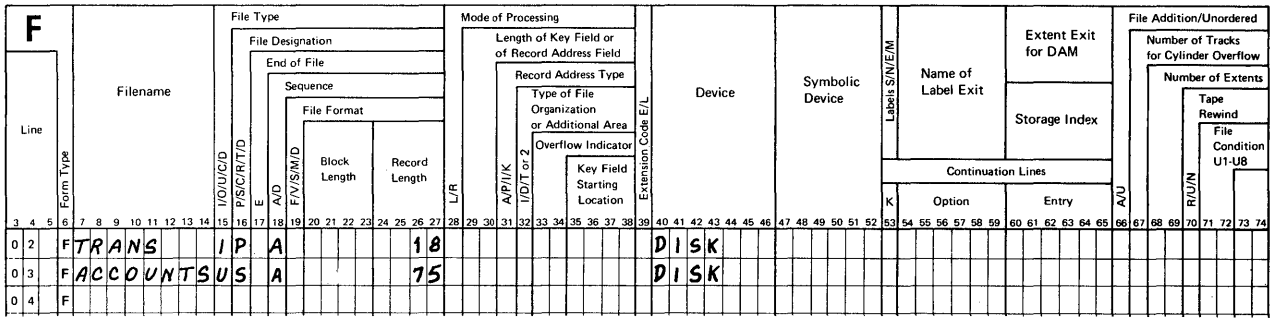
OUTPUT

1. What combination of indicators is needed to control the production of output records?

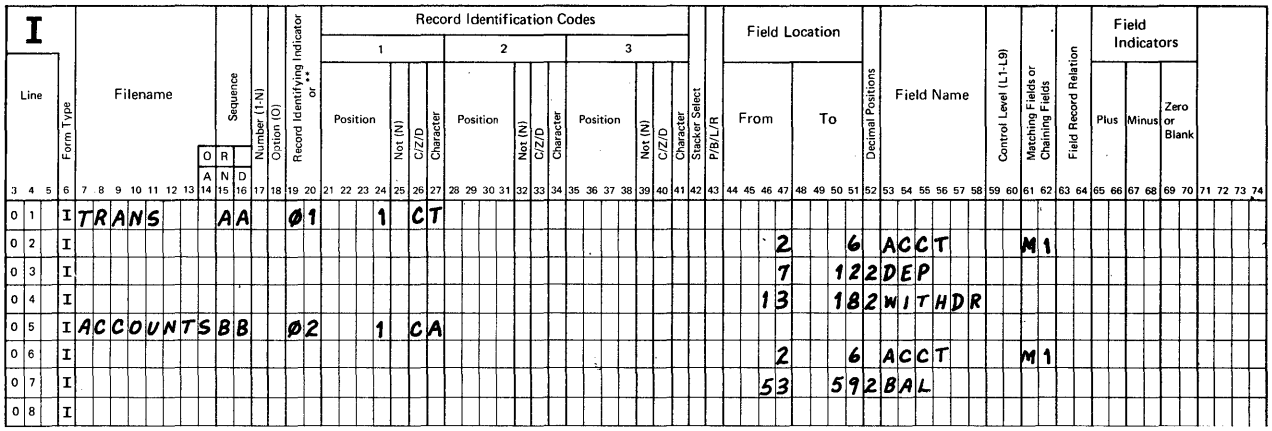
2. How many output fields need to be described to update the record?

OK. Now look at the book solution and verify your solution. Your filenames, field names and record identifying indicators will probably be different, but if they are used correctly that's fine.

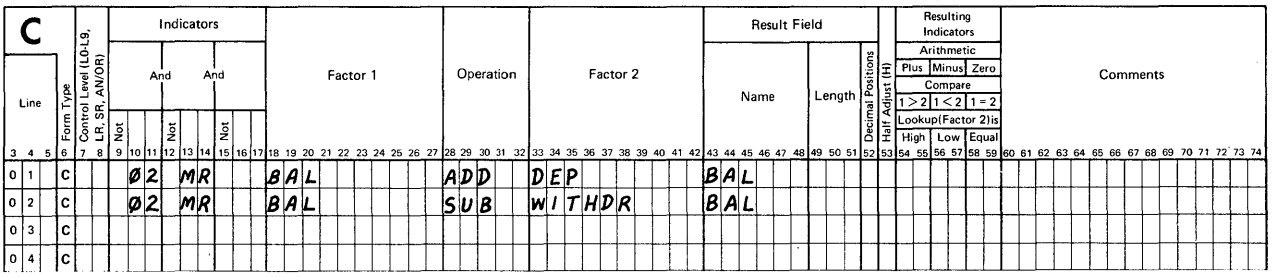
File Description Specification



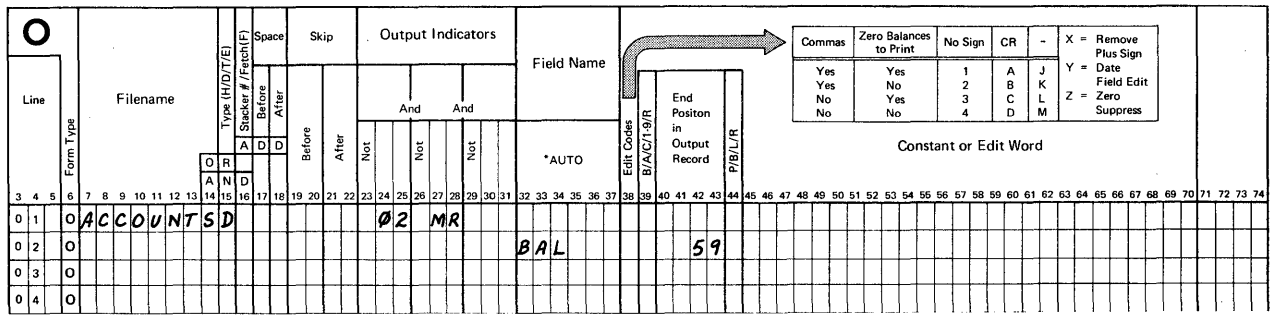
RPG INPUT SPECIFICATIONS



RPG CALCULATION SPECIFICATIONS



RPG OUTPUT SPECIFICATIONS



I did not label that set of entries as a book solution because I want to add two more entries on the file description sheet. First, an entry (E) in column 17 for End of File. Second, an entry for Block Length (20-23) for the update file.

In our program it is possible that the transaction file records update relatively few (let's say 20%) customer accounts. Since this is the case, we should specify an E for End of File for the transactions and make no entry in that position for the customer accounts file. Such an entry allows the program to end when all transactions have been processed. Specify the letter E for your transaction file.

* * *

Now consider entries for Block Length (20-23). Block length specifies the amount of processing unit storage that is to be used as an input or output area. When computer space is available, such an entry speeds up the processing because one read instruction will bring into storage two or more disk records. The more records in a block the fewer times reading (or writing) occurs which results in faster overall job completion. Here are the rules for coding <u>disk</u> block length.

1. Block Length must be a multiple of record length.

2. The maximum entry is 4096.

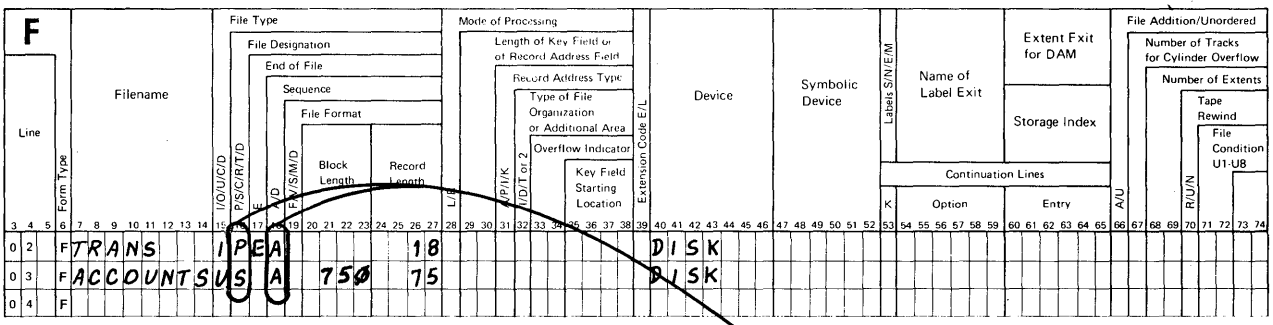
3. You may enter the same number as record length, in which case RPG II will assign an efficient block length.

For our problem assume that we will want to specify a block length equal to 10 records for the customer account file. What entry should be made to do this? Since records are 75 characters long, the block length would be 750. Make this entry for your update file. This completes the solution to problem five.

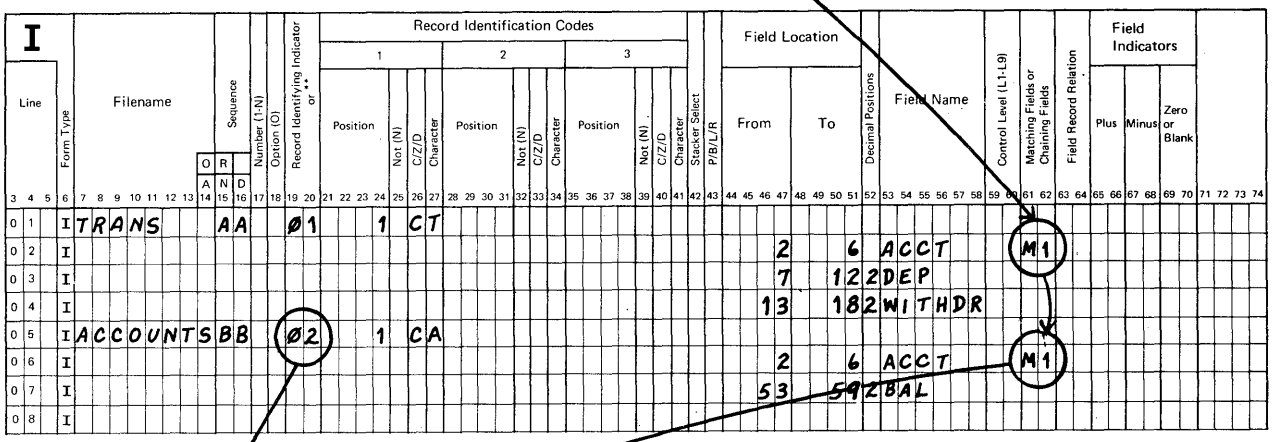
* * *

Problem 5: Book Solution

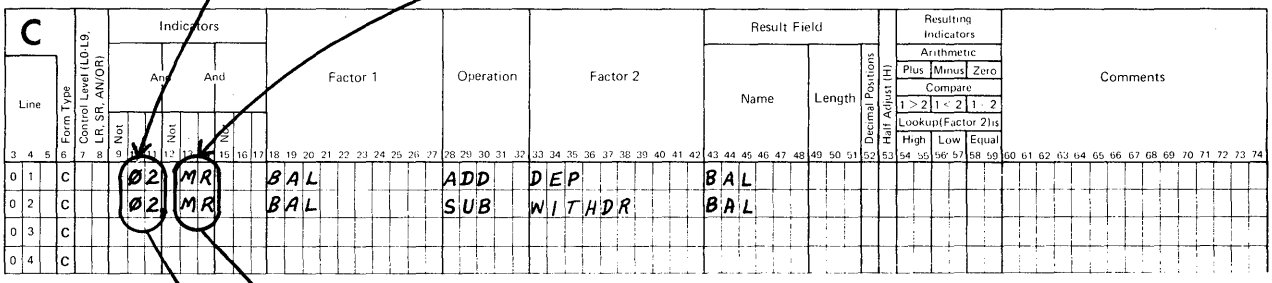
File Description Specification



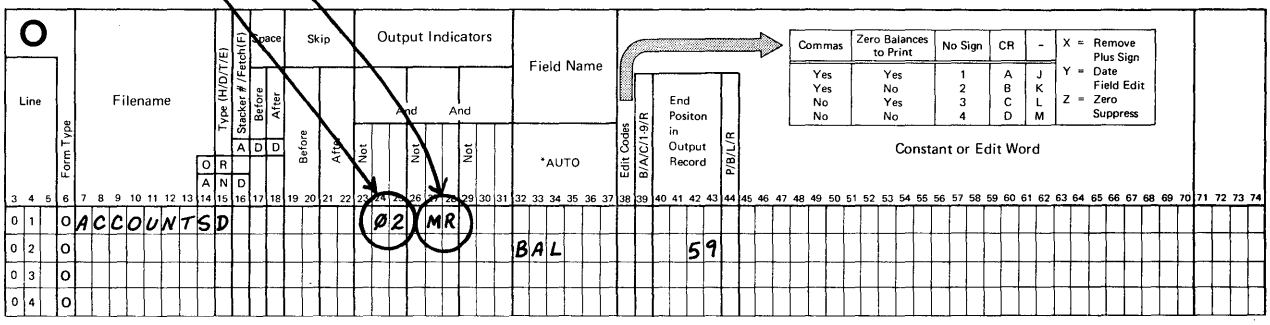
RPG INPUT SPECIFICATIONS



RPG CALCULATION SPECIFICATIONS



RPG OUTPUT SPECIFICATIONS



Chapter 3: Summary

After a disk file has been created, it must be maintained in order to reflect current status of every record it contains. File maintenance involves the processing of files in three ways: addition of new records, updating of information in active records, and flagging for future deletion those records no longer active. A desirable provision of disk file maintenance is the creation of an audit trail, usually a printed report of the actual transactions that occur during a file maintenance run.

By now you should be able to code all of the RPG II entries needed to maintain disk files, which are organized either sequentially or indexed, and include an audit trail of that activity.

Here is a list of the coding entries you learned to use in this chapter.

FILE DESCRIPTION

15	File Type (Update)
16	File Designation (Secondary, Chained)
17	End of File (E)
18	Sequence (Ascending)
20 - 23	Block Length (Disk)
28	Mode of Processing (Random)
66	File Addition (A)

INPUT

61 - 62	Matching Fields (M1)

CALCULATION

9 - 17	Indicators (MR for matching records)
28 - 32	Operation (CHAIN, SUB)
54 - 55	Resulting Indicators (for chaining)

OUTPUT

16 - 18	Add records to a file (ADD)
23 - 31	Output Indicators (MR for matching records)
23 - 31	Output Indicators (N, for Not)

SELF TEST - Chapter 3

Use a piece of scratch paper to record your answers to these questions. You should be able to answer them from memory. When you are finished, check your answers against those found on the next page.

1. What is meant when we say we are going to "update a file"?

2. True or False?

   An indexed file may be updated by using random processing and chaining.

3. True or False?

   The same record identifying indicator must be assigned to every input record type described for one job.

4. When should you specify an input field as being numeric?

5. What is the purpose of the CHAIN operation?

6. How should you specify that an operation is to be performed when indicator 05 is ON and indicator 09 is OFF?

7. When you describe the addition of records to a file, what special entry must be made on the File Description sheet? What special entry must also be made on the Output sheet?

8. True or False?

   When records are flagged for deletion using RPG II, they are erased from the disk.

ANSWERS - Chapter 3 Self Test

Compare your answers to the ones below. If you missed more than three, re-study the chapter before continuing.

1. To update a file means that the contents of one or more fields in a record are changed. Normally this is done by either adding or subtracting values in numeric fields, or by replacing alphameric field data.

2. True. An indexed file may be updated by using random processing and chaining.

3. False. Each input record type must have a unique indicator assignment.

4. There are two situations when you should specify an input field as being numeric. One, if that field is to be used in a calculation step. Two, if that field is to be edited on output.

5. The CHAIN operation causes a search of an indexed file in order to locate a record having the same key field data as that contained in the searching field. The search relates a record from one file to a record in another file.

   (Your answer should be similar to this in order to be correct.)

6. Use either 05N09 or N09 05 as a correct combination of indicators.

7. File Description sheet - Specify an A in position 66.

   Output sheet - Specify the word ADD in positions 16-18.

   This may be hard to remember at this point in the course. If you remembered, fine, If not, don't be upset.

8. False. When we speak of deleting records, we mean that they are "flagged" for later deletion, they are not erased from the disk at this time.

Rate Yourself:

8 = Super! 7 = Outstanding! 6 = Very Good 5 = Good

Chapter 4: Using /COPY, Operation Codes, Level 1 Control, RPG II Generated Program Logic (1.5 to 2.5 hours)

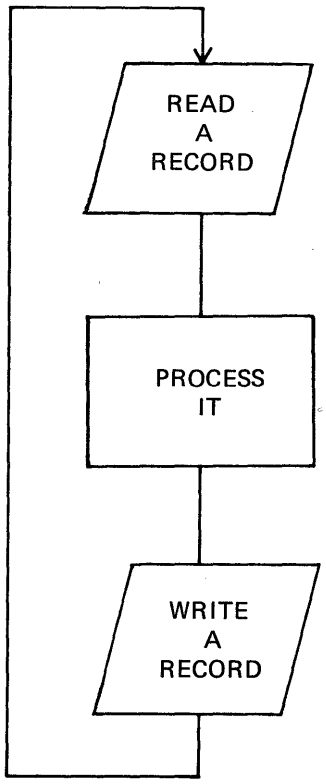
This chapter contains information about a variety of RPG II topics, all of which are useful to know about and you should be able to use when coding problem solutions. Since you will be coding a set of practice problems in the next chapter, these topics are covered now in order to facilitate their use when you need them.

The first topic, /COPY, is a function of the Auto Report feature. It is used to copy and make a part of your program, sets of coded RPG II specifications that are stored in a disk library.

The second topic, Operation Codes, presents rules and examples of selected, commonly used operations. After you know of their existence and have learned the rules for their use, you will be expected to include them as needed in the practice exercises.

The third topic, Level 1 Control, provides a means for processing groups of related records rather than processing individual records only.

The last topic, RPG II Generated Program Logic, is presented here to solidify your knowledge about describing files, records and fields in such a way that a meaningful program can be generated. Every RPG II generated program contains essentially the same sequence of data processing steps:



By examining an expanded version of this flow you will be able to trace what happens to fields of data as input, processing and output steps take place.

CHAPTER OBJECTIVES

When you have finished studying this chapter you should be able to:

1. use the /COPY function to reduce coding effort and reduce potential coding errors,
2. select and use appropriate operation codes,
3. specify all entries needed to identify groups of records, control their processing, and produce their output, and
4. predict what happens in the computer as each type of input record is read and processed, as well as what kind of output can be produced.

Various examples are presented in which these items will appear.

/COPY

MULT

DIV

MVR

SQRT

Z-ADD

Z-SUB

MOVE

MOVEL

GOTO

TAG

COMP

Level 1 Control (L1)

In addition to this text, you will need a few blank coding sheets of each type (File Description, Input, Calculation and Output).

A self test is included for your review at the end of the chapter.

The /COPY Function

The purpose of the copy function is to provide for the inclusion of RPG II statements that have previously been described and cataloged for reference. A copy statement is described in your program wherever you wish to have the cataloged set of RPG II statements included in your program. Let's look at an example.

Let's assume that some time in the past you described an input record with 5 fields, and then had these six RPG II Input statements cataloged (stored on the disk) with the name SALRY assigned for future reference.

SALRY

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or *,*	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	PAY	NS			Ø1	1		C	P																				
02	I																				2	10	Ø	SS						
03	I																				28	29	2	RATE						
04	I																				36	50		NAME						
05	I																				51	54	Ø	NUM						
06	I																				58	62	2	BASE						
07	I																													
08	I																													,
09	I																													

Now you are required to code another problem solution that uses the same record and fields as input. Since these RPG II statements are cataloged, all you do is code one copy statement as your Input specifications. When your entire program is coded and ready for compilation, the copy statement will be analyzed after which the cataloged input specifications will be located and inserted for compilation. Look at this coding example.

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1 (Position, Not (N), C/Z/D, Character)	Record Identification Codes 2 (Position, Not (N), C/Z/D, Character)	Record Identification Codes 3 (Position, Not (N), C/Z/D, Character)	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	
3-5	6	7-13	14-16	17	18	19-20	21-27	28-34	35-41	42	43	44-47	48-51	52	53-58	59-60	61-62	63-64	65-66	67-68	69-70	71-74
01	I	/COPY F1,SALRY																				
02	I																					
03	I																					
04	I																					

The copy statement as shown causes the six cataloged RPG II statements (from the previous illustration) to become a part of your source program when compilation begins.

Here are the coding rules for a copy statement.

1. Positions 7-11 must contain /COPY.
2. Position 12 must be blank.
3. Positions 13-15 must contain F1 (this code directs RPG II to find the set of cataloged statements on the disk).
4. Position 15 must be a comma.
5. Positions 16-23 must contain the name of the cataloged set of RPG II statements to be copied. This name must start in position 16 and may be from 1-8 characters long.
6. If you wish, you may enter a descriptive comment in positions 50-80.

This example illustrates the use of the /COPY function to copy a previously described and cataloged input record and its fields.

You are permitted to copy any cataloged RPG II statements, not just Input entries. For example, you may have cataloged a set of calculations that are exactly what you need to use in a new problem solution. How can you make those cataloged calculations a part of the new program? What coding rules must be followed? What is the benefit to you, the programmer, in using this approach?

1. To use the cataloged calculations, code a /COPY entry on a Calculation sheet.

2. The coding rules for /COPY statements are the same regardless of the type of sheet on which they are made.

   /COPY in positions 7-11

   leave position 12 blank

   F1 in positions 13-14

   , in position 15

   ..... use the name of the cataloged set in 16-23

3. One benefit is reduced coding time.

OK. How would you copy a set of eight Output specifications cataloged under the name RETALOT? Assume that these eight entries are what you need in your program. Use a blank Output sheet and specify a copy statement to do the job.

* * *

Your entry must include:

1. /COPY in 7-11

2. a blank in 12

3. F1, in 13-15

4. RETALOT in 16-22

You may have included a comment in positions 50-80, but it is not required.

Here is a rather unusual, but possible, situation. Using blank coding forms, specify entries that will copy a set of input entries named INEX25, a set of calculations named C6, and a set of output entries named OUTTEM.

* * *

Remember, you are permitted to copy any cataloged RPG II statements. You may also specify standard RPG II statements in addition to those you have copied. A benefit to you lies in reduced coding effort. Assuming that cataloged items have already been tested and are correct, another benefit is that these copied program steps are pretested. Also, there are fewer statements in your program that need to be keyed into the system for compilation. This reduces the time needed to enter programs and reduces the number of potential keying errors prior to compilation.

One more point. When the source program is compiled, the RPG II program will include each /COPY statement followed by the actual statements that were copied from the library on the disk where they are cataloged.

Your coded entries should be similar to the following. Check your entries.

```
I/COPY F1,INEX25

C/COPY F1,C6

O/COPY F1,OUTTEM
```

There is additional information related to the use of /COPY in the RPG II reference manual. I suggest that you look through this section at your convenience. Keep in mind that we have only begun the study of the RPG II language and some of the examples in the reference manual contain specifications we have not learned yet, for example, copied statements may be modified!

One last point on coding /COPY. The copy function is an aid to the programmer and the person who keys in source programs for compilation. There are restrictions to its use as it isn't the "cure-all" for problem solving. If a set of cataloged entries can be used as is or requires little modification, use /COPY. If not, describe all entries in standard RPG II form.

Operation Codes

This part of the chapter deals with specification of selected operation codes. We will present some information, show a few examples, and direct you to read in detail about specific operations. As you read, you should refer to these questions mentally.

1. What is the operation?
2. What does it do?
3. Are there any restrictions or considerations regarding its use?
4. How do I specify its use?

You will need a number of blank Calculation sheets and your RPG II reference manual, form number SC21-7595.

* * *

As the name of this form implies, it is the place where you describe calculations. We have already showed examples of problem solutions in which the ADD, SUB and CHAIN operations were used. At this point, we want to concentrate on arithmetic operation codes. Before we conclude this section, you should know how to describe the operations listed below:

1. ADD — Adds the values in 2 factors.
2. SUB — Subtracts one factor's value from another.
3. MULT — Multiplies the values in 2 factors.
4. DIV — Divides one factor's value by another.
5. MVR — Moves the remainder following a division operation.
6. SQRT — Computes the square root of a factor.
7. Z-ADD — Sets a result field to zero and then adds a factor to it.
8. Z-SUB — Sets a result field to zero and then subtracts a factor from it.

Here are some general rules to be remembered when using arithmetic operations.

1. Any factor to be used in an arithmetic operation must be either a numeric field or a literal. A literal is a specific number such as 1.732 or -275.
2. The result field holds the answer following an arithmetic operation. The result field must also be numeric and large enough to hold the largest anticipated answer during a job.
3. The length of any numeric field cannot exceed fifteen (15) positions. This field may have from 0 to 9 decimal positions.
4. Data that was stored in a result field is replaced by new data the next time that same operation statement is performed.
5. The result always contains a signed number.

6. The result may be "half adjusted" except for a (DIV) operation that is immediately followed by a move remainder (MVR) operation.

   "Half adjusted" means the same as "rounded off".

At this time, you should turn to the section of the reference manual that describes arithmetic operations and read about each of the following: Add, Zero and Add, Subtract, Zero and Subtract, Multiply, Divide, Move Remainder, and Square Root. This information is found in the second section of the manual.

* * *

Now that you have completed the reading, look at the coded examples on the next page and imagine that you are explaining to another person what each statement means. If you cannot explain it, re-read the reference manual section before continuing.

Operation Codes: Examples

RPG CALCULATION SPECIFICATIONS

	Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Resulting Indicators: Compare: 1>2	1<2	1=2	Resulting Indicators: Lookup(Factor 2)is: High	Low	Equal	Comments
	01	C				XYZ	ADD	TVAL	ST55		80											
(1)	02	C																				
	03	C				COLOR	ADD	1	COLOR													
(2)	04	C																				
	05	C				MIST	SUB	772.20	NOTER		62											
(3)	06	C																				
	07	C				A	DIV	16	F27		40	H										
(4)	08	C																				
	09	C				S	MULT	S	S2													
(5)	10	C																				
	11	C		N08			SQRT	FORML	ROOT													
(6)	12	C																				
	13	C		N10		TOTVAL	DIV	FREQ	MIX		31											
(7)	14	C					MVR		LEFT		20											
(8)	15	C																				
	16	C					Z-ADD	52	WKS													
(9)	17	C																				
	18	C					Z-SUB	-29	VAL													
(10)	19	C																				
	20	C				12	SUB	X	TX		20											
		C																				
		C																				
		C																				
		C																				
		C																				

1. What do you think would happen to the value in the result field if you switched the entries for factors 1 and 2 in both ADD statements?

2. If the value in MIST in statement #3 equals 700.00, what sign will be in the result field after the operation?

3. Why is the H specified in column 53 on statement #4?

4. What kind of problem is being described in statement #5?

5. Why do you suppose that the entry NØ8 is coded in positions 9-11 for statement #6?

6. Why is N1Ø coded in positions 9-11 for statement #7?

7. Consider statements 7 and 8. Can the field named MIX be half adjusted?

8. What value will be found in WKS after the Z-ADD operation is completed?

9. What sign will be found in VAL after the Z-SUB operation is completed?

10. If the value of X is 12, what value will be found in TX after subtraction takes place for statement #11?

* * *

If your answers do not agree with mine, you may wish to re-read the section on arithmetic operations before you continue.

1. No change. Exchanging entries for factors 1 and 2 in an ADD operation does not change the value of the result.

2. The value of NOTER will be negative because 772.20 is larger than 700.00. NOTER will have a minus sign.

3. The result field value is to be half adjusted after division is performed.

4. The value in the field named S is being multiplied by itself. This is the way you specify the squaring of a number.

5. The square root operation can only be performed if the value in factor 2 is positive. Evidently a test to determine this condition results in indicator Ø8 being turned ON when FORML has <u>negative</u> value, therefore, the entire operation shall be done when indicator Ø8 is <u>not</u> <u>on</u>!

6. Similarly, division cannot be done when factor 2 (FREQ) is zero (indicator 10 is ON). So, statement #7 means if factor 2 is not zero, divide TOTVAL by FREQ.

7. No. When MVR is specified, the result field of the divide operation just ahead of it cannot be half adjusted.

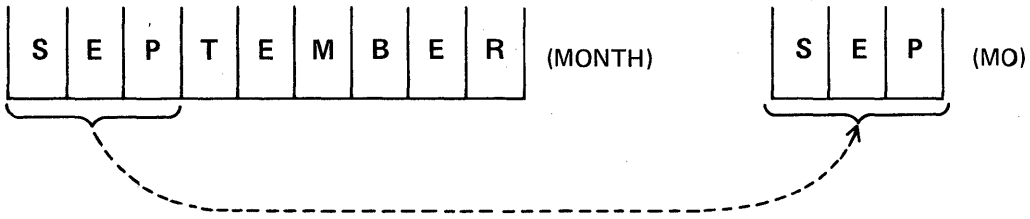
8. 52, because WKS is set to zero and then 52 is added to it.

9. A plus sign because VAL is set to zero and then -29 is subtracted from it. When a negative number is subtracted, the rule is change its sign and then add it.

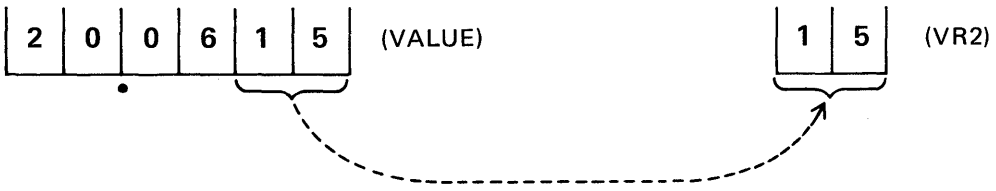
10. Zero

The next group of operations provides for the movement of data within the central processing unit storage area.

Suppose the computer read an input record that included a 9-position field for the name of the month in which the record was created, but in your new problem you simply want the first 3 letters of each month's name. By moving the first three characters of the month field to a new place in storage, you can now proceed to solve your problem in which abbreviations are to be used.



Here is another example. Let us say you are computing a value to four decimal places and you need to use the rightmost two decimal places for further calculations. You can use a move operation to extract those two decimal places.



At this time, you are to concentrate on only two of the move data operation codes, MOVE (move right-to-left) and MOVEL (move left-to-right). Here are some general rules about moving data.

1. Data in Factor 2 is moved to the Result Field. No entry is specified for Factor 1.

2. Specify as Factor 2 the name of the field from which data is to be moved. Factor 2's value is not changed by the operation.

3. The Result Field contains the characters that were moved.

4. The shorter field controls the operation. That is, if Factor 2 is shorter in length than the Result Field, moving of data stops when all characters in Factor 2 have been moved. If the Result Field is shorter, moving of data stops when the Result Field has been filled.

5. You may move numeric data to either a numeric or alphameric result field.

6. You may move alphameric data to either a numeric or alphameric result field.

7. MOVE is the operation code that causes data to be moved from right to left from factor 2 to the result field.

8. MOVEL is the operation code that causes data to be moved from left to right from factor 2 to the result field.

At this time, turn to the section of your RPG II reference manual that explains the move data operations. Concentrate on the examples that describe either MOVE or MOVEL. When you think you understand their uses, return to this text.

* * *

Refer to the following information in order to answer these questions.

Field A contains 2756

Field B contains MR

Field C contains X15Y29

Field D contains SYZYGY

In each example, think about what is specified and then figure out what would be stored in the result field after the operation was completed.

RPG CALCULATION SPECIFICATIONS

C	Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus, Minus, Zero); Compare (1>2, 1<2, 1=2); Lookup(Factor 2)is (High, Low, Equal)	Comments
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
01		C				MOVE A		HB	22				
02		C				MOVELA		XM1	3				
03		C				MOVELB		C					
04		C											
05		C				MOVELB		C					
06		C											
07		C				MOVE D		RT	B				
08		C											
09		C				MOVELD		M	1				
10		C											
11		C											
12		C											

* * *

After the move operations are completed, the result field values are:

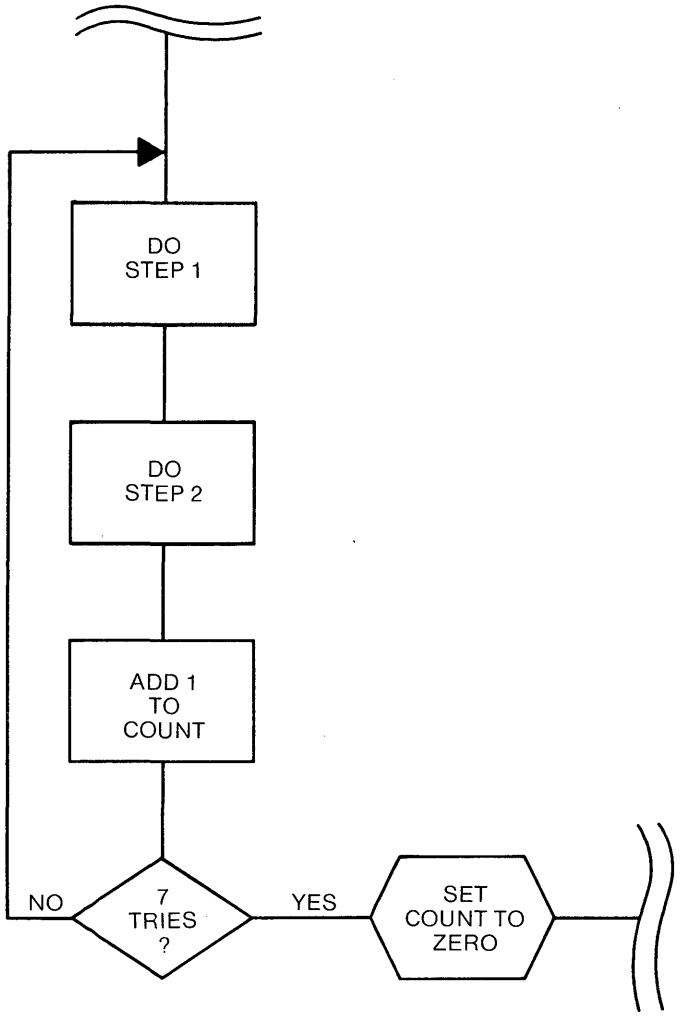
1. HB = 56 (with a plus sign)
2. XMI = 275
3. C = MR5Y29
4. RT = ??SYZYGY The ?? positions would contain whatever was there before the move was done.
5. M = S

A key point about the use of move data operations was made earlier. These operations provide for the movement of data within central processor storage. Moving data requires time and space in the computer, so do not use these instructions if your problem can be solved in a more efficient manner.

The third category of operations to be studied in this section includes GOTO, TAG and COMP. The operation GOTO directs the computer to either bypass one or more calculations, or to repeat a previous set of calculations. The TAG statement identifies the place on the calculation sheet where the next set of instructions shall begin. The COMP (compare) operation provides a test for a desirable condition.

Let's look at a simple example. Suppose we are supposed to repeat a 2-step calculation series seven times and then continue normal processing.

First, we could perform the desired 2-step calculation. Next, we could add 1 to a counter in order to keep track of the number of times we have done those steps. Third, we could test the counter value to see if we have done the series seven times. If not, we repeat the steps and add 1 to the counter. When we have done it seven times, we reset the counter to zero and continue normal processing. Look at the flowchart.



Here are the calculation statements we would use to describe this set of actions.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators High (1>2)	Low (1<2)	Equal (1=2)	Comments
01	C			STEPS	TAG									
02	C			A	MULT	X	AX	1Ø	Ø					
03	C			AX	ADD	6	A							
04	C			COUNTR	ADD	1	COUNTR	1Ø						
05	C			COUNTR	COMP	7							65	
06	C		N65		GOTO	STEPS								
07	C				Z-ADD	Ø	COUNTR							
08	C													

In this example the GOTO Operation was used to make the computer repeat a set of instructions. To compare a value in a field to a number, just specify that number as a numeric literal. The 7 in Factor 2 of the COMP statement is a numeric literal.

Now examine the next set of calculations. In this problem there are three situations. The field named CODE should contain either the letter A, or the letter B or the letter C.

CODE = A or B or C

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus / Minus / Zero; Compare 1>2 / 1<2 / 1=2; Lookup(Factor 2)is High / Low / Equal	Comments
01	C			CODE	COMP	'A'					71	
02	C		71		GOTO	SETA						
03	C			CODE	COMP	'B'					72	
04	C		72		GOTO	SETB						
05	C			CODE	COMP	'C'					73	
06	C		73		GOTO	SETC						
07	C				GOTO	END						
08	C			SETA	TAG							
09	C											
10	C											
11	C				GOTO	END						
12	C			SETB	TAG							
13	C											
14	C											
15	C											
16	C				GOTO	END						
17	C			SETC	TAG							
18	C											
19	C											
20	C											
	C											
	C			END	TAG							
	C											
	C											
	C											

This problem is more complex than the first one. Notice how letters are used as literals in the COMP statements. They are called alphameric literals.

RULE: When letters, or letters and numbers are used as literals, they must be enclosed by single quote marks.

Did you try to trace each condition step-by-step? If not, try it. Ask yourself these questions as you do the tracing.

1. If the field named CODE has a letter A in it, which steps will be taken?

2. If CODE has a B, which steps will be taken?

3. If CODE has a C, which steps will be taken?

4. What will happen if a record is read with a character in CODE that is neither an A, a B or a C?

* * *

The GOTO operation was used here to bypass some calculation steps. Now let's list some general rules for the use of GOTO and TAG operations.

1. A GOTO operation may be conditioned by 1, 2 or 3 indicators in positions 9-17.

2. The TAG operation <u>cannot</u> be conditioned by indicators.

3. The name used as Factor 2 in a GOTO statement <u>must</u> be the same as a name used as Factor 1 with a TAG statement somewhere in the same program. The "name" is called a label.

4. Each TAG statement must include a unique label.

5. Any number of GOTO statements may direct the computer to branch to the same TAG statement.

6. A TAG statement performs no activity. It simply provides a label for reference by GOTO statements.

Here are some general rules for using the compare (COMP) operation. Refer to a blank Calculation form as you read these rules.

1. A field in Factor 1 is compared against a field or a literal in Factor 2.

   A literal in Factor 1 is compared against a field in Factor 2.

2. At least 1 Resulting Indicator must be assigned in columns 54-59 in order to determine the results of the comparison.

3. If the value of Factor 1 is greater than the value in Factor 2, the indicator assigned to positions 54-55 will be turned ON.

4. If the value of Factor 1 is less than the value in Factor 2, the indicator assigned to positions 56-57 will be turned ON.

5. If the value of Factor 1 exactly equals the value of Factor 2, the indicator assigned to positions 58-59 will be turned ON.

6. Any indicator turned ON as a result of a COMP operation remains ON until that same statement is repeated, or the indicator is turned off (we'll cover the turning off of indicators in a later chapter).

7. No field values are changed in this operation.

8. No Result Field is specified.

9. Comparison is based upon algebraic rules for signed number sequences for numeric fields and upon collating sequences for alphameric fields.

At this point, read those sections of the RPG II reference manual that deal with GOTO, TAG and COMP operations.

* * *

Use a blank Calculation sheet to code a solution to the following problem. You are to include at least one GOTO, one TAG and one COMP statement to solve the problem. You may refer to this book or the RPG II reference manual if you wish. Sketch a flowchart if you wish before you code the solution.

Problem

Divide the field named EN by the field named XWON to compute a result named T5. Add XWON to T5 to get a new value for the same field named T5. Divide T5 by 2 and place the answer in the field named XTU. If the value in XTU is not equal to the value in XWON, replace the value in XWON with the value in XTU and repeat all of these calculations. When the value in XTU equals the value in XWON, the calculations are finished.

* * *

A solution is provided for your review. Examine it carefully. Also, examine your solution. To find out whether or not your solution is correct, trace the values for each field on a piece of scratch paper.

Test Data

Starting value of EN is 64.

Starting value of XWON is 4.

Assume that the field T5 has 1 decimal place.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (Plus, 1>2)	Resulting Indicators Low (Minus, 1<2)	Resulting Indicators Equal (Zero, 1=2)	Comments
01	C			REPEAT	TAG									
02	C			EN	DIV	XWON	T5	3	1					
03	C			T5	ADD	XWON	T5							
04	C			T5	DIV	2	XTU	2						
05	C			XTU	COMP	XWON							15	
06	C		N15		Z-ADD	XTU	XWON							
07	C		N15		GOTO	REPEAT								
08	C													

Here are key points in solving this problem.

1. If you expect to repeat a series of steps, allow room for a TAG statement just ahead of the first calculation in the series.

2. Each step must appear in the correct order, from top to bottom on the Calculation sheet.

3. Any unused indicator may be assigned to test the results of the comparison.

4. The label in Factor 1 of the TAG statement must be the same as the label in Factor 2 of the GOTO statement.

5. The value in the field named T5 was changed by adding a value to it and storing the result in T5. This is the method used to accumulate values.

Well, how was your solution? Did you include all of the calculation steps in the correct order? Did your label in the GOTO and TAG statements start with a letter? When you traced the test data through either your solution or mine, what value did you get for XWON when the calculations were finished?

If your solution (and mine) works properly, the value in XWON should be 8 when all calculations are finished for the test data shown.

Coding Level 1 Control Breaks

Each of the examples we have been working with up to this point in the course dealt with the processing of individual records in a file. At this time, we will study the processing of groups of records.

Refer to this list of student records as we talk about them in groups and consider the RPG II coding required to describe these groups.

RECORD	CLASS	NAME	AGE
1	7	BILL	11
2	7	ANN	11
3	8	SAM	13
4	8	JANE	15
5	9	SALLY	14

These students are already arranged in order by class number. Let us say that we are to count and then print the number of students in each class and a total of all students at the end of the report. What information is needed to solve the problem?

First, we need to be able to describe one field of data as a "control field", that is, the field that is to be examined as each record is read by the computer in order to determine which records belong to the same group. RPG II provides special Input sheet entries to describe (or define) a field as a "control field". Look at a blank Input sheet, positions 59-60. Acceptable entries for these columns are L1-L9. When an entry such as L1 (level 1) is made in these columns, it "defines" a field as a control field of a designated level. Level 1 is the lowest level of control in RPG II and level 9 is the highest level that can be specified.

RULE: To define a level 1 control field, specify L1 in positions 59-60 on the Input sheet for the field that controls the grouping of records for the job being run.

Now that the control level field has been defined on the Input sheet, we may control both the processing and output of data by specifying L1 as the controlling indicator for particular calculations and/or output statements.

Here is how it works. First, the indicator L1 automatically will turn ON when the contents of the control field in a record are different from the contents of that same control field in the record just before it. Second, the indicator L1 automatically turns OFF when all processing and output specified under its control have been produced. Once you define a field as being a control field, the program compiled by RPG II examines that field as each record is read, compares it to the previous record (which it stores for comparison), turns the control level indicator on if necessary, and later turns it off when its activity is completed. Yes, the program does all of this if you enter L1 (or L2-L9) in 59-60 on the Input sheet.

Now consider the desired output. Here is a copy of the kind of report we expect to print.

```
     STUDENT TALLY

CLASS          # OF STUDENTS

  7                    2

  8                    2

  9                    1

                       5*
```

When should totals be printed for each class? When should the final total be printed?

* * *

Each record must be counted. All class 7 records are counted as they are read. When the first record for class 8 is read, RPG II turns on indicator L1. Why? Next, RPG II will produce as output all report fields, totals and constants you described as belonging to class 7. The indicator L1 remains off until when? What is done by RPG II when L1 goes on again? Then what happens?

I have asked a lot of questions. See if you are on the right track so far. The indicator L1 turns on when the first record for the next class is read because the class number field is different from the class number of the record just before it. After the calculations and output are completed for class 7, indicator L1 turns off and records for class 8 are counted. When the first record for class 9 is read, indicator L1 will turn on again. After calculations and output are done for class 8, indicator L1 turns off and the records for class 9 are counted.

By now you might think we have got a few situations to take care of before this data processing problem is really solved. You are right.

Situation #1: How does RPG II handle the situation when the very first record is read in for the entire job? If it compares class 7 against class "nothing", wouldn't L1 be turned ON?

Situation #2: Doesn't the count field have to be set to zero after being printed for one class so we can start counting properly for the next class?

Situation #3: How can we keep a running total of all records counted?

Situation #4: How does RPG II produce output for the very last class (class 9 in this example)? Doesn't L1 have to be turned on for it?

Let's take these situations one at a time. RPG II automatically bypasses both calculations and output statements controlled by L1-L9 indicators when the very first record for the job is read in. This is done so that no "false totals" will appear on reports. No calculations or output statements conditioned by control level indicators will be processed until at least 2 records have been read for a job.

After printing the count field record for one class, you must specify an entry to reset that counter to zero. To do this, you simply enter the code letter B in position 39 on the Output sheet for the count field. This coding causes the count field to be "blanked after producing output" so that counting can start at zero for the next group of records.

A running total of all records is kept by specifying an additional operation that is controlled by indicator L1. This step is needed so that the records counted for a class can be included in a final total.

NOTE: Calculations for a control level always occur before its related output is produced.

We will look at the coding for this example shortly.

To complete a job, a special "end of file" record must be detected by your program. In our example, the special record must follow the last class 9 record. This special record causes all control level indicators (L1-L9) and the last record indicator (LR) to be turned ON when it is read.

NOTE: When the last data record has been keyed from the console keyboard, the operator enters a special "last record" from the keyboard to let the program take care of the final group.

When the last data record from the disk file is processed, one more special "last record" is read in from the same disk file. When disk files are created, the special "end of file" record is automatically added for this purpose.

Now to get back to the processing and output for class 9 records. Since L1-L9 and LR indicators are all turned ON by the special last record, all processing for all groups of records is completed and then all output for all groups is produced, followed by the output of the final total on the report. When all output has been produced, the computer ends this job.

Here are two definitions you need to remember when solving problems that involve groups of records.

1. Detail-time calculations are those which are performed for each input record. They are controlled by a record identifying indicator in positions 9-17.

2. Total-time calculations are those which are performed for each group of records. They are controlled by a control level indicator in positions 7 and 8.

When one or more result fields contain totals for a group of records, they must be set to zero before totals are calculated for the next group. To reset such an output field, specify a "B" in column 39 on the Output sheet following that field name. The value in the field will be set to zero after the field value is printed (or written out on disk).

In order to relate the words to the problem we are doing, I will restate the problem and show a coded solution. When you look at the coding, notice how the L1 entries are made in order to:

1. define the control field on the Input sheet,

2. control the accumulation of a running total on the Calculation sheet, and

3. control the printing of class totals and a final total on the Output sheet.

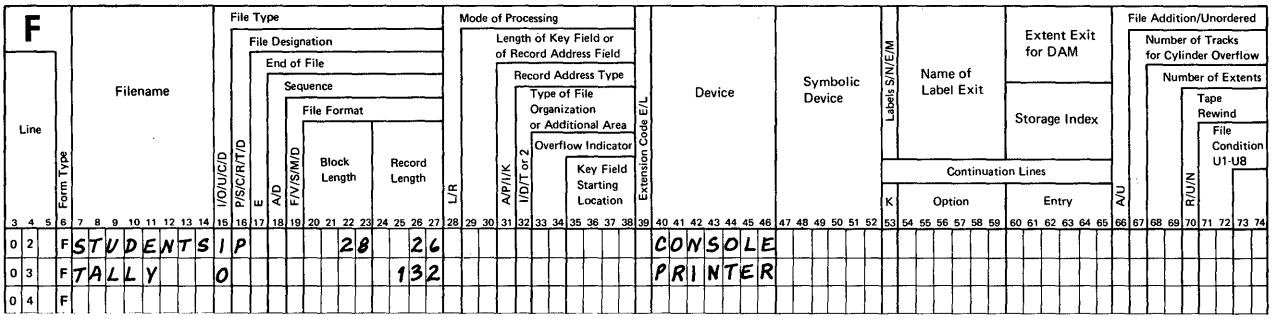
If you are unsure of any coding steps, I suggest that you re-read this section called "Coding Level 1 Control Breaks".

Problem

Print a tally of the number of students in each class. Include a final total of all student records. Input record fields include CLASS, NAME and AGE.

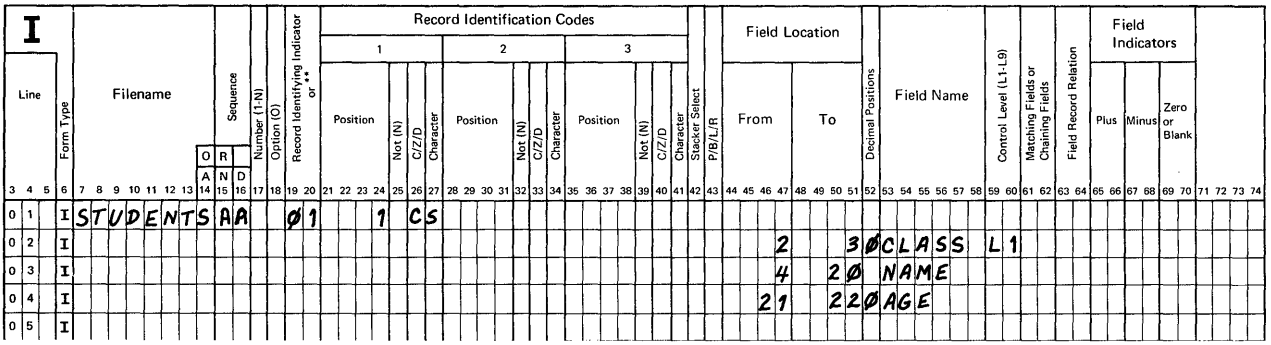
Solution

File Description Specification



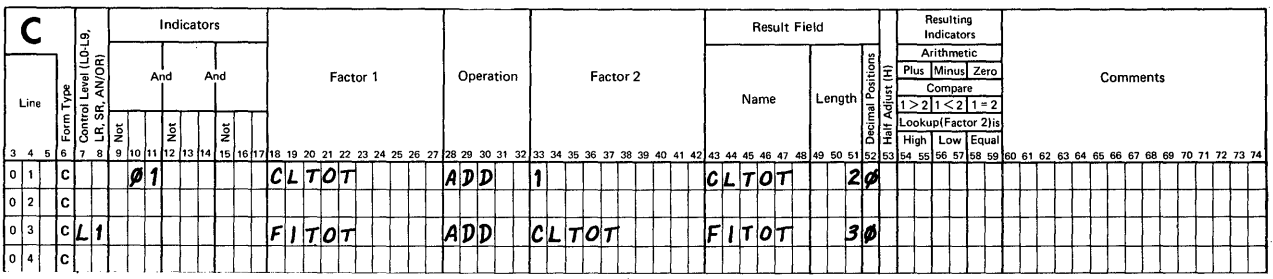
Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	STUDENTSIP						28	26						CONSOLE														
03	F	TALLY	O						132						PRINTER														
04	F																												

RPG INPUT SPECIFICATIONS



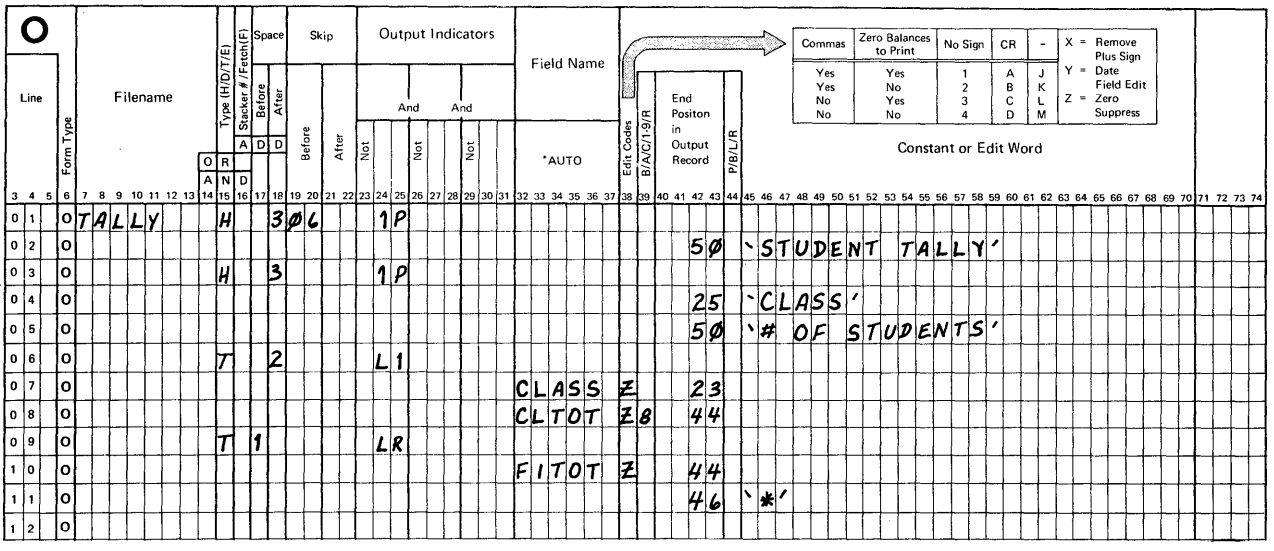
Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	STUDENTS	AA			01		1		CS																				
02	I																				2	30		CLASS	L1					
03	I																				4	20		NAME						
04	I																				21	220		AGE						
05	I																													

RPG CALCULATION SPECIFICATIONS



Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators And Not	Indicators And Not	Indicators And Not	Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions	Half Adjust (H)	Resulting Indicators Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
01	C		01			CLTOT	ADD	1	CLTOT		20											
02	C																					
03	C	L1				FITOT	ADD	CLTOT	FITOT		30											
04	C																					

RPG OUTPUT SPECIFICATIONS



Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	TALLY	H		3	06			1P							
02	O													50		'STUDENT TALLY'
03	O		H		3				1P							
04	O													25		'CLASS'
05	O													50		'# OF STUDENTS'
06	O		T		2				L1							
07	O											CLASS	Z	23		
08	O											CLTOT	Z	44		
09	O		T		1				LR							
10	O											FITOT	Z	44		
11	O													46		'*'
12	O															

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

At this time read the section of your RPG II reference manual that presents information about Control Level Indicators. When you have finished, return to this text.

* * *

Try to answer these questions about control level indicators.

1. Where are control level indicators defined?

2. If indicator L3 is turned on because of a control break, what happens to indicators L1 and L4?

3. Calculations controlled by a control level indicator in positions 7-8 are called what kind of calculations?

* * *

1. Control level indicators are defined on the Input sheet in positions 59-60. After a control level indicator is defined, it may be used for total-time calculations or for total-time output control.

2. When L3 is turned on because of a control break, L1 and L2 are also turned on automatically because they are lesser levels of control. L4 is not affected by a control break on level 3.

3. They are called "total-time" or "total" calculations.

Here is a coding rule you need to remember for making entries on the Calculation sheet.

RULE: When detail-time calculations and total-time calculations are described in a program, all of the detail-time calculations must be placed ahead of the total-time calculations.

RPG II Generated Program Logic

I have added this brief topic because it is related to the topic of control levels. As you read through this section of the text, you should find out "when" the following indicators are turned ON and OFF.

1P	First Page Indicator
Ø1-99	When used as Record Identifying Indicators on the Input sheet
L1-L9	Control Level Indicators
LR	the Last Record Indicator

All programs described in the RPG II language need to be compiled in order to run data processing jobs. When RPG II source programs are compiled, you find that they all have a basic logic that is the same. We will examine this basic logic in three parts:

1. Initialization,
2. Main Program, and
3. Last Record.

Initialization steps are used to start a job. They include the setting of all storage areas used as counters to zero, the turning off of all record identifying indicators, the turning off of all control level indicators, and the turning ON of the first page indicator 1P. If any output is controlled by 1P, it is produced at this time, and then indicator 1P is turned OFF and remains off for the remainder of that job! Note that initialization does not include the reading of any input records!

Here are the steps taken during initialization.

1. Load the desired object program.
2. Set all counters to zero.
3. Turn ON indicator 1P.
4. Turn OFF all other indicators.
5. Produce output controlled by indicator 1P.
6. Turn OFF indicator 1P.

The main program sequence deals with the reading of a record, processing it either singly or as a part of a group, and producing output for either a record or a group of records.

Here are the steps taken during the main program.

1. Read a record.
2. Turn ON its associated Record Identifying Indicator.
3. Test for a control break.
4. If a control break occurs:
    a. Turn ON the associated control level indicator and all lesser levels.
    b. Perform all total-time calculations associated with these indicators.

c. Produce all output controlled by these indicators.

NOTE: For the very first record in a file, bypass steps 4b and 4c.

5. Perform detail-time calculations and produce associated output.

   a. Process the record.

   b. Produce output for that record.

   c. Turn OFF that Record Identifying Indicator for that record type.

6. Turn OFF all control level indicators.

7. Repeat steps 1-6 for all records in the file (or files) being processed.

The last record sequence terminates a job. Each input file includes a special "last record" in addition to all of the data records. When this special record is read during the main program sequence (it is read just as any other record), the normal main program steps are bypassed in order to do the following steps:

1. Turn ON all control level indicators (L1-L9) and turn ON the last record indicator (LR).

2. Perform all total-time calculations controlled by L1-L9 and LR.

3. Produce all total-time output records controlled by L1-L9 and LR.

4. End the job.

Initialization, the main program and the last record section of an object program have been described in general. Your RPG II reference manual includes a very detailed RPG II logic flowchart that you will want to examine after completing this course!

The detailed chart includes quite a few items we have not yet covered in this self-study course, and I suggest that you examine the simplified flowchart included in this text rather than the one in the reference manual.

To summarize, calculations and the producing of output records are controlled by indicators. RPG II controls the settings (both ON and OFF) of:

1. The first page indicator, 1P,

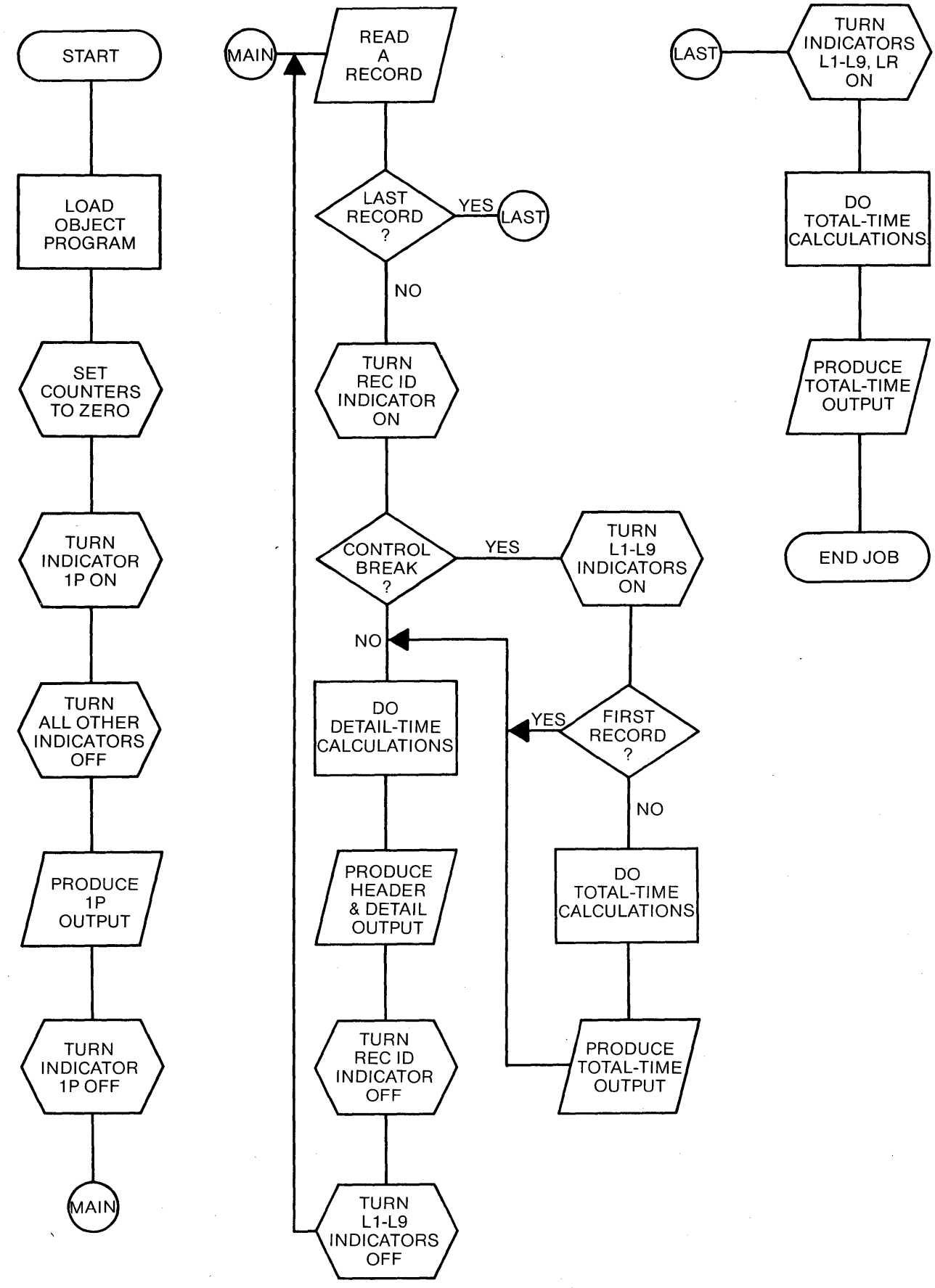
2. Record Identifying Indicators, Ø1-99 as assigned on the Input sheet,

3. Control level indicators, L1-L9 as defined on the Input sheet, and

4. The Last Record Indicator, LR.

The object programs that are generated as a result of compiling RPG II source programs all have a basic logic which is illustrated in the following chart for your reference.

RPG II PROGRAM LOGIC



You have completed the study of this chapter. It provides rules and examples for using the /COPY function of Auto Report, the use of selected operation codes, and the coding of level 1 control breaks. You were directed to read about each of these items in the RPG II reference manual so that you became somewhat familiar with it as it contains all RPG II coding rules and many examples of their uses.

In the next chapter, you will be expected to use any of these items you need in order to code solutions to new types of data processing problems. Do not hesitate to use the reference manual whenever you feel it is needed.

Chapter 4: Summary

This chapter included a variety of topics that should have provided information, rules and examples regarding their application.

At this point you should be able to use the /COPY function to reduce coding effort when sets of specifications you need are already cataloged. Also, you should be able to select and specify arithmetic operation codes, move data codes, and branching codes as needed for processing data. You should be able to describe record grouping for one level of control, making appropriate entries on Input, Calculation and Output sheets as needed. Finally, you should have a grasp of the program logic that exists in each RPG II generated program.

Here is a list of the coding entries you have learned to use in this chapter.

FILE DESCRIPTION

7 - 23	/COPY Fl,name

INPUT

7 - 23	/COPY Fl,name
59 - 60	Control Level (L1)

CALCULATION

7 - 23	/COPY Fl,name
7 - 8	Control Level (L1)
28 - 32	Operation (MULT,DIV,MVR,SQRT,Z-ADD,Z-SUB,MOVE,MOVEL, GOTO,TAG,COMP)
54 - 59	Resulting Indicators (High, Low, Equal - for COMP operation)

OUTPUT

7 - 23	/COPY Fl,name
23 - 31	Output Indicators (L1)
39	Blank After (B)

SELF TEST - Chapter 4

Try to answer these questions from memory. Write your answers on a piece of scratch paper so that you can check them when you've finished.

1. True or False. The /COPY function of Auto Report allows the inclusion of previously cataloged RPG II statements.

2. The Result Field specified in either a subtraction (SUB) or multiplication (MULT) calculation statement must be defined as:

   A. Numeric

   B. Negative

   C. Alphameric

   D. Positive

3. What is the size of the largest result field that can be used in an arithmetic operation such as ADD?

4. What happens to the value in a result field under this condition?

   A new input record is read and the same arithmetic operation statement is performed.

5. Assume that data is being moved using the MOVE operation. What causes data to stop moving?

6. Which field's data is changed during a MOVEL operation?

   A. Factor 1

   B. Factor 2

   C. Result Field

7. How many Resulting Indicators (in 54-59) must be specified for a compare (COMP) operation?

8. True or False. Control level indicators may be assigned to input fields.

9. True or False. Calculations for a control level always occur after its related output is produced.

ANSWERS - Chapter 4 Self Test

1. True

2. A (numeric)

3. 15 positions

4. The old result field value is replaced by the new one.

5. Movement is controlled by the shorter field.

6. Result Field

7. At least 1 must be specified.

8. True

9. False

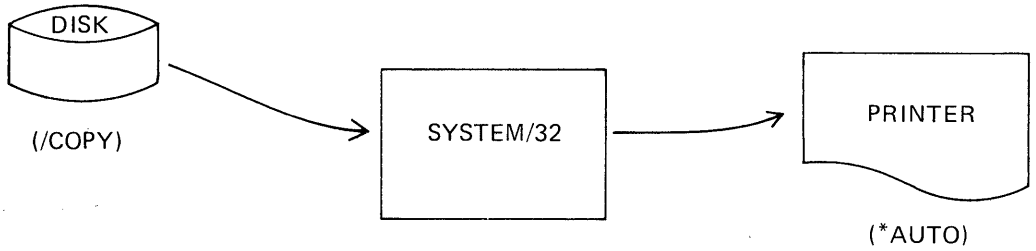
A score of 8 or 9 points is excellent. If you had a score less than 5, you should re-read this chapter before continuing.

```
**********************************************************************
*                                                                    *
*  You have studied enough about RPG II fundamentals by now so       *
*  that you can describe solutions to data processing problems       *
*  in which files are created, records are added to existing         *
*  files, data in records is updated, and data can be processed      *
*  using arithmetic, move, compare and branching operations.         *
*                                                                    *
*  One fundamental of RPG II is that all programs generated from     *
*  coded solutions have the same basic logic.  This logic pro-       *
*  vides for the processing of a single input record, or group       *
*  of related input records, in order to produce a single output     *
*  record.                                                           *
*                                                                    *
*  Another fundamental of RPG II is that each problem solution       *
*  is described on a combination of RPG II specifications sheets.    *
*  Files are described on a File Description sheet; input records    *
*  and fields are described on an Input sheet; processing activ-     *
*  ity is described on a Calculation sheet; while output records,    *
*  fields and constants are described on an Output sheet.            *
*                                                                    *
**********************************************************************
```

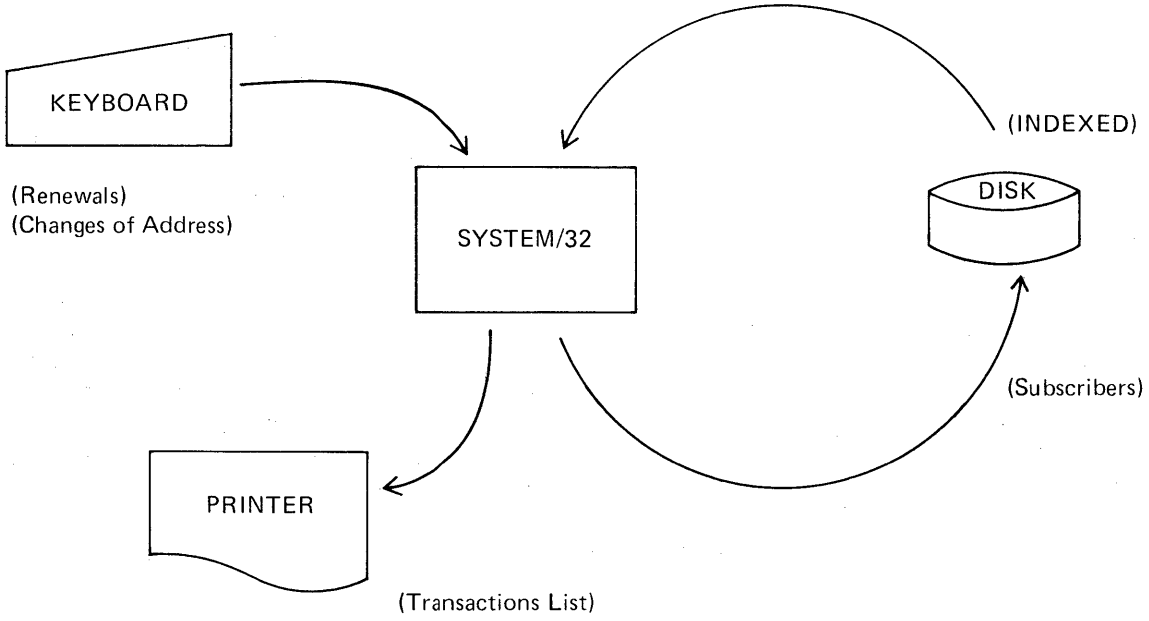
Chapter 5: Practice Problems (3 to 6 hours)

As the name implies, this is a chapter in which practice is provided for you so that you can apply the principles of problem solving, and apply the rules for coding solutions using whatever entries are appropriate to get each job done. A set of four exercises is used in this chapter.

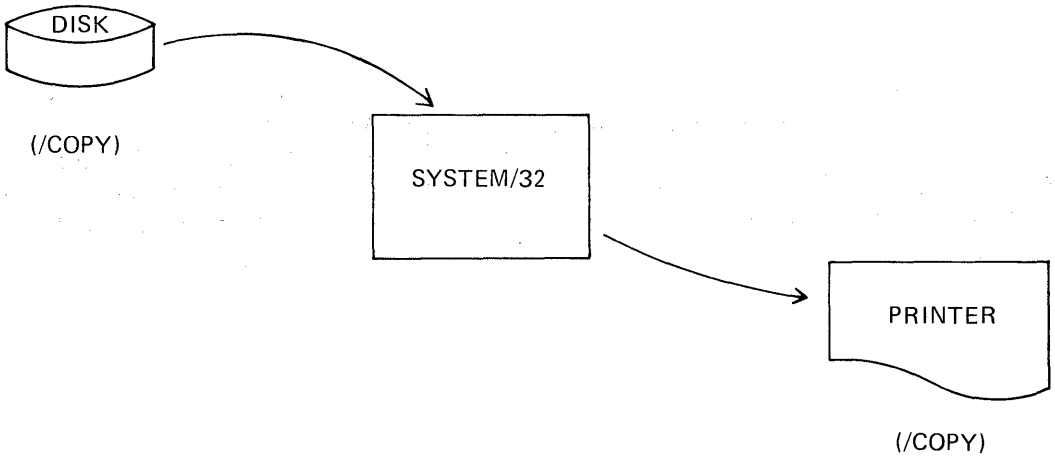
1. Print an "Accounts Receivable Register" from a file of accounts receivable records found on disk. Make use of the two Auto Report functions, /COPY and *AUTO.



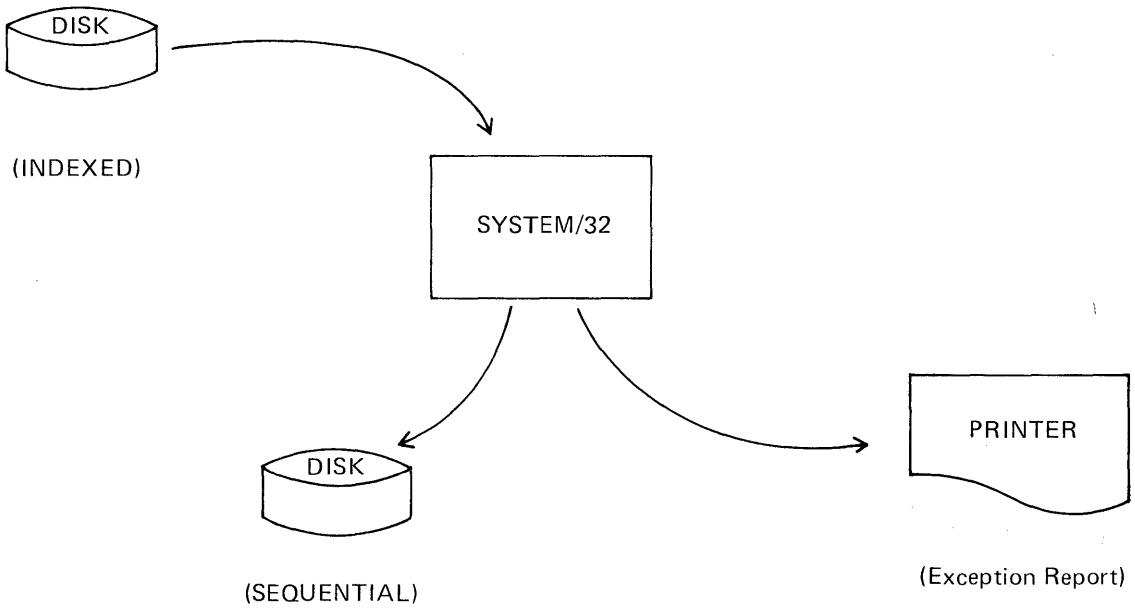
2. Update the records in a "Master Subscriber" file by entering records from the console keyboard that are either renewals or changes of address. The master subscriber file is an indexed file. Include a list of the keyed transactions.



3. Compute monthly electric bills for customers. The File Descriptions, Input and Output specifications for this problem have been cataloged so they will be incorporated as a part of the problem by using the /COPY function of Auto Report.



4. Create a new sequential file of payroll records by copying the payroll records in an existing indexed file. While this is being done, verify each payroll record. If any record includes too many deductions (which would result in either no net pay or worse, negative net pay) print information about that record as a part of an exception report.



No new coding entries are required to code solutions to problems 1, 2 and 3. However, RPG II has a special approach that is useful in producing exception output records. You will learn about this approach as you code the solution to problem 4.

CHAPTER OBJECTIVES

When you have finished this chapter, you should be able to specify solutions to a variety of problems. This will include the description of exception output records, which actually are produced at the time that calculations are being performed.

In addition to this text you will need a number of blank RPG II specification sheets and the System/32 RPG II language manual, SC21-7595. Also, get one or two blank Print Chart sheets, GX20-1816.

There is no self test for chapter 5.

CHAPTER 5: PRACTICE PROBLEM CODING

1. Accounts Receivable Register

    /COPY

    *AUTO

2. Master Subscriber File Update

    CHAIN

3. Computing Electric Bills

    /COPY

    COMP

    ADD

    Z-ADD

    SUB

    MULT

4. Computing Payroll Deductions

    Create a New File

    Print Exception Output

    EXCPT

    UDATE

    PAGE

    Exception Output Records

    Edit Words

You are to code a solution for each of the data processing problems in this chapter. Refer to the RPG II reference manual for specific coding rules you may have forgotten. You will need blank coding sheets of all types in order to complete this set of exercises. You also need a blank Print Chart (GX20-1816) for problem number 4.

I suggest that you concentrate on one problem at a time. Examine all output shown, read and re-read the problem statement until you are sure about what needs to be coded. Then proceed in this manner:

1. Describe the input and output files on a File Description sheet.

2. Describe each type of input record and field on an Input sheet.

3. Describe the desired output on an Output sheet.

   NOTE: If possible, make use of the Auto Report functions, /COPY and *AUTO.

4. Describe calculations not included in *AUTO entries on a Calculation sheet. Remember to place all detail-time calculation entries ahead of all total-time entries on this sheet.

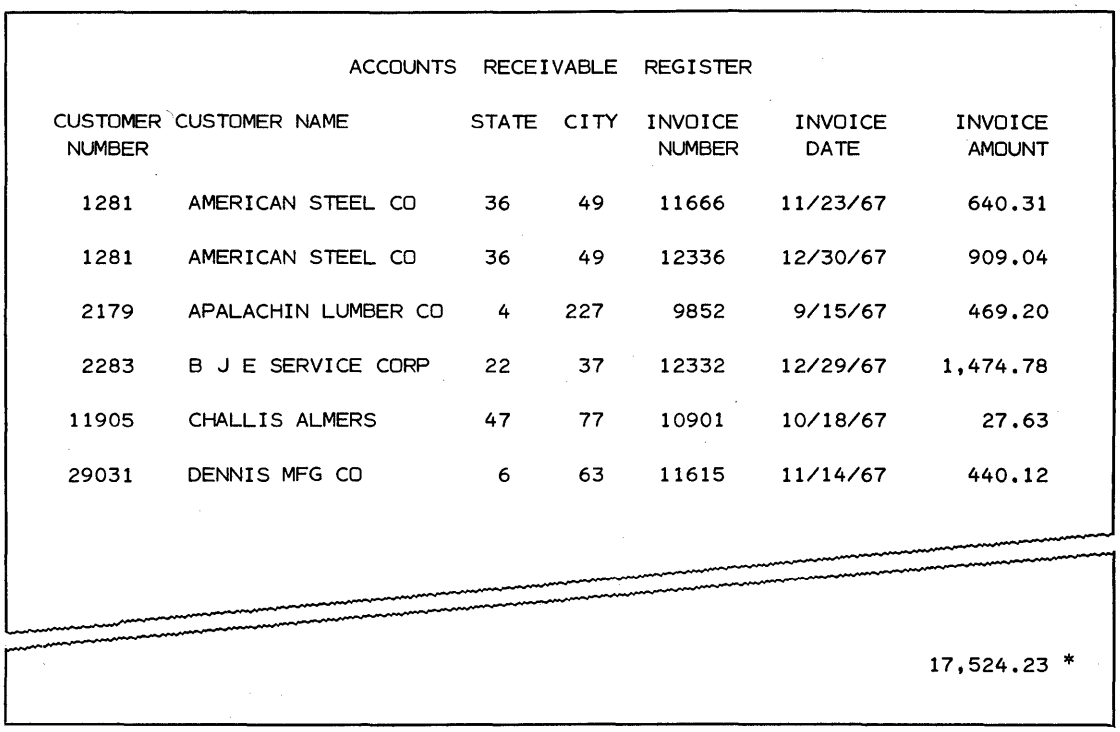
5. Re-read the problem, examine all illustrations and review all of your coding.

When you are satisfied that your coding is accurate, return to this text.

Problem 1: Accounts Receivable Register

A file of 80-character disk records contains information about a certain company's accounts receivable. An "Accounts Receivable Register" is printed for management each month so that it can be determined which invoices have not yet been paid by customers who purchased goods from the company. A sample of that report is on the following page.

Report



ACCOUNTS RECEIVABLE REGISTER

CUSTOMER NUMBER	CUSTOMER NAME	STATE	CITY	INVOICE NUMBER	INVOICE DATE	INVOICE AMOUNT
1281	AMERICAN STEEL CO	36	49	11666	11/23/67	640.31
1281	AMERICAN STEEL CO	36	49	12336	12/30/67	909.04
2179	APALACHIN LUMBER CO	4	227	9852	9/15/67	469.20
2283	B J E SERVICE CORP	22	37	12332	12/29/67	1,474.78
11905	CHALLIS ALMERS	47	77	10901	10/18/67	27.63
29031	DENNIS MFG CO	6	63	11615	11/14/67	440.12

17,524.23 *

The input records contain these fields in the positions shown.

POSITIONS	FIELD
8-29	Customer Name
34-38	Invoice Number
39-43	Customer Number
44-45	State (number code)
46-48	City (number code)
49-54	Invoice Date
74-80	Invoice Amount

This input file has been specified for use in other data processing jobs for the company. The specifications have been cataloged under the name ARIN. The File Description and Input entries shown here make up ARIN.

ARIN

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	Length of Key Field or of Record Address Field	A/P/I/K	Record Address Type	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	ACCTSRECIP							80										DISK													
03	F																															
04	F																															

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	ACCTSRECNS				01																								
02	I																				8	29		CNAME						
03	I																				34	38		INNO						
04	I																				39	43		CUSNO						
05	I																				44	45		STATE						
06	I																				46	48		CITY						
07	I																				49	54	0	INDATE						
08	I																				74	80	2	INAMT						
09	I																													

One small detail. Look at the sample report. The invoice date fields include slashes between month, day and year for reader convenience. Which edit code (column 38 on the Output sheet) will edit this information properly? The correct edit code is the letter Y. Be sure to include it in your solution.

Now, code your solution to this problem. You are to use the /COPY function and the *AUTO function of Auto Report in your specifications.

* * *

Book Solution: Problem 1

The /COPY statement includes one File Description and eight Input statements. In order to complete the coding you need:

1. One File Description statement to describe the output printer file for the report, and
2. A number of Output statements to describe the report. These statements include *AUTO where useful.

Compare your solution to the book solution at this time. Did your solution:

1. Include the name of ARIN in the /COPY statement?

2. Use the exact spellings for output fields, as they are the same as the pre-defined input field names?

3. Specify field names in the same order from left-to-right as information about them was shown on the printed report?

4. Include double spacing of detail lines (code 2 in 18) for "space after"?

5. Include continuation statements on the Output sheet for the column titles (code C in 39)?

6. Include a Y as the edit code (in 38) for the invoice date field?

7. Include an accumulation code (letter A in 39) for the invoice amount field in order to compute a final total?

If your answer to each of the seven questions was "yes", you probably coded the solution correctly. Now take a look at the book solution again. Did you notice that the report title words need spaces between them in order to look like the desired report? Notice that my solution includes 3 blanks between ACCOUNTS and RECEIVABLE and that 3 blanks follow RECEIVABLE before the quote mark is entered at the right.

Suggestion: To spread out the words in a report title when using Auto Report, include a few blanks between them when they are specified as constants.

When you specify continuation statements for column headings, Auto Report attempts to center the shorter constant over (or under) the longer one. If you want to control the positioning yourself, simply include additional spaces in your constant. Was this done for any entries in the book solution?

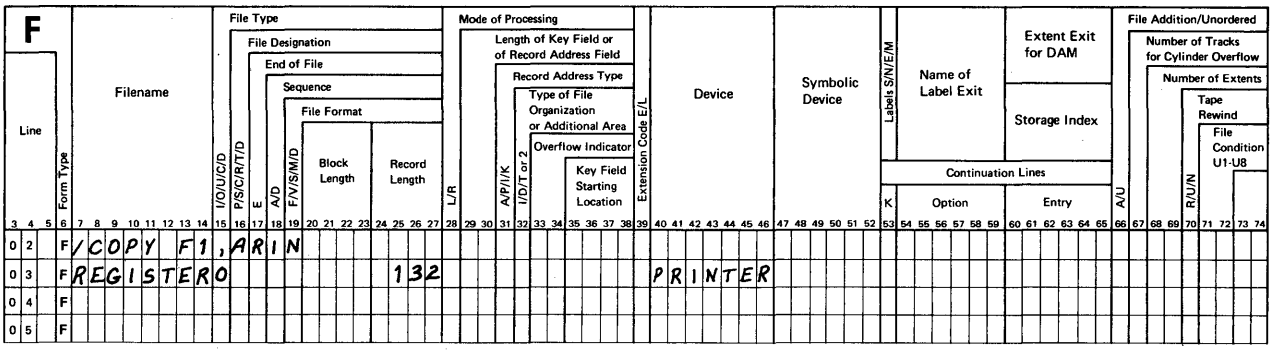
* * *

Yes, the column heading for invoice date included a space. To complete this problem, you should enter a "program identification" at the top right of each page where a set of six blocks is provided. This name must begin with a letter and it may have from one to five additional letters or numbers. Name this problem C5P1 on your sheet.

* * *

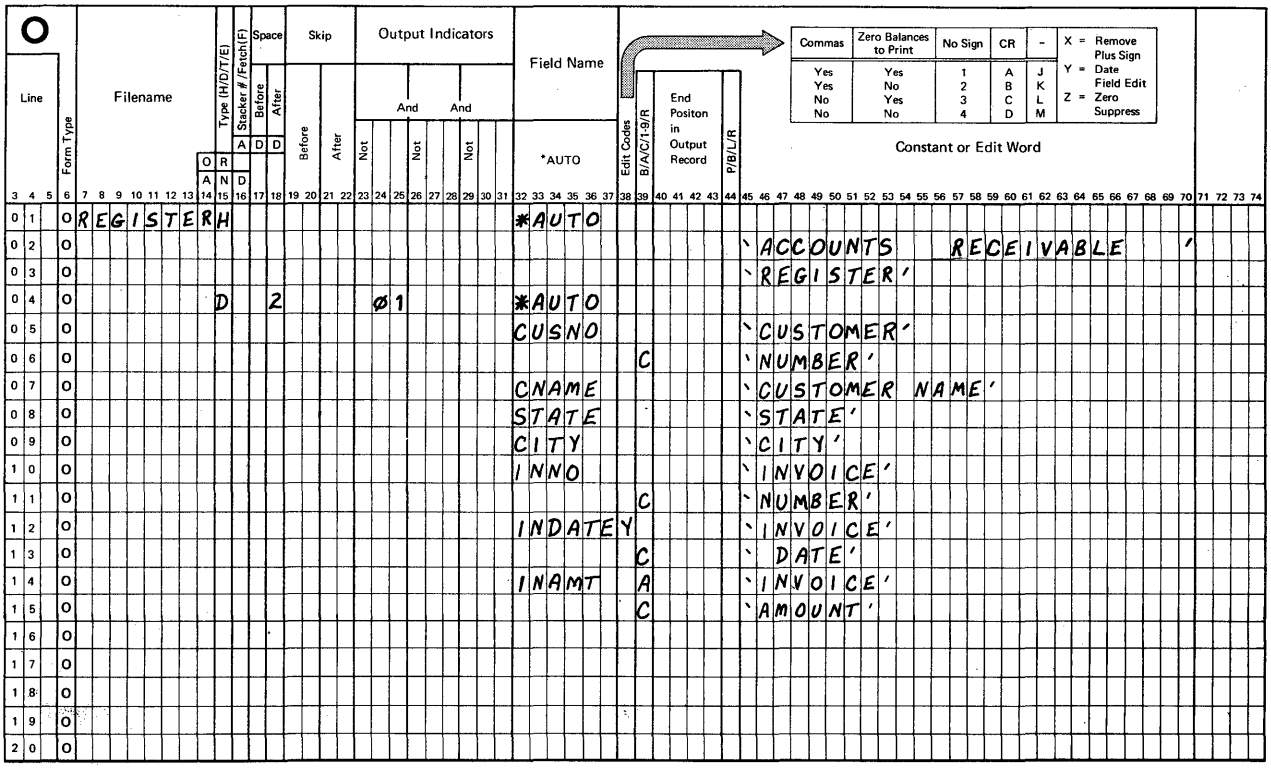
Problem 1: Book Solution

File Description Specification



Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	COPY F1,																											
03	F	REGISTERO							132						PRINTER														
04	F																												
05	F																												

RPG OUTPUT SPECIFICATIONS



Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And/Not	Field Name	Edit Codes	End Positon in Output Record	P/B/L/R	Constant or Edit Word
01	O	REGISTER	H							*AUTO				'ACCOUNTS RECEIVABLE '
02	O													'REGISTER'
04	O		D		2				01	*AUTO				
05	O									CUSNO				'CUSTOMER'
06	O										C			'NUMBER'
07	O									CNAME				'CUSTOMER NAME'
08	O									STATE				'STATE'
09	O									CITY				'CITY'
10	O									INNO				'INVOICE'
11	O										C			'NUMBER'
12	O									INDATEY				'INVOICE'
13	O										C			'DATE'
14	O									INAMT	A			'INVOICE'
15	O										C			'AMOUNT'
16	O													
17	O													
18	O													
19	O													
20	O													

OK. Let's move on to a second problem in which you will be asked to specify the solution to a problem that involves file update.

Problem 2: Master Subscriber File Update

A master file of subscriber records is stored as an indexed disk file. Each record is 85 characters long and includes a key field in positions 1-7. The fields of information in the master file records are:

Positions	Field
1- 7	Subscriber Number (key field)
8-31	Subscriber Name
32-55	Street Address
56-79	City, State and Zip
80-85	Expiration Date of Subscription

The records in this file are updated on a daily basis for 2 kinds of transactions:

1. renewal of subscription, and
2. change of address.

The changes are keyed in from the console as an Interactive Data Entry file and contain these fields:

Type	Positions	Field
Renewal	1	Code - letter R
	2- 8	Subscriber Number
	9-10	Number of Years Renewed
Change of Address	1	Code - letter C
	2- 8	Subscriber Number
	9-32	New Street Address
	33-56	New City, State and Zip

You are to code a solution to this problem so that these requirements are met:

1. The master file is to be updated randomly using "chaining".

2. For a "renewal", add the number of years in the renewal record to the "year" position of the expiration date field in the master record.

3. For a "change of address", replace the street address and the city, state and zip code fields in the master record.

4. Include a simple Auto Report list of the keyed in transactions. Here's a sample of that listing:

CODE	SUBSCRIBER	YEARS	STREET	CITY/STATE
---	--------	--		
---	--------		------------------	----------
---	--------	--		

5. If no master record is found to match either a renewal or change of address record, print the message, NO MASTER FOUND.

6. For printing, specify the edit code Z (zero suppress only) for subscriber number and for number of years to be renewed.

7. Include a "program identification" entry at the top of each page. You may use any name provided it starts with a letter. It may be up to six characters long.

When you are ready to code a solution, you may refer to the RPG II reference manual or to this text. When you have finished coding and you think it is a correct solution, return to this text.

* * *

Before you examine the book solution, examine your own entries. Be sure you have:

1. Re-read the entire problem, and

2. Re-read the list of 7 requirements that are to be met for a satisfactory problem solution.

* * *

Problem 2: Book Solution File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8	R/U/N
02	F	SUBMAS	U	C					85	R	7	A	1	1	DISK													
03	F	CHANGES	I	P				62	60						CONSOLE													
04	F	LIST	O						132						PRINTER													

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence (OR / AND)	Number (1-N) Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	CHANGES	AA		01		1		CR																				
02	I																			1	1		CODE						
03	I																			2	80		SUBSNO						
04	I																			9	100		YEARS						
05	I		BB		02		1		CC																				
06	I																			1	1		CODE						
07	I																			2	80		SUBSNO						
08	I																			9	32		NEWST						
09	I																			33	56		NEWCTY						
10	I	SUBMAS	CC		03																								
11	I																			84	85		EXDATE						

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
01	C		01		SUBSNO	CHAINSUBMAS												11			
02	C		01N11		EXDATE	ADD	YEARS	EXDATE													
03	C		02		SUBSNO	CHAINSUBMAS												22			

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # (Fetch)	Space Before	Space After	Skip Before	Skip After	Output Indicators: And Not	And Not	And Not	Field Name / *AUTO	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	SUBMAS	D						01N11							
02	O											EXDATE		85		
03	O		D						02N22							
04	O											NEWST		55		
05	O											NEWCTY		79		
06	O	LIST	D	2					01			*AUTO				
07	O		OR						02							
08	O											CODE				'CODE'
09	O											SUBSNOE				'SUBSCRIBER'
10	O								01			YEARS Z				'YEARS'
11	O								01	11						'NO MASTER FOUND'
12	O								02			NEWST				'STREET'
13	O								02			NEWCTY				'CITY / STATE'
14	O								02	22						'NO MASTER FOUND'

Commas | Zero Balances to Print | No Sign | CR | -
Yes | Yes | 1 | A | J
Yes | No | 2 | B | K
No | Yes | 3 | C | L
No | No | 4 | D | M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

NOTE: When a detail line is to be printed using an OR condition with the *AUTO function, enter *AUTO for the first detail record specification. Do not use it on an "OR" line.

Problem 3: Computing Electric Bills

In the next problem, you are to concentrate on the calculations needed to solve a problem in which:

1. All files are already described and cataloged for use under the name ALLFILES.

2. All input records and fields have been described and cataloged for use under the name INRECS.

3. All output records and fields have been described and cataloged for use under the name OUTRECS.

The calculations you are to describe shall be those used to compute a customer's electric usage bill. This particular electric company has the following rates:

Usage	Rate
1- 50 kwh	$.05 each
51-100 kwh	$.04 each over the first 50, + $2.50
101-300 kwh	$.03 each over the first 100, + $4.50
over 300 kwh	$.025 each over the first 300, + $10.50

Here are some examples:

Customer	Usage	Monthly Bill
123	30 kwh	?
776	70 kwh	?
406	120 kwh	?
882	310 kwh	?
396	50 kwh	?
451	100 kwh	?
275	300 kwh	?
666	0 kwh	?

The monthly bill for customer #123 comes to $1.50 for the 30 kwh of electricity used. From the table, the rate is $.05 each.

The monthly bill for customer #776 comes to $3.30 for the 70 kwh used. From the table, the rate is $.04 each over 50 + $2.50.

Use a piece of scratch paper to determine the bill for each customer. Round all answers to the nearest whole cent.

* * *

Check your answers against these. If you disagree, re-calculate the bill or bills in question before you continue.

Customer	Usage (kwh)	Calculation	Bill
123	30	30 X .05	$ 1.50
776	70	20 X .04 + 2.50	$ 3.30
406	120	20 X .03 + 4.50	$ 5.10
882	310	10 X .025 + 10.50	$10.75
396	50	50 X .05	$ 2.50
451	100	50 X .04 + 2.50	$ 4.50
275	300	200 X .03 + 4.50	$10.50
666	0	0 X .05	$ 0.00

Naturally, your billing calculations depend upon the meter readings taken last month and this month because the number of kwh used equals the reading this month less the reading last month. Normally this is no particular problem to compute, but every so often, the meter "turns over", that is, the number shown passes the all 9's reading. When this happens, you need to imagine that a 1 carried into the next place when you compute usage. Keep in mind that computers don't "imagine" things like that, they need to be programmed to include the necessary steps to actually perform the calculation.

Here are two examples:

This Reading	Last Reading	Usage
02830	02750	80
00150	99900	250

For the first example, we simply subtract the last reading (02750) from this reading (02830) to compute usage.

For the second example, you really need to tell the computer to subtract 99900 (last reading) from 100150 (this reading plus a 1 carry) in order to get the right usage. We'll talk more about how to describe this calculation later on.

You will refer to the following list of steps we need to code when you begin to describe the calculations:

1. Compute the usage for the month.

   a. Subtract last month's reading from this month's reading.

   b. If the answer is a negative value, we know that the meter went past all 9's at some time during the month.

   c. To correct this answer, increase this month's reading by 100000 and then subtract last month's reading from this larger value.

2. If the usage is up to and including 50 kwh, multiply the answer from step 1 by $.05 to compute the bill.

3. If the usage is from 51 to 100 kwh inclusive, find out how many kwh over 50, multiply them by $.04, and then add $2.50 to compute the bill.

4. If the usage is from 101 to 300 kwh inclusive, find out how many kwh over 100, multiply them by $.03, and then add $4.50 to compute the bill.

5. If the usage is over 300 kwh, find out by how many, multiply them by $.025 (for 2.5¢) and round the answer to the nearest cent, and then add $10.50 to compute the bill.

You probably noticed that 4 out of the 5 steps started with "if"! When it's time to describe these statements, which operation code should you consider using? Well, when a sentence includes "if", there are two choices, and that indicates the need for a comparison. The operation code I'd keep in mind is compare, COMP.

Before you can code the solution to this problem, you need to know:

1. The field that will print the amount of the bill is named BILL in the output specifications. It is to be 5 positions long with 2 decimal positions.

2. The field that will print the usage during the month is called KWH. It is to be 5 positions long with no decimals.

3. The numeric input fields needed are defined as follows:

Field	Size	Purpose
NEW	5, no decimals	holds this month's meter reading
LAST	5, no decimals	holds last month's meter reading

Now, take a blank Calculation sheet and specify the statement needed to compute the usage for the month. Call the result USE with 5 positions, no decimals.

* * *

Coding that first statement is straightforward. If you look back (or can remember) you'll see that it is possible for the result to be negative in value if the meter turned past all 9's during the month. Look at columns 54-59 on your Calculation sheet. Note that "Resulting Indicators" may be specified for either Arithmetic, Compare or Lookup (table lookup) operations. At the moment we are concerned about making an entry following subtraction, an arithmetic operation. In which columns should you specify an indicator to determine whether or not the result of a subtraction is negative?

The correct answer is columns 56-57 which you see is marked "Minus". You may enter any indicator from 01 to 99 that has not yet been assigned in your program. For convenience of the author, specify that indicator 20 shall be used to test for the negative condition. Specify this indicator on the same line as your subtraction operation at this point.

* * *

Remember the two examples of meter readings we used earlier? If last month's reading was 02750 and this month's is 02830, will indicator 20 be ON or OFF after the operation has been completed? If last month's reading was 99900 and this month's is 00150, will indicator 20 be ON or OFF after the operation?

In the first example, indicator 20 stays OFF, but in the second, it turns ON. When it stays off, we can use the result as is. When it is turned on, we've got to do some more describing.

1. We must change this month's reading value by adding 100000 to it.

2. Then we must subtract last month's meter reading from the new value.

These two steps require the use of a new, larger size field. Since the field named NEW has only 5 positions, we cannot add 100000 to it and store the answer in a 5-position field.

Instead, add 100000 to NEW and store the answer in a 6-position field named BIG (just so you and I use the same names). After that, subtract last month's meter reading from this new field (BIG) in order to compute this month's usage and store it in the 5-position field named USE.

RULE: The result field must be large enough to hold the answer to an arithmetic operation, or the result will not be accurate.

Remember, those two operation steps are to be done only when indicator 20 is ON. Now, code two statements that satisfy our requirements.

* * *

Just to make sure you're on the right track, here's what your coding should be like.

RPG CALCULATION SPECIFICATIONS

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators (9-11)	And (12-14)	And (15-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / High / 1>2 (54-55)	Minus / Low / 1<2 (56-57)	Zero / Equal / 1=2 (58-59)	Comments (60-74)
01	C					NEW	SUB	LAST	USE	5	Ø			2Ø		
02	C		2Ø			NEW	ADD	1ØØØØØ	BIG	6	Ø					
03	C		2Ø			BIG	SUB	LAST	USE							
04	C															
05	C															

Before we forget, the field named BIG is only a convenience field. That is, it is used to hold an intermediate value needed to complete the arithmetic operation we really want to perform. In this case, when indicator 20 is ON and we re-compute a value for USE, the field called BIG is of no further use. <u>But don't forget what is stored in it!</u> If we get another set of readings in which the meter has turned past 9's, we must be sure that BIG is reset before we use it again (or by this time you will be adding 100000 for a second time).

To take care of this, we must code a statement that will reset BIG to zero so we can use it properly that next time it's needed. How would you code a reset for this field?

You have a number of choices. Here's my choice.

```
C     2Ø                    Z-ADD  Ø          BIG
```

From your reading of this operation, zero and add, you remember that its purpose is to set the result field named to zero and then add what is in Factor 2 to it. Since we specified Ø as Factor 2, this operation will reset BIG properly.

Here's another way of doing the same thing.

```
C     2Ø    BIG          SUB   BIG          BIG
```

Here we tell the computer to subtract what is in the field named BIG (Factor 2) from the field named BIG (Factor 1) and store that answer in the field named BIG (the Result Field).

What will happen in this case?

```
C     2Ø     6           MULT  Ø            BIG
```

BIG will have a value of zero after the operation because the answer when any number is multiplied by zero is always zero. My choice is still the Z-ADD operation.

At this time specify a statement that will set BIG to zero in your program. Why should you include the indicator 20 in this entry?

* * *

We are now ready to use the value found in the field, USE, in order to complete the calculation of the customer's bill. However, we need to store the usage figure for printout as a field named KWH with 5 positions, no decimals. What operation can be used for this purpose? Again the zero and add (Z-ADD) operation is useful. If you thought of using either MOVE or MOVEL (move operations) you are also correct, since both USE and KWH are 5-position numeric fields with no decimal places. Make an entry to store the value in USE in the field called KWH at this time.

* * *

Look back to the 5-step list if you want to refresh your thinking about what needs to be done in order to complete the billing calculations.

Next, we will specify entries that take care of this requirement.

> "If the usage is up to and including 50 kwh, multiply by $.05."

The question you need to ask yourself is, "How do I determine if the usage is 'up to and including 50 kwh'?" We know that the usage is stored in the field named USE. What operation can be specified to determine how large that value is?

You can compare the value in USE against 50, or you can subtract the value in USE from 50 to see if the result is negative. For this example, use the compare operation and assign indicator 30 for the condition you think should be tested in columns 54-59.

The column heading over 54-55 is 1>2 and means, "if the value in Factor 1 is greater than the value in Factor 2, turn on the indicator that is specified in 54-55." Over 56-57, the title 1<2 means just the opposite, "if Factor 1 is less than Factor 2, turn on the specified indicator." Over 58-59, the column heading is simply for 1 = 2. Specify the comparison you want now.

* * *

RULE: When COMP is specified as the operation, you may enter an indicator in 54-55, 56-57, or 58-59. You may enter the same indicator in any two sets of these columns, but not all three.

Assume that the comparison proves that USE has no more than 50 as a value. Code a statement below yours to compute the value for the field called BILL under these conditions. BILL has 5 positions with 2 decimals and the rate for this amount of usage is $.05.

* * *

Here's the way I specified these 2 entries.

RPG CALCULATION SPECIFICATIONS

C				Indicators							Factor 1	Operation	Factor 2	Result Field			Resulting Indicators						Comments
Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)		And		And								Name	Length	Decimal Positions	Half Adjust (H)	Arithmetic: Plus	Arithmetic: Minus	Arithmetic: Zero	Compare: 1>2	Compare: 1<2	Compare: 1=2	
				Not	Not	Not	Not											Lookup(Factor 2)is: High	Lookup(Factor 2)is: Low	Lookup(Factor 2)is: Equal				
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				
01	C								USE	COMP	50						30						
02	C		N30						USE	MULT	.05	BILL		52									
03	C																						
04	C																						

NOTE: When you specify a numeric literal as a factor, you may include a decimal point if needed for the computation.

In the first coded statement, I tested the condition, Factor 1 greater than Factor 2, by entering indicator 30 in columns 54-55.

Here's another way of coding a correct solution.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup (Factor 2) is (High / Low / Equal)	Comments
01	C			USE	COMP	50					3030	
02	C		30	USE	MULT	.05	BILL		52			
03	C											

In this first statement, I test for 2 conditions, Factor 1 less than Factor 2 or Factor 1 equals Factor 2.

Your solution may be done either way. Check yours to see if you included all of the entries you needed.

Consideration: Coding solutions to data processing problems is a personal matter. The important point is that your coded description must meet all of the problem requirements. In short, "You can code anything you want, provided it's right."

We have completed the steps needed to calculate a bill for any customer who uses from 1-50 kwh during a month. But, we have more calculation steps to specify. Therefore we need to be able to "bypass" all statements that follow these. What operation is used to bypass others? How is it coded? What entry should we make in order to finish the coding of this portion of the problem?

The operation for bypassing statements is GOTO. To code it you enter GOTO in 28-31 and "invent" a name for the label entry as Factor 2. To satisfy our requirement, I used this statement which includes an indicator entry.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup (Factor 2) is (High / Low / Equal)	Comments
01	C											
02	C		N30		GOTO	END						
03	C											

As you recall, we'll need to code a TAG statement at the place where all other detail-time calculations have been bypassed. We will fill in that statement later on. Specify a GOTO statement on your sheet and include the use of indicator 30.

* * *

Read step 3 in the list of requirements again. How many different calculations are needed to satisfy this one?

* * *

I counted 3 of them. First, find out how many more than 50 kwh were used; second, multiply that number by $.04; third, add $2.50 to compute the bill. When you specify these 3 calculations, you may find that you need even more. Use the compare operation and compute the bill. If an indicator is needed, assign 40. Use a 5-position field with no decimals named OVER to figure out how many kwh over 50 there are, and bypass all other operations as your final statement. Specify your entries for this requirement just below the ones already coded.

* * *

Here are the entries I used to satisfy requirement #3. Your entries should be like these. Did you remember to add $2.50 to the bill? Did you include a GOTO statement as the last?

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (1>2)	Low (1<2)	Equal (1=2)	Comments
01	C			USE	COMP	1ØØ					4Ø			
02	C		N4Ø	USE	SUB	5Ø	OVER	5	Ø					
03	C		N4Ø	OVER	MULT	.Ø4	BILL							
04	C		N4Ø	BILL	ADD	2.5Ø	BILL							
05	C		N4Ø		GOTO	END								
06	C													
07	C													

This is a good time to state some rules about calculations in RPG II.

RULES: The COMP operation requires entries for Factors 1 and 2, and at least one Resulting Indicator assignment.

When a Result Field is used for the first time, it must be defined by including a length. If this field is to be numeric, it also requires an entry for Decimal Positions. Thereafter, columns 49-52 are not used for this same field in this program.

If a field is defined on the Input sheet and used as a Result Field on the Calculation sheet, no entries are used in columns 49-52 for it.

If a numeric literal is used as either Factor 1 or Factor 2, you may include a decimal point where needed. You may also include either a + or - sign at the far left of the number if you need it.

To add a value to a field, specify that field as one of the factors, and as the Result Field.

Calculations will be performed in the program in the same order as specified, from top to bottom of the sheet. If any indicators have been specified in columns 9-17, the operation will be done only when the condition(s) described exist.

Now, back to our problem. Re-read requirement #4. When you are ready, make all of the entries you feel are needed to satisfy this requirement. Assign indicator 50 if you need to use an indicator. You may need an extra blank Calculation sheet to complete all 5 requirements in this job.

* * *

Requirement #5 includes a rate of 2.5¢ ($.025) for all kwh over 300. It is possible that the calculation for such a bill will come out with one-half cent as a part of the answer. When you code your solution for requirement 5, include an entry to round off this answer (half adjust it). If you need an indicator, use 60. Code your entries for requirement #5.

* * *

Since this step is the last of the calculations, by that I mean that there are no rates beyond the one for over 300 kwh, we do not need a COMP operation. Also, we don't need an indicator. All you needed to do was:

1. Figure out how many kwh over 300,

2. Multiply them by $.025 and round the answer (column 53, code H), and

3. Add $10.50 to this value for the total bill.

Here's another point. In all prior coding steps, we had to include a statement to bypass statements that followed. Once again, since this is the last requirement we do not need a GOTO statement. Now, how does your coding look?

We need one more entry to complete the job. In every GOTO statement I used the label END as Factor 2. Now I've got to include a TAG statement with that same label as Factor 1. By placing this entry last, any GOTO in the series will bypass everything else.

Here's the way I coded requirements 4 and 5 in the list. When you are satisfied that you understand what we've been doing, take a few minutes to review the entire set of 5 requirements and all of <u>your</u> calculations. If you omitted any entries, make corrections on your sheets before we continue.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (LO-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus / Minus / Zero; Compare 1>2 / 1<2 / 1=2; Lookup(Factor 2)is High / Low / Equal	Comments
01	C			USE	COMP	300					50	
02	C		N50	USE	SUB	100	OVER					
03	C		N50	OVER	MULT	.03	BILL					
04	C		N50	BILL	ADD	4.50	BILL					
05	C		N50		GOTO	END						
06	C											
07	C			USE	SUB	300	OVER					
08	C			OVER	MULT	.025	BILL			H		
09	C			BILL	ADD	10.50	BILL					
10	C											
11	C			END	TAG							
12	C											
13	C											

We concentrated on the calculation entries in this third problem, but the total job is not completed. Refer to the page where problem 3 starts. There are 3 additional /COPY statements needed to complete this coding. Using a blank File Description sheet, a blank Input sheet and a blank Output sheet, code the copy function statements needed for the job. When you have done that, enter the name C5P3 as the "program identification" at the top right of every sheet including your Calculation sheets.

If you don't remember how to specify a /COPY entry, use your RPG II manual for reference.

* * *

One more thing. Most data processing jobs require the use of multiple sheets of entries of different types. In order to keep them in the proper order for later reference during compilation, there are 2 blocks on the top of each page near the program identification blocks where you should enter a page number. When you are ready to number pages, place the sheets in this order.

1. File Description
2. Input
3. Calculation
4. Output

Now organize and page number your sheets for problem #3.

* * *

Problem 3: Book Solution, Page 1

File Description Specification

Page 01 of 4 — Program Identification C5P3

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	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	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
			I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D			L/R		A/P/I/K	I/D/T or 2											A/U			R/U/N	
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
02	F	/COPY F1,ALLFILES																										
03	F																											
04	F																											
05	F																											
06	F																											
07	F																											
08	F																											
09	F																											
10	F																											
	F																											
	F																											

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic								Card Electro Number
Programmer	Date		Punch								

Page 02 of 4 — Program Identification C5P3

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
			O A 14	R N 15	D 16		1				2				3															
3 4 5	6	7 8 9 10 11 12 13				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
01	I	/COPY F1,INRECS																												
02	I																													
03	I																													
04	I																													
05	I																													

Problem 3: Book Solution, Page 2

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 03 of 4 — Program Identification C5P3

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup(Factor 2)is (High / Low / Equal)	Comments
01	C			NEW	SUB	LAST	USE		50		20	
02	C		20	NEW	ADD	100000	BIG		60			
03	C		20	BIG	SUB	LAST	USE					
04	C		20		Z-ADD	0	BIG					
05	C				Z-ADD	USE	KWH		50			
06	C			USE	COMP	50					30	
07	C		N30	USE	MULT	.05	BILL		52			
08	C		N30		GOTO	END						
09	C			USE	COMP	100					40	
10	C		N40	USE	SUB	50	OVER		50			
11	C		N40	OVER	MULT	.04	BILL					
12	C		N40	BILL	ADD	2.50	BILL					
13	C		N40		GOTO	END						
14	C			USE	COMP	300					50	
15	C		N50	USE	SUB	100	OVER					
16	C		N50	OVER	MULT	.03	BILL					
17	C		N50	BILL	ADD	4.50	BILL					
18	C		N50		GOTO	END						
19	C			USE	SUB	300	OVER					
20	C			OVER	MULT	.025	BILL			H		
21	C			BILL	ADD	10.50	BILL					
22	C			END	TAG							
	C											

RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 04 of 4 — Program Identification C5P3

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And / Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	/COPY F1,OUTRECS													
02	O														
03	O														
04	O														

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

The final problem to be coded in this chapter requires the use of two new operation codes, exception record output, special named fields (UDATE and PAGE), and the use of edit words.

All coding for this problem is to be done with standard RPG II entries, that is, we will not use any Auto Report functions. You have already learned how to make entries on each specification sheet required to solve this problem, but you need to find out about the new items mentioned in the first paragraph.

This is the problem to be described. Don't start to code until you have read all of the information about the new items or are directed to do certain coding.

Problem 4: Computing Payroll Deductions

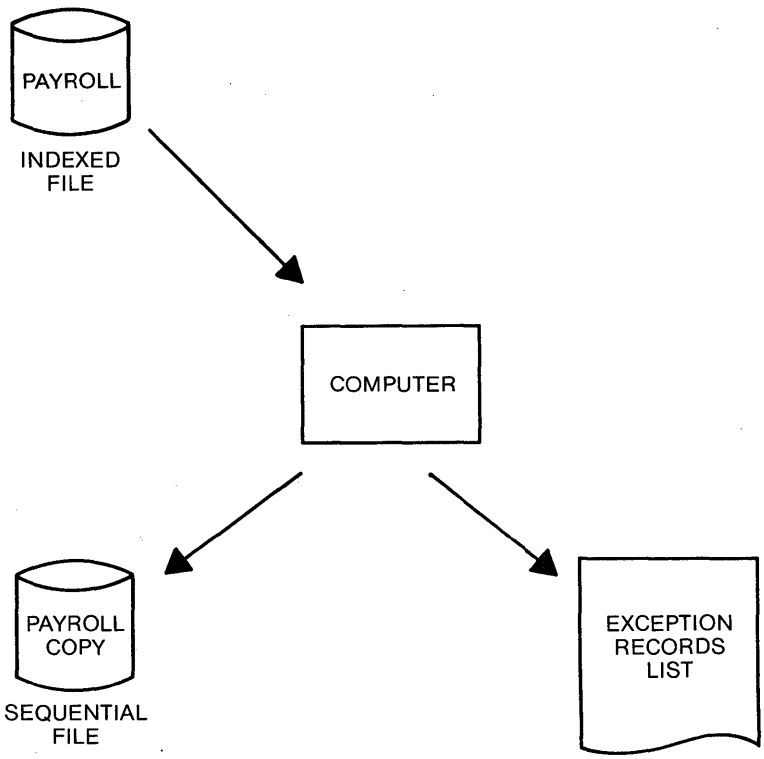
An indexed disk file of 100-character records is to be copied for reference. Its key field is in positions 1-8. The new file will contain identical records, but is to be organized sequentially.

As each input record is read, a computation for net pay is to be performed by following this formula:

Net Pay = Gross Pay - Federal Income Tax - Social Security Payment - Deduction 1 - Deduction 2 - Deduction 3 - Deduction 4

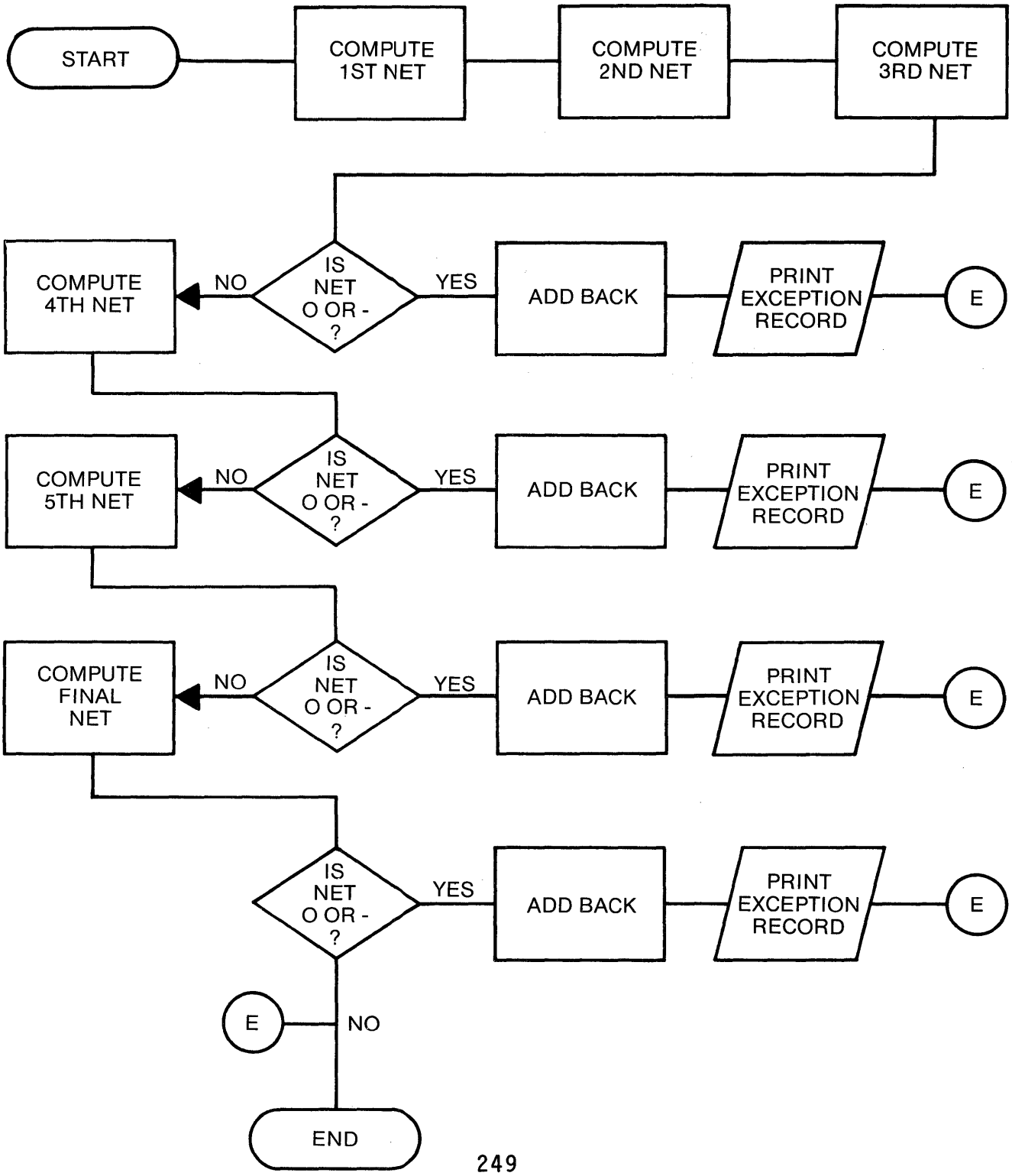
The last four items (deduction payments 1-4) have been authorized by each employee and are to be taken out each pay period, unless by taking it out the net pay value becomes either zero or negative. If that should happen, you are directed to add back the deduction amount just subtracted, print an exception record right at that point in the calculations, and bypass all further net pay calculations for that record.

Here's a general flowchart of the problem to be solved.



The flowchart on this page is also for your reference. In order to compute the net pay value for a record, remember this series of steps:

1. First Net Pay = Gross Pay - Federal Income Tax
2. 2nd Net Pay = First Net - FICA (social security payment)
3. 3rd Net Pay = 2nd Net - Deduction 1
4. 4th Net Pay = 3rd Net - Deduction 2
5. 5th Net Pay = 4th Net - Deduction 3
6. Final Net Pay = 5th Net - Deduction 4



You will need blank coding forms of each type. You may need 2 or 3 Calculation sheets and 2 or 3 Output sheets in addition to the File Description and Input sheets.

Code this problem solution in a couple of phases rather than all at once. First, let's concentrate on making a copy of the indexed file records. The output file is to contain the same information, but shall be sequentially organized rather than indexed. If you think you have enough information, code all of the entries needed to copy this file. If you want more information, read on.

* * *

What information do you think you need? Re-read the problem. How big are the indexed file records? Where is the key field? How big are the new records supposed to be on the sequential file? How many input fields and output fields must you describe in order to solve the problem? What must be coded?

* * *

The indexed file records are 100 characters long. The key field is 8 positions in length from positions 1-8. The sequential file shall also contain 100-character records. Only 1 input field of 100 positions and 1 output field need to be described. Now try to code your solution for this phase of the problem.

* * *

You should have described two disk files on a File Description sheet. For the indexed file, include entries in 29-30, 31, 32, and 35-38. No entries are needed in these positions for sequential files at any time.

You should also have coded two input statements. One to describe the record and another to describe the field. Since a field may contain as many as 256 characters, only one field is described for the 100-character records.

No calculation entries are needed for this phase of the problem.

You should have made two entries on the Output sheet. One to describe the output record type and another to describe the 100-character field it is to contain. Did you enter the same indicator and the same field name on both the Input and Output sheets? If not, change your coding to make it correct before continuing.

* * *

The first phase is complete. No new coding entries were required in order to satisfy these requirements. Next, we need to describe all entries needed to calculate net pay and the printing of exception records.

Let's think about the calculations. In order to calculate the final net pay amount, we need to describe a whole series of steps.

RULE: When a series of steps is needed in order to calculate a result, describe each step on one coding line in the sequence needed to solve the problem.

The first step in this series is straightforward. Simply subtract the amount of Federal income tax from gross pay. The result of this subtraction is to be stored, as it is the First Net Pay amount and will be used for the next step.

Wait a minute. The fields of data needed to compute net pay are not specified as yet. Use your input sheet and make these additional entries. You are to make up field names for them.

Positions	Decimals	Field
1- 3	0	Department Number
4- 8	0	Man Number
9-17	0	Social Security Number
18-22	2	Gross Pay
23-27	2	Federal Income Tax
28-32	2	Social Security Tax Amount
53-56	2	First Deduction Amount
57-60	2	Second Deduction Amount
61-64	2	Third Deduction Amount
65-68	2	Fourth Deduction Amount
*	*	*

The calculation series for computing net pay on the flowchart and the list of steps mentions the calculations for: 1st Net, 2nd Net, 3rd Net, 4th Net, 5th Net and Final Net Pay. When coding, keep in mind that only 1 field is used to store the net pay. That is, each time you compute a value for net pay, it is stored in the same field.

Earlier we were ready to start coding a solution. Let's get to it now. The first step, as I said, is straightforward. Simply subtract the amount of Federal income tax from gross pay to calculate the first net pay amount. Refer to your Input sheet for the names you assigned to the desired input fields and then describe this entry. Make up a name for your net pay field. It shall be 5 positions long and have 2 decimals.

RULE: When there is only one input record type, and calculations are to be performed for every record, no indicator entry is needed in positions 9-17 on the Calculation sheet.

Refer to the list of calculation steps in this text. What should be specified for the next step? What do you code as Factor 1? as Factor 2? as the Result Field?

* * *

To compute the second net pay value, subtract the social security tax amount (use the field name you assigned) from the first net pay value. Factor 1 should contain the same name as the Result Field you specified in step 1. Factor 2 is the social security payment field. The Result Field is again the same as the one you used for step 1. Specify this new step now.

* * *

Did you include a field length and a number of decimals for this second step in positions 49-52? You should not include them because you already described them on the first step. Remember the rule:

> "Define a result field by name and length the first time you specify it. Thereafter, simply use the same name."

Refer to your Input specifications and the problem flowchart for all of the remaining calculation steps.

To compute 3rd net, subtract the first deduction amount from the 2nd net amount. After that, you are to determine whether or not the 3rd net amount (the new result) has become zero in value or has turned negative. If the 3rd net is either zero or negative, we are to "add back" the deduction amount that caused it to happen, and then produce an exception output record to identify the employee for whom not all authorized payroll deductions could be taken.

Before coding any of the entries to satisfy these requirements, spend some time reading about the topics below in your RPG II reference manual.

Readings

1. Calculation Specifications, positions 54-59: Resulting Indicators

2. Operation Codes: EXCPT, SETON, SETOF

3. Output Specifications, position 15: Type E

When you think you understand the use of these 3 items so that you are ready to continue coding, return to this text.

* * *

Try to complete the required coding for these actions now.

1. Compute 3rd Net amount,
2. Test the result in 3rd net for zero and negative values,
3. If the result is either zero or negative, add back the deduction amount;
4. Then, for the same condition, include an exception operation, and
5. Again, for the same condition, bypass all further calculations (which we will be specifying shortly).

* * *

How many statements did you code in order to describe these steps? Did you remember to use the same name for the net pay field in every step that required it? Which indicator did you assign to test the result after subtracting the first deduction amount from the 2nd net amount? How did you bypass all further calculations after the exception operation?

In order to satisfy this series of calculations, I coded the following 5 statements. From reading this problem, I remembered that some other steps will fall between the GOTO and TAG operations, so I placed the TAG statement at the bottom of the sheet. Compare these entries to the ones you made.

RPG CALCULATION SPECIFICATIONS

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And Not	Indicators: And Not	Indicators: And Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Result Field: Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Resulting Indicators: Arithmetic: Minus	Resulting Indicators: Arithmetic: Zero	Resulting Indicators: Compare: 1>2	Resulting Indicators: Compare: 1<2	Resulting Indicators: Compare: 1=2	Resulting Indicators: Lookup(Factor 2)is: High	Resulting Indicators: Lookup(Factor 2)is: Low	Resulting Indicators: Lookup(Factor 2)is: Equal	Comments
01	C					GROSS	SUB	FIT	NET		52											
02	C					NET	SUB	SSTAX	NET													
03	C					NET	SUB	DED1	NET											21	21	
04	C		21			NET	ADD	DED1	NET													
05	C		21				EXCPT															
06	C		21				GOTO	END														
07	C																					
08	C																					
09	C																					
10	C																					
11	C																					
12	C																					
13	C																					
20	C																					
	C																					
	C																					
	C																					
	C																					
	C					END	TAG															

I selected indicator 21 (any one that has not yet been used is OK) and assigned it in columns 56-57 (minus) and 58-59 (zero). The next 3 steps then are conditioned to be performed if indicator 21 is turned ON. First, add back the amount deducted. Second, interrupt the calculations to produce an exception record. Third, bypass all further calculations. The TAG statement is not conditioned by any indicators.

If you did not include indicator assignments for all required points, correct it at this time on your coding sheets.

* * *

What you have completed for one deduction amount needs to be repeated for the other deduction amounts, with this change:

> "Assign a different indicator for the test of the result field each time."

Keeping that in mind, describe the steps to compute 4th net, 5th net and the final net amounts.

* * *

The steps that I coded are shown for your reference after you finish. How do yours compare with them? Notice that I did <u>not</u> use a GOTO statement following the exception statement for final net value. Why is this approach acceptable?

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Result Field: Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Resulting Indicators: Arithmetic: Minus	Resulting Indicators: Arithmetic: Zero	Comments
01	C					NET	SUB	DED2	NET					2222		
02	C		22			NET	ADD	DED2	NET							
03	C		22				EXCPT									
04	C		22				GOTO	END								
05	C															
06	C					NET	SUB	DED3	NET					2323		
07	C		23			NET	ADD	DED3	NET							
08	C		23				EXCPT									
09	C		23				GOTO	END								
10	C															
11	C					NET	SUB	DED4	NET					2424		
12	C		24			NET	ADD	DED4	NET							
13	C		24				EXCPT									
14	C															
15	C					END	TAG									
16	C															
17	C															

I did not include a GOTO statement following the EXCPT statement associated with final net calculation, because the next step is already the end of all calculations. A GOTO operation is used to bypass other statements. In this instance there are no further calculations and so there is no need for the additional GOTO statement. If you included a GOTO in your solution, it is correct but unnecessary.

Information: When calculations are being performed and a TAG statement is encountered, the program moves on to the statement that follows the TAG.

If the TAG statement is the last statement (either for detail-time or total-time calculations), all calculations are completed and output, if properly conditioned, is produced.

Here's something you should know about indicators.

RULE: When a Resulting Indicator (positions 54-59) is turned ON it remains on until it is turned off in one of these ways.

1. The same statement is repeated. This automatically causes the indicators assigned in columns 54-59 to be turned off before the operation is performed. After the operation, one or more assigned indicators may turn on again.

2. The same program is loaded so that the job may be started over.

3. A different operation in which the same indicator(s) is assigned, is performed.

NOTE: Two special operation codes, SETOF (set off indicators) and SETON (set on indicators) may be used by the programmer to control the condition of most indicators. We plan to cover indicator usage in more detail later on.

In our example of the net pay calculations what will cause indicators 21, 22, 23, and 24 to be turned off?

They will turn off because of point 1 in the rule:

"The same statement is repeated."

In order to be sure that the TAG statement is entered properly, we should include a "line" entry at the far left on the Calculation sheet. Notice that the last few lines on the page do not have pre-printed entries. These were left blank intentionally so that inserts as well as additions could be made by programmers. Since the entry we made for the TAG operation is the very last, specify the entry 25 in positions 3-4.

* * *

To assure yourself that your calculations satisfy the job requirements, use the following test data and "run it" through your program sequence. Use a piece of scratch paper to keep track of:

1. The setting of each assigned indicator, and

2. The value of the net pay field at each step.

#	Gross	FIT	FICA	D1	D2	D3	D4	Net Pay	Deductions
1	100	20	5	20	1	2	3	$49	All
2	200	40	10	50	50	40	20	$10	D1, D2, D3
3	100	18	5	20	50	10	2	$ 7	D1, D2
4	200	45	10	70	80	10	10	$75	D1
5	100	10	6	90	10	5	4	$84	None

NOTE: The abbreviation, FICA, is used for Social Security payments.

* * *

If you do not get the predicted answers for the Net Pay field, you should re-program your set of calculations to take care of the errors. After making changes, re-check all calculations to make sure you didn't accidentally create a new error when you fixed the old one.

We are now ready to describe the output records for the report that is to show for which employees no deductions, or only some could be taken. Each of these records is an exception to the normal payroll calculation and it has been determined by management that the list of exception records shall include:

1. Some information to identify the employee

    a. department number
    b. man number
    c. social security number

2. The amount of money earned this pay period

    a. gross pay

3. The tax amounts deducted

    a. federal income tax
    b. social security tax

4. The authorized deduction amounts

   a. deduction 1
   b. deduction 2
   c. deduction 3
   d. deduction 4

NOTE: If deduction 4 can also be taken, no exception record is printed.

5. A message that tells which, if any deductions were taken

   a. NO DEDUCTIONS TAKEN

   b. INCLUDING D1 — if only 1 deduction is taken

   c. INCLUDING D1, D2 — if only 2 deductions are taken

   d. INCLUDING D1, D2, D3 — if only 3 deductions are taken

We are not using the Auto Report feature of RPG II in solving this example, but we will want to print a report showing the exception records. What we need to do then is to position the report title and fields ourselves. An aid for doing this is a worksheet called a Print Chart, IBM form number GX20-1816. It consists of small squares that are identified by numbers across the top and on the left from top to bottom. Here's a portion of such a chart for your reference.

150/10/6 PRINT CHART PROG. ID _____ PAGE _____

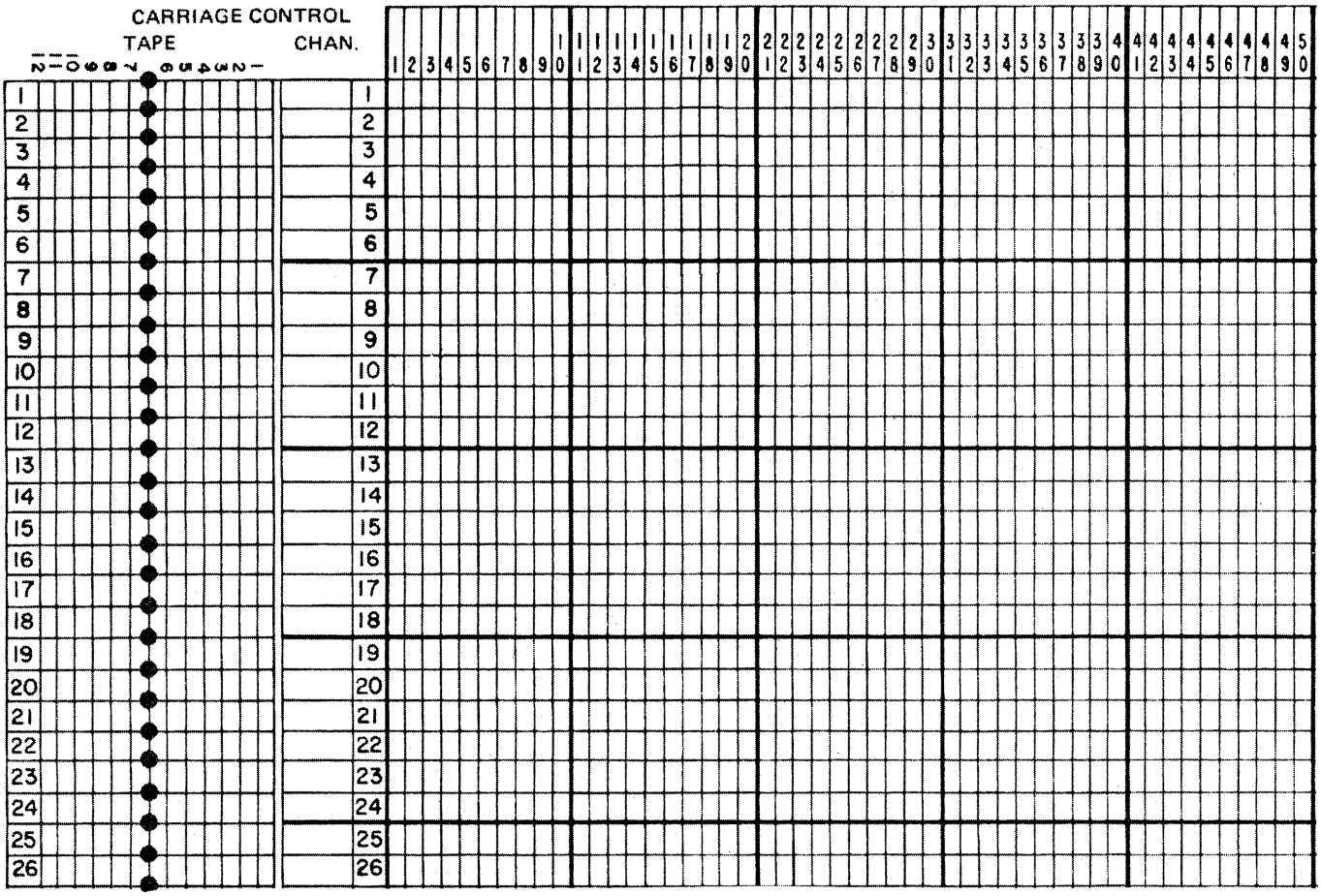
(SPACING: 150 POSITIONS AT 10 CHARACTERS PER INCH, 6 LINES PER VERTICAL INCH) DATE _____

PROGRAM TITLE _____

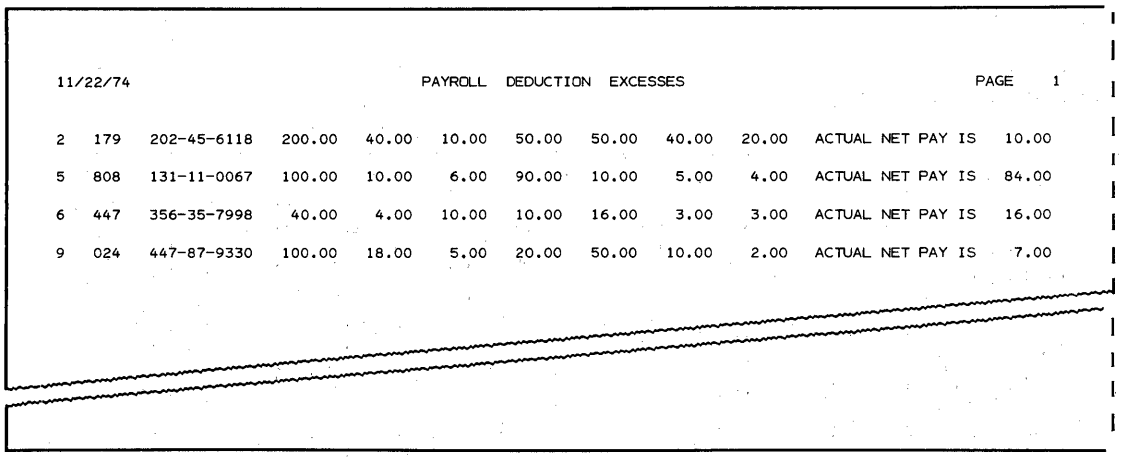
PROGRAMMER OR DOCUMENTALIST: _____

CHART TITLE _____

CARRIAGE CONTROL TAPE CHAN.



This kind of form is used to lay out heading and body line formats of reports to be produced from programs. In our report for problem 4, we will have a report title and a number of body lines as shown below.



11/22/74 PAYROLL DEDUCTION EXCESSES PAGE 1

2	179	202-45-6118	200.00	40.00	10.00	50.00	50.00	40.00	20.00	ACTUAL NET PAY IS	10.00
5	808	131-11-0067	100.00	10.00	6.00	90.00	10.00	5.00	4.00	ACTUAL NET PAY IS	84.00
6	447	356-35-7998	40.00	4.00	10.00	10.00	16.00	3.00	3.00	ACTUAL NET PAY IS	16.00
9	024	447-87-9330	100.00	18.00	5.00	20.00	50.00	10.00	2.00	ACTUAL NET PAY IS	7.00

Get a blank Print Chart and we will fill it out for use as a worksheet.

* * *

Here's how I suggest that you use such a chart. First, identify from the report sample the number of different heading lines to be printed. In this example we have only one, the report heading line. Next, print the letter H to the left of the number 6 that identifies line 6 on the print chart. Then identify the number of different body lines to be printed. In this example, all body lines are alike in format, so we need to plan for only 1 body line. In our example, this body line represents an exception record format and we will print the letter E to the left of the number 9 on line 9.

Now we are ready to fill in the blanks. Since the exception records must include many fields and some messages, we will plan it first.

The report title line is easy to center after we have laid out the body line. Here's the scheme to follow:

1. If a field contains alphameric data, print an X in the square where it shall start and print another X in the square where it shall end, and then connect the two X's with a straight line like this.

2. If a field is numeric and shall contain special edit characters such as commas, decimal points, slashes or dashes, print an X in each square that is to contain a number and then print the actual editing character(s) where it shall be printed. Here are some examples:

X,XXX.XX	Money amounts
XX.XXCR	Money amounts showing a credit
XX/XX/XX	A date field
XX-X-XXXXX	A part number

3. If a field is numeric and is to be printed with zero suppression but not have any other punctuation characters, you print a Z in each square.

Let's use that scheme and fill in the fields to be printed for our exception records. These are the fields to be included from left to right on the report.

1.	Department number	3 positions, zero suppress
2.	Man number	5 positions, zero suppress
3.	Social Security number	9 positions, separate with dashes

NOTE: Social Security numbers shall print like this:

024-68-1037

4.	Gross pay amount	5 positions, include a decimal point
5.	Federal income tax	5 positions, include a decimal point
6.	Social Security tax	5 positions, include a decimal point
7.	Deduction 1	4 positions, include a decimal point
8.	Deduction 2	4 positions, include a decimal point
9.	Deduction 3	4 positions, include a decimal point
10.	Deduction 4	4 positions, include a decimal point

Which character will you use for department number? for man number? Why? How will you set up the Social Security number field?

* * *

Department number is simply ZZZ while man number is ZZZZZ. Both fields are to be zero suppressed, but there are no puncutation characters in these fields. For the Social Security number I assume you selected XXX-XX-XXXX.

Now let's put them onto the print chart. Remember, we are planning the body line for exception fields and messages on line 9. Leave 2 spaces between fields and start in position 1 for the department number field. Put in the department number field and the man number field at this time.

* * *

If you followed the scheme correctly, you have ZZZ in positions 1 to 3; positions 4 and 5 are blank; and ZZZZZ should appear in positions 6 to 10. All of these are on line 9. If yours are different, correct them before we continue.

* * *

The Social Security number field is to include 9 digits and 2 dashes. To include the printing of high-order (left-most) zeros for numbers such as 009-27-6080, we need to include 1 extra space to the left of the field as a part of it. RPG II uses this extra position to provide for the printing of the high order zeros. OK. Leave positions 11 and 12 blank and then fill in X's and the dashes for Social Security number. Be sure to include the extra space as an X on the left.

* * *

What does your entry look like? I have four X's in 13-16, a dash in 17, two X's in 18-19, a dash in 20, and four X's in 21-24.

The next set of fields are all similar in that they are numeric fields that include a decimal point. Here's what my line looks like <u>after</u> I add the gross pay field.

```
ZZZ   ZZZZZ  XXXX-XX-XXXX  XXX.XX
```

You must remember to include the decimal point for each of these money fields and separate them with 2 spaces each time. Fill in all of the remaining input fields we listed in the book earlier and then return to this text.

* * *

Now the line should extend to position 76 and look like this.

```
ZZZ ZZZZZ XXXX-XX-XXXX  XXX.XX  XXX.XX  XXX.XX  XX.XX  XX.XX  XX.XX  XX.XX
```

The right-hand side of the report should look like this.

```
    ACTUAL NET PAY IS  210.00  NO DEDUCTIONS TAKEN

    ACTUAL NET PAY IS    7.00  INCLUDING D1

    ACTUAL NET PAY IS   75.00  INCLUDING D1, D2

    ACTUAL NET PAY IS   24.00  INCLUDING D1, D2, D3
```

What do you call the message that is the same on every line? What field's value is to be printed? How many different messages must be planned on the print chart?

* * *

If a message appears on every body line it is called a constant. The field is the Net Pay Amount field. There are 4 different messages to be included in the plan. Now we will add these to the print chart.

RULE: Print the exact constant or message on your chart. Include every possible message.

Since position 76 is the last X on line 9, positions 77 and 78 should be left blank and then the constant should be entered. Include the Net Pay field and the first message on line 9. Directly underneath the first message (on line 10) print the second message. Likewise, print the third message on line 11 and the fourth message on line 12. Complete your plan for all these items. Again, leave 2 spaces between constants, fields and messages.

* * *

Your plan, starting in position 79, should look like this.

```
ACTUAL NET PAY IS  XXX.XX  NO DEDUCTIONS TAKEN
                           INCLUDING D1
                           INCLUDING D1, D2
                           INCLUDING D1, D2, D3
```

That should finish the plan for the exception record formats. Examine your print chart to answer these questions.

1. In which position does the constant start?

2. Where does the constant end?

3. In which position is the decimal point for the Net Pay field?

4. Where does the Net Pay field end?

5. What is the right-most position for each message?

* * *

If you left 2 spaces between each constant, field and message, your answers should be:

1. 79
2. 95
3. 101
4. 103
5. 124, 117, 121 and 125

Earlier we identified line 6 as the place where we will describe the report title. Suppose the title looks like this:

```
10/01/74          PAYROLL   DEDUCTION   EXCESSES    PAGE    1
```

We want to center the title. Since the longest exception line in the body goes to position 125, half of that gives us 63 (rounded up). The report title has 30 characters including 3 spaces between constants. So we simply subtract 15 (half of 30) from 63 (the middle of the longest line) to find position 48 where we start filling in the first word of the title. Also, include a 6-position date field (plus 2 slashes to separate it into month, day and year) starting in position 11, the constant PAGE in positions 111-114, and the page number field of 4 positions in 115-118. The page number field is to be zero suppressed.

Fill in every part of the report title line <u>on line 6</u> of your print chart.

* * *

The print chart is a worksheet for planning purposes. You can add notes and other information for later reference as you wish. Here's what we usually do.

1. Put the field name in parentheses under or near the place where the field is positioned.

2. Add a note about spacing between heading and body lines.

3. Write the formulas used to do calculations if there are any.

4. Add a note about the condition under which particular messages ought to be printed.

5. Fill in the program name, date, programmer's name, etc., in the spaces provided at the top of the print chart.

Take a few minutes to identify each field on your print chart by referencing the Input and Calculation sheets you have already coded. Also remember that the system date field named UDATE and the page numbering field named PAGE are special field names reserved for RPG II use. If you wish, make notes about other facts about this problem; for example, "double space the body lines".

* * *

Before coding the exception report Output specifications, you need to describe this output file on the File Description sheet. Here's what you need to include:

1. a file name (in 7-14)

2. a file type (in 15)

3. a record length (in 24-27) for the 132-position print records

4. an Overflow Indicator (in 33-34); choose any one of these: OA, OB, OC, OD, OE, OF, OG or OV

5. a device name (in 40-46)

Add this line of coding to your File Description sheet.

* * *

You should keep your print chart nearby as you need to refer to it as you complete the output coding. First, describe the heading line. It shall be printed on the first page or on overflow pages of the report. Use the filename and the overflow indicator you just specified on the File Description sheet. Include entries to skip to line 06 before printing and space 3 (to get to line 9) after printing it.

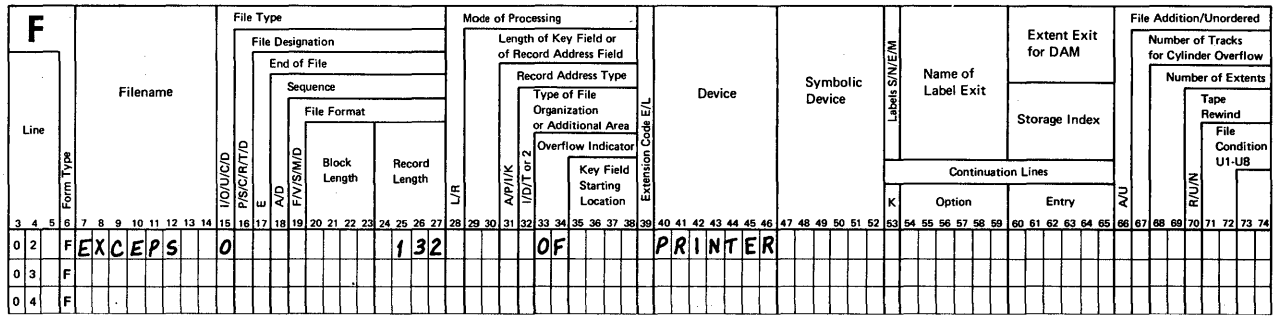
* * *

Next, specify each field and constant that is a part of the heading line shown on your print chart. Include an entry for "End Position in Output Record" (in 40-43) for each item. End position refers to the right-hand end of a field or constant. Also include appropriate edit codes for the date field and the page number field.

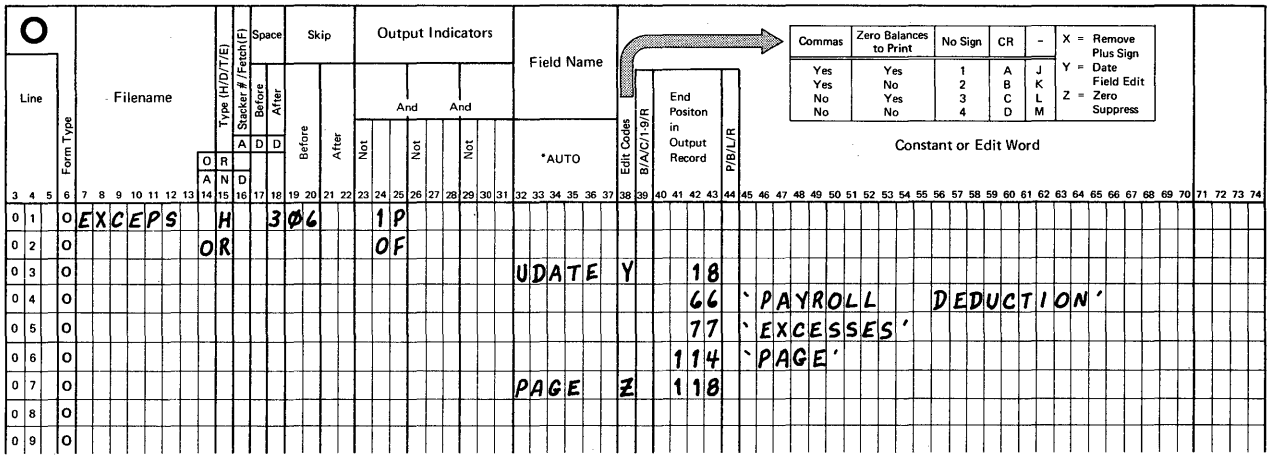
* * *

Now compare your entries to those on the following page. Did you include every kind of entry shown? Your filename and field names must agree with the entries on your Input and Calculation sheets.

File Description Specification



RPG OUTPUT SPECIFICATIONS



RULE: When no entries are made for spacing and skipping in an "OR" line, RPG II automatically uses the same numbers described in the line above it.

Suggestion:

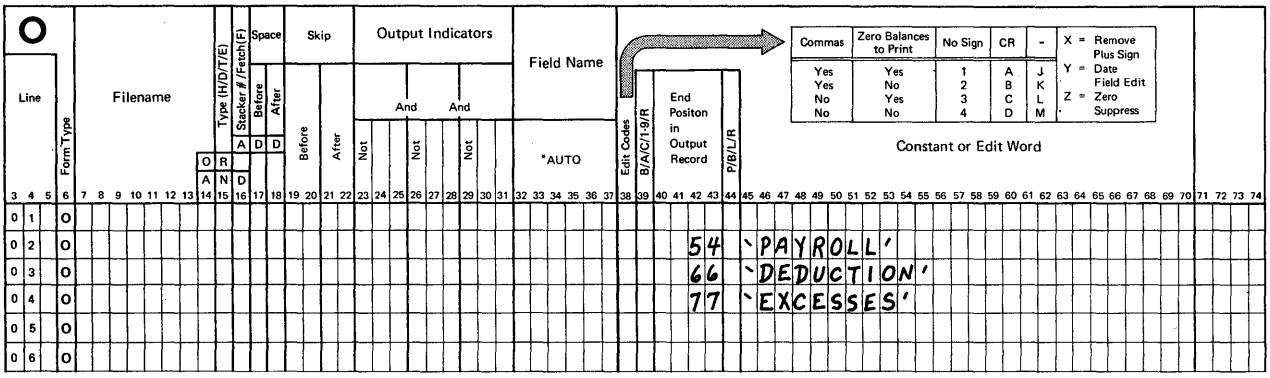
Select edit codes rather than edit words for date and page number fields.

Code as much of a constant as is convenient in one line.

RULE: The overflow indicator used to condition output must be the one assigned on the File Description sheet.

You could have coded the report title as 3 separate constants in which case your entries would be like this.

RPG OUTPUT SPECIFICATIONS



You should be sure that your coding is accurate before continuing. If you are unsure of any entries, re-read the last few pages.

We are ready to concentrate on describing the printing of exception records. How many variations are there on your print chart?

* * *

I have 4 variations because of the different messages that are to be included. Re-examine your calculation sheet and then note, to the far right side on the print chart, which Resulting Indicator you assigned for each situation that has a corresponding message. My indicators are as follows:

Message	Indicator ON
NO DEDUCTIONS TAKEN	21
INCLUDING D1	22
INCLUDING D1, D2	23
INCLUDING D1, D2, D3	24

From this analysis I determine that I am to have an exception line printed if indicator 21 is ON, or if indicator 22 is ON, or if indicator 23 is ON, or if indicator 24 is ON. The note on my print chart says I need to double space body lines. Since we are describing Exception output records, enter code letter E under "Type" in position 15. Complete this coding now.

* * *

Does your coding look like this? Did you remember the rule about spacing entries?

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker #/Fetch(FI)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Field Name	Edit Codes	End Position in Output Record	P/B/L/R
0 1	O		E			2			21					
0 2	O		OR						22					
0 3	O		OR						23					
0 4	O		OR						24					
0 5	O													
0 6	O													

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Constant or Edit Word

At this time, specify each field or constant that is to be a part of the exception line for the first possible situation. Include:

1. the field name
2. an End Position in Output Record
3. the constant

For convenience, I work from left to right on the print chart as I code. Use one coding line for each field or constant.

* * *

Double check your names for exact spellings. Re-examine all end positions on the chart. Make sure that each constant has a single quote mark ahead of it and following it.

Next, specify edit codes wherever desired.

* * *

No edit code exists in RPG II to provide for inserting dashes for the Social Security number field, so what do we do? We will describe an "Edit Word" rather than an edit code for this field.

You will recall that we added an extra space to the left of this field when we planned for its use on the print chart. Now we'll make use of it to provide for proper punctuation and zero suppression.

RULE: All leading (left-most) zeros are suppressed unless a zero or asterisk is specified in the edit word.

RULE: Any zeros (in the data field) following the left-most zero or asterisk position (in the edit word) are treated as constants; that is, they will be printed.

Suppose I used an edit word like this (Ø stands for a blank space):

'ØØØØ-ØØ-ØØØØ'

and the Social Security number was 007290625, what would be printed according to the rules? What would happen if I used 'ØØOØ-ØØ-ØØØØ' and 003037726?

In the first example, the printed Social Security number would be

007-29-0625

In the second example, it would be

3-03-7726

Why not use the edit word 'OØØ-ØØ-ØØØØ' for a number like 084900265? I used a shorter edit word in this example and so the zero in the field on the left side will be suppressed rather than be printed. Here's what would print in this case.

84-90-0265

To sum up about this edit word:

RULE: An extra space can be left in the edit word if the first character in the edit word is a zero. In this case, the field to be edited is not zero suppressed, but all other specified editing is performed.

At this time, specify the correct edit word for your 9-position Social Security number field.

* * *

We need to add an entry to control "when" the message, NO DEDUCTIONS TAKEN, shall be printed. What is that entry? For my example, it is "when indicator 21 is turned ON". How can this be specified?

RULE: The printing of a field or constant may be conditioned by specifying one or more indicators on the same line on which that field or constant is specified. The entry is made in positions 23-31.

Since my indicator for the message is 21, what should I specify? Enter 21 in positions 24-25 on the line where the constant is coded.

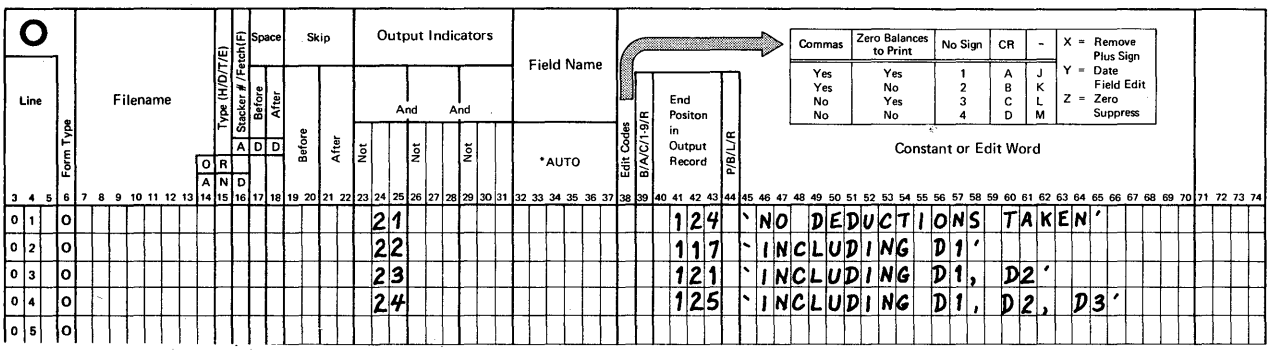
Re-examine your print chart. Which part of exception records needs to be conditioned as a field or constant for every possible record that shall be printed?

Only the messages. At this time specify the constants for all of the messages and condition each one with the proper indicator.

* * *

Your entries for the messages should be similar to these. Check yours carefully.

RPG OUTPUT SPECIFICATIONS



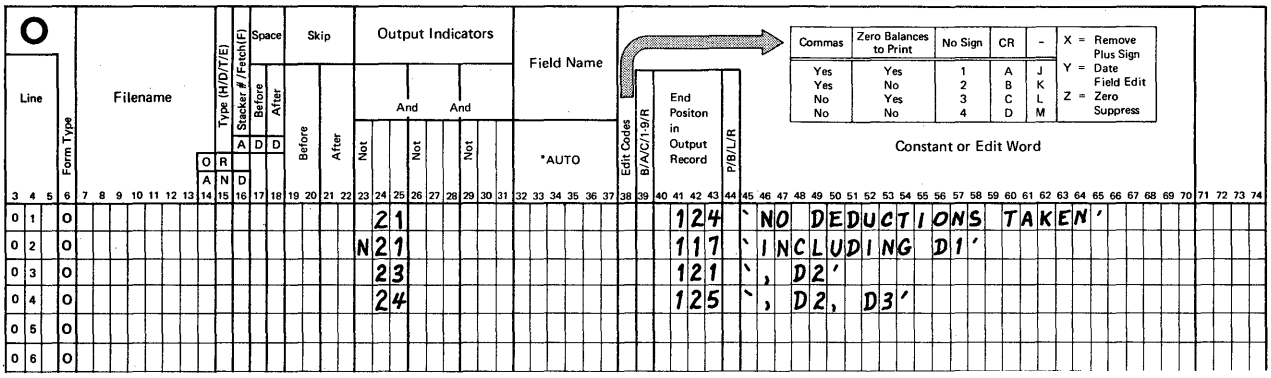
Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name *AUTO	Edit Codes	B/A/C/1-9/R	End Positon in Output Record	P/B/L/R	Constant or Edit Word
01	O								21						124		'NO DEDUCTIONS TAKEN'
02	O								22						117		'INCLUDING D1'
03	O								23						121		'INCLUDING D1, D2'
04	O								24						125		'INCLUDING D1, D2, D3'
05	O																

Commas | Zero Balances to Print | No Sign | CR | -
Yes | Yes | 1 | A | J
Yes | No | 2 | B | K
No | Yes | 3 | C | L
No | No | 4 | D | M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

What do you think the following entries would do?

RPG OUTPUT SPECIFICATIONS



Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name *AUTO	Edit Codes	B/A/C/1-9/R	End Positon in Output Record	P/B/L/R	Constant or Edit Word
01	O								21						124		'NO DEDUCTIONS TAKEN'
02	O								N21						117		'INCLUDING D1'
03	O								23						121		', D2'
04	O								24						125		', D2, D3'
05	O																
06	O																

Commas | Zero Balances to Print | No Sign | CR | -
Yes | Yes | 1 | A | J
Yes | No | 2 | B | K
No | Yes | 3 | C | L
No | No | 4 | D | M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Looks to me like it will accomplish the same thing that we had in mind. This was a sneaky way to do the same thing, but it's not as easy to follow or invent.

Have you ever tried to solve a problem before anyone else knew it existed? That's what we are going to do next, hopefully.

When you think of your program as it will be running, you can predict that some undesirable events may occur. In our example, let's say that one record in which deductions 1, 2 and 3 were taken (but not deduction 4) is followed immediately by a record in which only deduction 1 can be taken. Because we used the GOTO operation, indicator 23 was not turned off before indicator 21 was turned on for the next record. When such a situation exists (2 or more resulting indicators are on at the same time), it is possible that we get either:

1. the wrong message, or

2. a mixture of both messages

There is the special operation called SETOF that may be used to prevent such an occurrence. It causes 1, 2 or 3 designated indicators to be turned off. Look at your Calculation sheet. Think through the steps you have coded.

1. Will the test for Resulting Indicators occur for every record?

2. If the answer is "yes", the assigned indicator is automatically turned <u>off</u> before the operation is performed, <u>and then</u> the test is done to see if that indicator should be turned on after the operation.

3. If the answer is "no", the programmer should probably use the SETOF operation to make sure it is turned off before the operation is done.

In Problem 4, the test for resulting indicator condition is made in 3 cases at steps bypassed by a GOTO operation. These hidden steps will not occur for every record. Therefore, I, the programmer, should set off indicators 22, 23 and 24 every time a new record is processed. I will place the SETOF instruction ahead of all other calculations so that I'm sure only 1 message will print for each exception record. Here's how.

1. Enter SETOF in 28-32.

2. Enter one, two or three indicators to be turned off in 54-55, 56-57 and 58-59.

To set off more than 3 indicators, I need to specify a number of SETOF operations.

<u>Hint</u>: Use SETOF only when needed. It takes space and time to perform its function.

Oh, Oh. We've already used the first coding line on the Calculation sheet! How can I specify the SETOF instruction so that it will be positioned correctly for compilation?

Look at your Calculation sheet again. There should be 4 unused and unnumbered coding lines near the bottom of your sheet (the TAG statement is last). To use SETOF, code it as I have directed on one of the unused lines and then enter a number in positions 3-5 of that line to indicate where it really belongs in the sequence of calculation steps. The top line is preprinted as 01Ø in 3-5 which is like 010. So you can use 001 for your entry in 3-5 for the SETOF operation statement. Be sure to include the 3 indicators you assigned for the last 3 test steps.

* * *

All coding is now completed. As we did in an earlier example, enter page numbers and the program name at the upper right of each page in your program. Call this program C5P4. The correct order of specifications for numbering is:

1. File Description
2. Input
3. Calculation
4. Output

* * *

The reason for including this chapter was to simulate RPG II programming activity in the real world. You were exposed to four different types of data processing problems and asked to code solutions to them. I've decided to repeat each problem statement here and ask that you review each statement along with your coded solutions. Take time to make sure that you understand why each entry on each line of each specification sheet is needed. You may wish to re-read certain parts of this text and the RPG II reference manual as you participate in this review.

PROBLEM 1: C5P1

"A file of 80-character disk records contains information about a certain company's accounts receivable. An 'Accounts Receivable Register' is printed for management each month so that it may be determined how much money has not yet been collected from customers who purchased goods from the company."

Directions:

1. Use /COPY to include the Input File Description and Input records in your program. These entries are already cataloged under the name ARIN.

2. Describe the necessary output file.

3. Use *AUTO to describe the headings and body lines of the register.

4. Refer to the sample report to code Output entries, and to the cataloged RPG II entries for the /COPY entry.

Sample Report Page

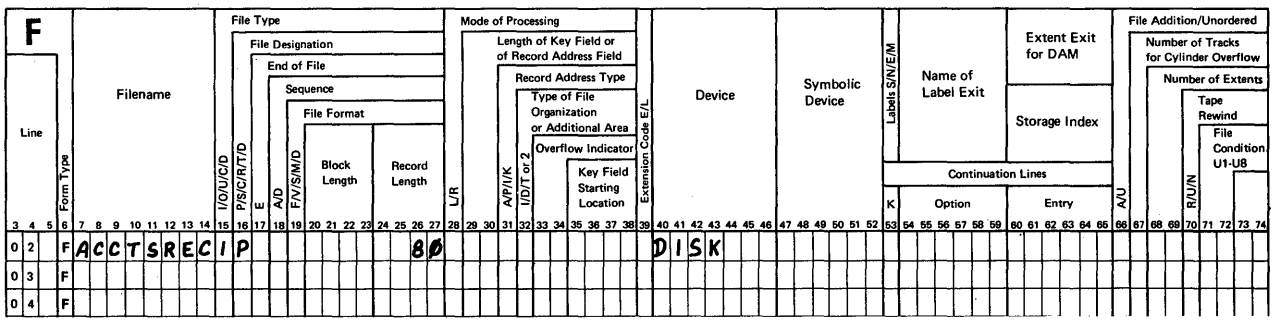
ACCOUNTS RECEIVABLE REGISTER

CUSTOMER NUMBER	CUSTOMER NAME	STATE	CITY	INVOICE NUMBER	INVOICE DATE	INVOICE AMOUNT
1281	AMERICAN STEEL CO	36	49	11666	11/23/67	640.31
1281	AMERICAN STEEL CO	36	49	12336	12/30/67	909.04
2179	APALACHIN LUMBER CO	4	227	9852	9/15/67	469.20
2283	B J E SERVICE CORP	22	37	12332	12/29/67	1,474.78
11905	CHALLIS ALMERS	47	77	10901	10/18/67	27.63
29031	DENNIS MFG CO	6	63	11615	11/14/67	440.12

17,524.23 *

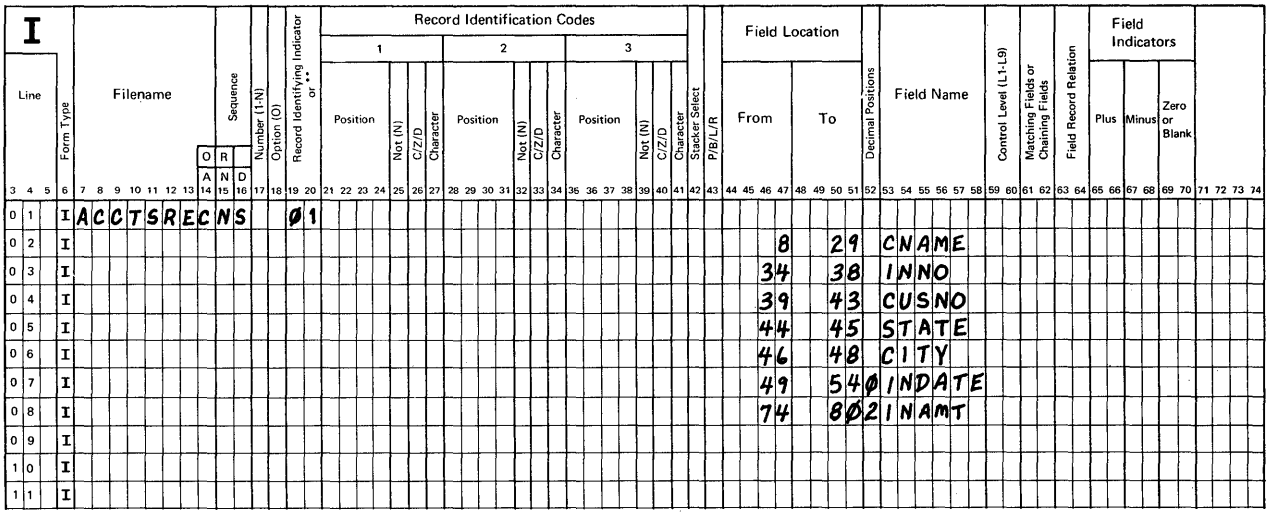
Cataloged as ARIN

File Description Specification



Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	R/U/N	Tape Rewind	File Condition U1-U8	File Addition/Unordered
02	F	ACCTSREC	I	P					80								DISK														
03	F																														
04	F																														

RPG INPUT SPECIFICATIONS



Line	Form Type	Filename	Sequence (OR AND)	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	ACCTSREC	NS			01																								
02	I																				8	29		CNAME						
03	I																				34	38		INNO						
04	I																				39	43		CUSNO						
05	I																				44	45		STATE						
06	I																				46	48		CITY						
07	I																				49	54	0	INDATE						
08	I																				74	80	2	INAMT						
09	I																													
10	I																													
11	I																													

Problem 1: Book Solution

File Description Specification

Page 01 of 2 — Program Identification: C5P1

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/O/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	R/U/N	Tape Rewind	File Condition U1-U8	File Addition/Unordered
02	F	/COPY F1,ARIN																														
03	F	REGISTERO							132								PRINTER															
04	F																															
05	F																															
06	F																															
07	F																															
08	F																															
09	F																															
10	F																															
	F																															
	F																															

RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

Program | Punching Instruction | Graphic | Card Electro Number

Programmer | Date | Punch

Page 02 of 2 — Program Identification: C5P1

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	REGISTERH										*AUTO					
02	O																'ACCOUNTS RECEIVABLE '
03	O																'REGISTER'
04	O		D		2				01			*AUTO					
05	O											CUSNO					'CUSTOMER'
06	O													C			'NUMBER'
07	O											CNAME					'CUSTOMER NAME'
08	O											STATE					'STATE'
09	O											CITY					'CITY'
10	O											INNO					'INVOICE'
11	O													C			'NUMBER'
12	O											INDATEY					'INVOICE'
13	O													C			'DATE'
14	O											INAMT		A			'INVOICE'
15	O													C			'AMOUNT'
16	O																
17	O																

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

PROBLEM 2: C5P2

"A master file of subscriber records is stored as an indexed disk file. Each record is 85 characters long and includes a key field in positions 1-7. The records in this file are updated on a daily basis for two kinds of transactions: renewal of subscription and change of address. The changes are keyed in from the console as an interactive data entry file."

These are the programming requirements:

1. Update the master file randomly using the chaining method.
2. For renewals, change the expiration date.
3. For changes of address, replace the street and city/state/zip code fields.
4. Include a list of keyed transactions.
5. If no master record is found for a particular transaction, print a message to that effect.

Master File

Positions	Field
1- 7	Subscriber Number (key field)
8-31	Subscriber Name
32-55	Street Address
56-79	City, State and Zip
80-85	Expiration Date of Subscription

Transaction File

Type	Positions	Field
Renewal	1	Code - letter R
	2- 8	Subscriber Number
	9-10	Number of Years Renewed
Change of Address	1	Code - letter C
	2- 8	Subscriber Number
	9-32	New Street Address
	33-56	New City, State and Zip

Problem 2: Book Solution, Page 1

File Description Specification

Page 01 of 4 — Program Identification C5P2

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/O/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind R/U/N	File Condition U1-U8
02	F	SUBMAS	U	C					85	R	7	A	I			1	DISK													
03	F	CHANGES	I	P				62	60								CONSOLE													
04	F	LIST	O						132								PRINTER													
05	F																													
06	F																													
07	F																													
08	F																													
09	F																													
10	F																													
	F																													
	F																													

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

Program / Programmer / Date / Punching Instruction / Graphic Punch / Card Electro Number

Page 02 of 4 — Program Identification C5P2

Line	Form Type	Filename	Sequence (AND/OR)	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	CHANGES	AA			01	1		C	R																				
02	I																				1	1		CODE						
03	I																				2	80		SUBSNO						
04	I																				9	100		YEARS						
05	I		BB			02	1		C	C																				
06	I																				1	1		CODE						
07	I																				2	80		SUBSNO						
08	I																				9	32		NEWST						
09	I																				33	56		NEWCTY						
10	I	SUBMAS	CC			03																								
11	I																				84	85		EXDATE						
12	I																													

Problem 2: Book Solution, Page 2

RPG CALCULATION SPECIFICATIONS

GX21-9093-2 UM/050* Printed in U.S.A.
*No. of forms per pad may vary slightly.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 03 of 4 Program Identification C5P2

Line	Form Type	Control Level (LO-L9, LR, SR, AN/OR)	Indicators: And / Not	And / Not	Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Minus	Zero; Compare 1>2	1<2	1=2; Lookup(Factor 2) High	Low	Equal	Comments
01	C		01			SUBSNO	CHAINSUBMAS												11	
02	C		01N11			EXDATE	ADD YEARS		EXDATE											
03	C		02			SUBSNO	CHAINSUBMAS												22	
04	C																			
05	C																			

RPG OUTPUT SPECIFICATIONS

GX21-9090-2 U/* 050* Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 04 of 4 Program Identification C5P2

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress

Constant or Edit Word

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch (F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And / Not	And / Not	Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	SUBMAS	D						01N11								
02	O											EXDATE			85		
03	O		D						02N22								
04	O											NEWST			55		
05	O											NEWCTY			79		
06	O	LIST	D		2				01			*AUTO					
07	O		OR						02								
08	O											CODE					'CODE'
09	O											SUBSNO	Z				'SUBSCRIBER'
10	O								01			YEARS	Z				'YEARS'
11	O								01	11							'NO MASTER FOUND'
12	O								02			NEWST					'STREET'
13	O								02			NEWCTY					'CITY / STATE'
14	O								02	23							'NO MASTER FOUND'
15	O																
16	O																

PROBLEM 3: C5P3

"Compute customer electric usage bills. Refer to this table of rates and then describe the necessary calculations."

Usage	Rate
1- 50 kwh	$.05 each
51-100 kwh	$.04 each over the first 50, + $ 2.50
101-300 kwh	$.03 each over the first 100, + $ 4.50
over 300 kwh	$.025 each over the first 300, + $10.50

Directions:

1. Use /COPY for all coding except Calculations. Cataloged items are:

   ALLFILES for all File Descriptions

   INRECS for Input records and fields

   OUTRECS for Output records and fields

2. Name the result field for calculated usage, USE

3. Name the result field for calculated costs, BILL

4. Input records include these fields:

   NEW, for the new period's meter reading

   LAST, for the last period's meter reading

Problem 3: Book Solution, Page 1

File Description Specification

Page 01 of 4 — Program Identification C5P3

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/O/T or 2	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8	R/U/N
02	F	/COPY F1,ALLFILES																											
03	F																												
04	F																												

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9094-2 U/M 050 Printed in U.S.A.

Program | Punching Instruction | Graphic | Card Electro Number

Programmer | Date | Punch

Page 02 of 4 — Program Identification C5P3

Line	Form Type	Filename	Sequence AND	Sequence OR	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank
01	I	/COPY F1,INRECS																												
02	I																													
03	I																													

Problem 3: Book Solution, Page 2

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9093-2 UM/050* Printed in U.S.A.
*No. of forms per pad may vary slightly.

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 03 of 4 | Program Identification C5P3

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: 1>2	1<2	1=2	Lookup(Factor 2)is: High	Low	Equal	Comments
01	C				NEW	SUB	LAST	USE		50				20							
02	C		20		NEW	ADD	100000	BIG		60											
03	C		20		BIG	SUB	LAST	USE													
04	C		20			Z-ADD	0	BIG													
05	C					Z-ADD	USE	KWH		50											
06	C				USE	COMP	50							30							
07	C		N30		USE	MULT	.05	BILL		52											
08	C		N30			GOTO	END														
09	C				USE	COMP	100							40							
10	C		N40		USE	SUB	50	OVER		50											
11	C		N40		OVER	MULT	.04	BILL													
12	C		N40		BILL	ADD	2.50	BILL													
13	C		N40			GOTO	END														
14	C				USE	COMP	300							30							
15	C		N50		USE	SUB	100	OVER													
16	C		N50		OVER	MULT	.03	BILL													
17	C		N50		BILL	ADD	4.50	BILL													
18	C		N50			GOTO	END														
19	C				USE	SUB	300	OVER													
20	C				OVER	MULT	.025	BILL			H										
21	C				BILL	ADD	10.50	BILL													
22	C				END	TAG															
	C																				
	C																				

RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 04 of 4 | Program Identification C5P3

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And	And	Field Name	Edit Codes	End Positon in Output Record	.P/B/L/R	Constant or Edit Word
01	O	/COPY F1,OUTRECS													
02	O														
03	O														
04	O														
05	O														
06	O														
07	O														

Commas: Yes/Yes/No/No | Zero Balances to Print: Yes/No/Yes/No | No Sign: 1/2/3/4 | CR: A/B/C/D | -: J/K/L/M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

PROBLEM 4: C5P4

"An indexed disk file of 100-character records is to be copied for reference. Its key field is in positions 1-8. The new file will contain identical records, but is to be organized sequentially. Compute Net Pay using this formula.

NET = GROSS - FIT - FICA - D1 - D2 - D3 - D4

All deductions are to be taken from gross pay unless doing so causes the net pay amount to become zero or negative in value. If that happens, add back the last deduction, print an exception record on a special report, and bypass all further calculations for that record. Include messages on the exception report to indicate which, if any, deductions were actually taken."

Directions:

1. Do not use any Auto Report function.

2. Prepare a print chart to represent heading and body lines of the exception record report.

3. Include UDATE and PAGE fields on the heading line.

11/22/74 PAYROLL DEDUCTION EXCESSES PAGE 1

2	179	202-45-6118	200.00	40.00	10.00	50.00	50.00	40.00	20.00	ACTUAL NET PAY IS	10.00
5	808	131-11-0067	100.00	10.00	6.00	90.00	10.00	5.00	4.00	ACTUAL NET PAY IS	84.00
6	447	356-35-7998	40.00	4.00	10.00	10.00	16.00	3.00	3.00	ACTUAL NET PAY IS	16.00
9	024	447-87-9330	100.00	18.00	5.00	20.00	50.00	10.00	2.00	ACTUAL NET PAY IS	7.00

Problem 4: Book Solution, Page 1

File Description Specification

Page 01 of 5 — Program Identification: C5P4

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM / Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	PAYINDEX	I	P					100		BA	I		1	DISK													
03	F	PAYSEQ	O						100						DISK													
04	F	EXCEPS	O						132				OF		PRINTER													
05	F																											
06	F																											

RPG INPUT SPECIFICATIONS

GX21-9094-2 U/M 050* Printed in U.S.A.

IBM International Business Machine Corporation

Program | Punching Instruction | Graphic | Card Electro Number

Programmer | Date | Punch

Page 02 of 5 — Program Identification: C5P4

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	PAYINDEX	AA			01																								
02	I																				1	100		RECORD						
03	I																				1	30		DEPT						
04	I																				4	80		NUMBER						
05	I																				9	170		SOCSEC						
06	I																				18	222		GROSS						
07	I																				23	272		FIT						
08	I																				28	322		STAX						
09	I																				53	562		DED1						
10	I																				57	602		DED2						
11	I																				61	642		DED3						
12	I																				65	682		DED4						
13	I																													
14	I																													
15	I																													
16	I																													
17	I																													

Problem 4: Book Solution, Page 2

GX21-9093-2 UM/050* Printed in U.S.A.
*No. of forms per pad may vary slightly.

IBM International Business Machine Corporation

RPG CALCULATION SPECIFICATIONS

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 03 of 5 Program Identification C5P4

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Resulting Indicators: Arithmetic: Minus	Resulting Indicators: Arithmetic: Zero	Resulting Indicators: Compare: 1>2	Resulting Indicators: Compare: 1<2	Resulting Indicators: Compare: 1=2	Resulting Indicators: Lookup(Factor 2)is: High	Resulting Indicators: Lookup(Factor 2)is: Low	Resulting Indicators: Lookup(Factor 2)is: Equal	Comments
01	C					GROSS	SUB	FLT	NET		52											
02	C					NET	SUB	SSTAX	NET													
03	C					NET	SUB	DED1	NET											21	21	
04	C		21			NET	ADD	DED1	NET													
05	C		21				EXCPT															
06	C		21				GOTO	END														
07	C					NET	SUB	DED2	NET											2222		
08	C		22			NET	ADD	DED2	NET													
09	C		22				EXCPT															
10	C		22				GOTO	END														
11	C					NET	SUB	DED3	NET											2323		
12	C		23			NET	ADD	DED3	NET													
13	C		23				EXCPT															
14	C		23				GOTO	END														
15	C					NET	SUB	DED4	NET											2424		
16	C		24			NET	ADD	DED4	NET													
17	C		24				EXCPT															
18	C					END	TAG															
19	C																					
20	C																					
001	C						SETOF													222324		
	C																					
	C																					
	C																					
	C																					

Problem 4: Book Solution, Page 3

RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9090-2 U/M 050* Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number

Page 04 of 5 | Program Identification C5P4

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name / *AUTO	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	PAYSEQ	OR AND D						01								
02	O											RECORD			106		
03	O	EXCEPS	H		3	6			1P								
04	O		OR						OF								
05	O											UDATE	Y		18		
06	O														66		'PAYROLL DEDUCTION'
07	O														77		'EXCESSES'
08	O														114		'PAGE'
09	O											PAGE	Z		118		
10	O		E		2				21								
11	O		OR						22								
12	O		OR						23								
13	O		OR						24								
14	O											DEPT	Z		3		
15	O											NUMBERZ			10		
16	O											SOCSEC			24		'0 - - '
17	O											GROSS	1		32		
18	O											FIT	1		40		
19	O											SSTAX	1		48		
20	O											DED1	1		55		
	O											DED2	1		62		
	O											DED3	1		69		
	O											DED4	1		76		
	O														95		'ACTUAL NET PAY IS'
	O											NET	1		103		

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress

RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9090-2 U/M 050* Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number

Page 05 of __ | Program Identification C5P4

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name / *AUTO	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O								21						124		'NO DEDUCTIONS TAKEN'
02	O								22						117		'INCLUDING D1'
03	O								23						121		'INCLUDING D1, D2'
04	O								24						125		'INCLUDING D1, D2, D3'
05	O																
06	O																

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress

Chapter 5: Summary

This chapter was designed to provide practice in solving data processing problems that involve the printing of reports, updating disk file records, and the describing of complex series of calculations. One new consideration, the description of exception output records was brought out in the fourth example.

At this point in your studies you should be able to specify solutions to problems of the types presented in chapters 2, 3, 4 and 5. You would not be expected to do them from memory, rather, you would refer to the System/32 RPG II Language manual as needed.

Here is the list of new coding entries you used in chapter 5.

FILE DESCRIPTION

No new entries.

INPUT

No new entries.

CALCULATION

28 - 32	Operation (EXCPT, SETOF)
54 - 59	Resulting Indicators (Minus, Zero; for subtract), (All three for SETOF)

OUTPUT

15	Type (E - exception output record)
38	Edit Codes (Z)
39	C for continuation line of an Auto Report constant
45 - 70	Edit Word (Øbbb-bb-bbbb for social security number)

You've come a long way into RPG II programming and should know that the bulk of this course is behind you. Here's a list of all the kinds of coding you've seen and done in just a short time.

1. File Description Sheet - describing files

   a. Input files - Console, Disk

   b. Output files - Printer, Disk

   c. Update disk files, including the addition of records.

   d. Input files designated as primary, secondary or chaining.

   e. End of file control.

   f. Block length and record length.

   g. Random processing of indexed file records on update.

   h. Page overflow indicator assignment for printer files.

2. Input Sheet - describing input records

   a. Assigning Record Identifying Indicators

   b. Designated Record Identification Codes

   c. Identifying field locations.

   d. Designating fields as either numeric or alphameric.

   e. Assigning level 1 control for control breaks.

   f. Assigning M1 for matching fields

3. Calculation Sheet - describing operations

   a. Use of indicators for either "detail-time" or "total-time" calculations.

   b. Using numeric literals as factors.

   c. Operations.

      1) ADD

      2) SUB

      3) MULT

      4) DIV

      5) MVR

      6) Z-ADD

      7) Z-SUB

8) SQRT

9) MOVE

10) MOVEL

11) COMP

12) GOTO

13) TAG

14) CHAIN

15) EXCPT

16) SETOF

d. Half adjust numeric result field values.

e. Test result through Resulting Indicator assignment.

4. Output Sheet - describing output records

a. Heading (H), detail (D), total (T), and exception (E) types

b. OR lines

c. Adding records to an update file

d. Spacing and skipping on report pages

e. Output indicators: 1P, OF, 01, 02, L1, MR and LR

f. *AUTO usage

g. Special fields: UDATE & PAGE

h. Edit codes: 1, Y, Z

i. Resetting a field value ("blank after")

j. Edit words: '0bbb-bb-bbbb'

k. Constants

l. Auto report

1) Field value accumulation code (A)

2) Continuation column heading constants code (C)

5. Correspondence of entries from coding sheet to coding sheet.

    a. Filenames

    b. Field names

    c. Indicators assigned and used

6. The generated program RPG II logic cycle.

7. The Auto Report function /COPY as used to insert cataloged statements into new programs.

Are you surprised at how many new things you've studied and learned? Keep in mind that RPG II is a descriptive language and when you concentrate first on understanding the problem requirements, the coding is relatively simple.

There is no self test for this chapter. In the next chapter you will need a few blank Extension specification sheets as we will be describing tables of data and the look-up operation, LOKUP.

Chapter 6: Tables and Arrays (1.5 to 3 hours)

There are data processing problems in which tables of data are useful in the processing of data records. I'll use a somewhat different approach in introducing the topic of tables. I'll start by stating a problem and then show you a coded solution. After that I'll tell you about the entries that deal with the table data. We will be referring to an RPG II specification form called the Extension specification sheet in addition to those with which you are already familiar. Here's the problem.

> Print a list of items as ordered by customers. The list is to look like the sample report. One set of table data contains 50 item number entries, one for each item that is for sale. A second set of table data contains 50 corresponding cost entries. Here is a partial list of table data from both of them.

Table of Items	Table of Costs
12354	2.50
22615	7.79
74002	9.85
33675	.25

> Input records contain the item number and the quantity sold. In order to print the report, it is necessary to look up the unit price for the item and then multiply this price by the quantity to find out the amount of the sale.

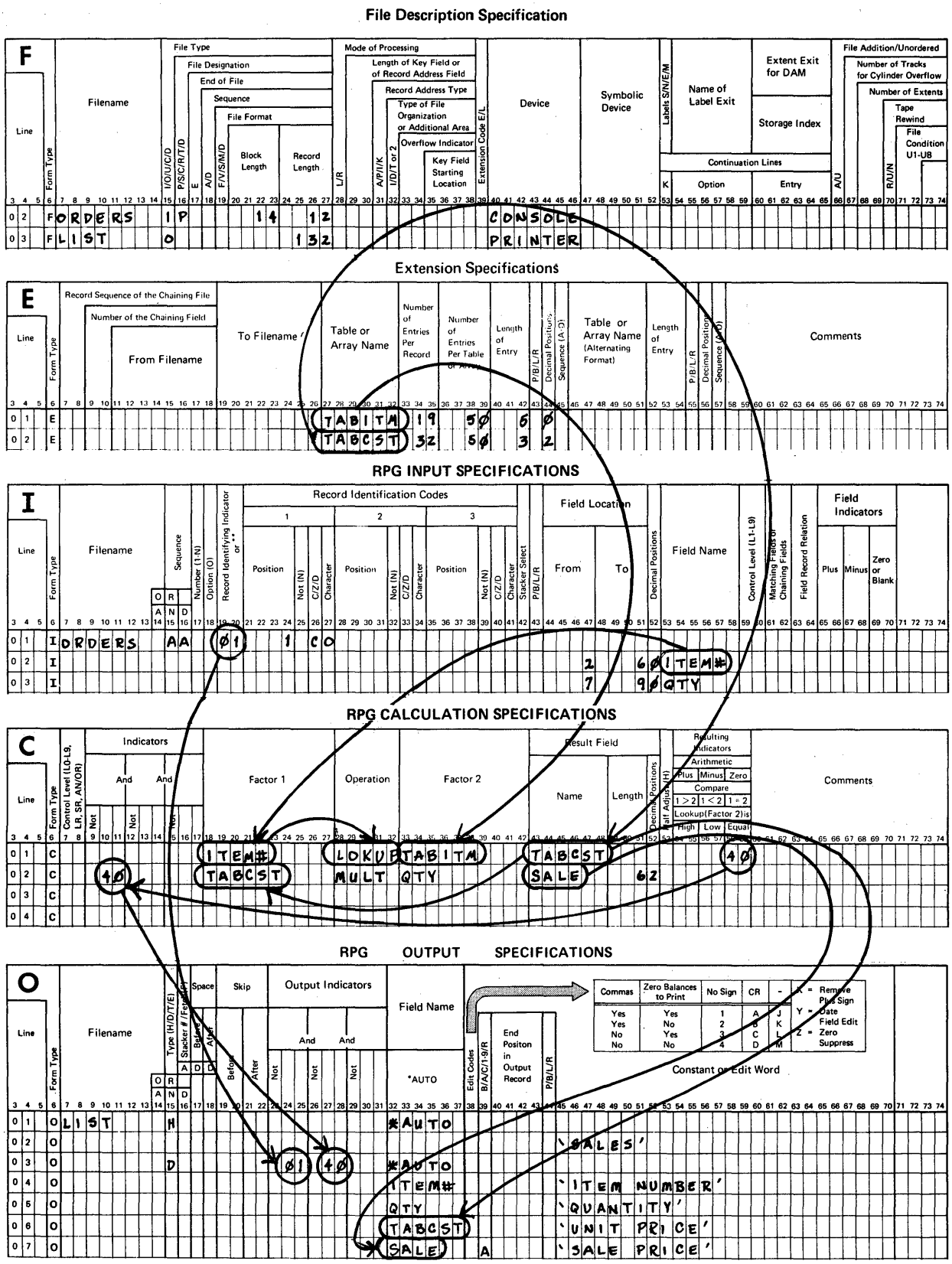
REPORT

SALES

ITEM NUMBER	QUANTITY	UNIT PRICE	SALE PRICE
22615	100	7.79	779.00
33675	50	.25	12.50
12354	740	2.50	1,850.00

When table data is used to solve a problem, we need to describe the table or tables being used and describe any calculations and/or output associated with this data. Tables are described on an Extension specification sheet. A table lookup operation (LOKUP) is used on the Calculation sheet to search one table in order to find a corresponding table value in the other. After a desired value is found, it may be used for additional calculations and/or output.

Coded Solution



Explanation of the Solution

1. File Description - Two files are described. Orders are keyed and a report is printed.

2. Extension - Two tables (TABITM and TABCST) are described. Table names always start with TAB in RPG II. Included in the description of tables is the "Length of Entry" and the "Number of Entries Per Table".

   For numeric table entries, include the number of "Decimal Positions".

   NOTE: We will discuss other specifications used on the Extension Specification sheet later on in this chapter.

3. Input - The keyed record and its fields (ITEM# and QTY) are described.

4. Calculation - Use an input field (ITEM#) as Factor 1, specify the table lookup operation (LOKUP), the table to be searched (TABITM) as Factor 2, the table containing corresponding unit prices (TABCST) as the Result Field, and assign an unused indicator for the desired search condition.

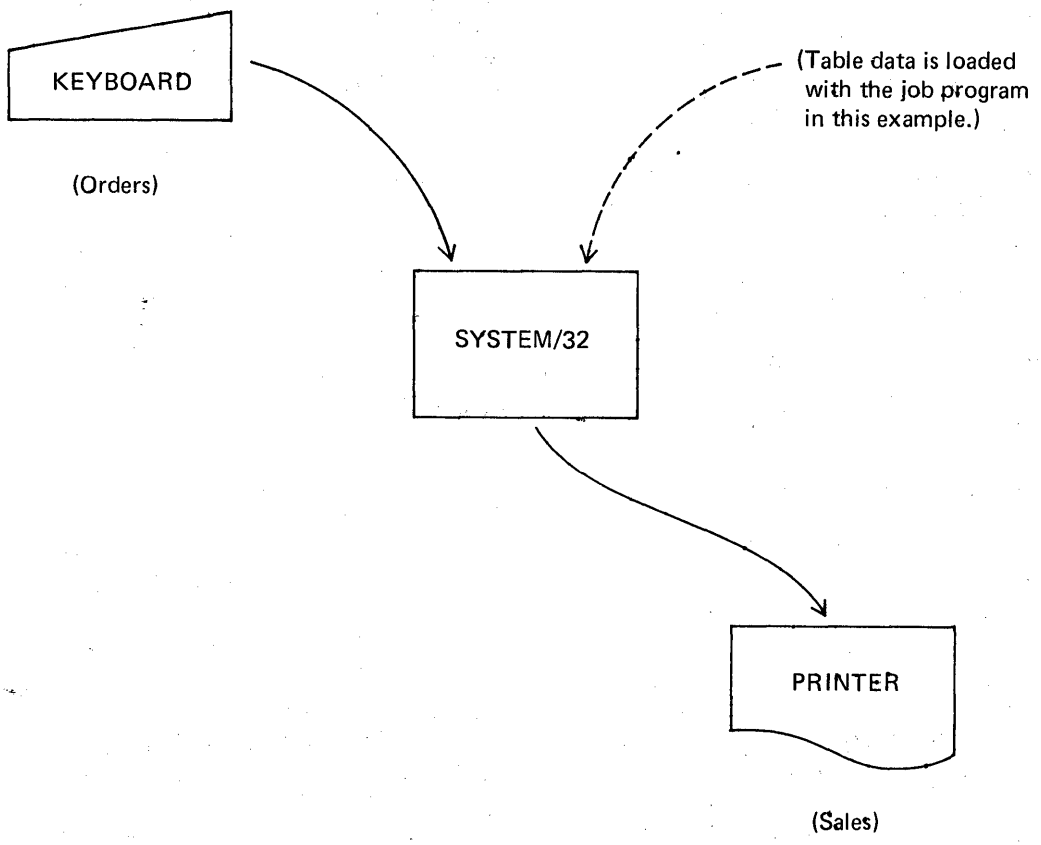
   In our example we are searching for an "Equal" condition, so indicator 40 was assigned in positions 58, 59 (see the headings for lookup above positions 54-59).

   The second calculation (multiply unit price by number of items) is to be done when a desired table entry is found, so it is controlled by indicator 40. The cost value in the corresponding table (TABCST) is multiplied by the quantity (QTY) to determine the sales price (SALE) of the item we keyed as input (ITEM#).

5. Output - Four fields are to be printed when the unit price is found for the keyed item.

   a. The indicators used to control the printing of the detail line are 01 and 40 to be sure that a line is printed when information about the keyed record (ITEM#) is found in the table (TABCST).

Problem System Flowchart



There are data processing problems in which it is convenient to store information in tables so that a desirable piece of data such as a shipping cost can be found and applied during calculations. System/32 RPG II has the capability of using tables of data. Here are some instances where it might be used.

1. When an item number is known, look up its description.

2. When a length is known in inches, look up its equivalent in centimeters.

3. When a temperature is given in degrees Fahrenheit, find its equivalent in degrees Centigrade.

An array is a table of data that is used in special ways in addition to being used as a table.

CHAPTER OBJECTIVES

When you've finished studying this chapter, you should be able to describe tables and arrays, use them in calculations and state the differences that exist between tables and arrays as used in the System/32 RPG II language. This includes:

1. describing tables and arrays as to size, type of content and name using the RPG II Extension sheet,

2. specifying the look-up operation on the Calculation sheet, and

3. printing out the contents of an active table or array at the end of a job.

To describe tables and arrays you will be making use of an Extension specification sheet. So, in addition to this text, you will need a few sheets of each of the following forms.

GX21-9091	Extension and Line Counter
GX21-9092	Control and File Description
GX21-9094	Input
GX21-9093	Calculation
GX21-9090	Output

You also need your System/32 RPG II Language reference manual SC21-7595.

A self test is included in this chapter.

Tables

A table is an arrangement of data items having like characteristics. Each data item in a table is known in RPG II as a table entry or element.

RULES:

1. Every table entry must be the same length.

2. Every table entry must be of the same type, either all numeric or all alphameric.

3. Every entry in a numeric table must have the same number of decimal positions.

Each table used in a program is defined on the Extension specifications sheet. Get a blank Extension sheet (E in column 6) so we can define a few sample tables.

1. Specify a table name starting in position 27.

RULE: Every table name must start with TAB and may include additional letters or numbers in any combination.

Which of the following are acceptable table names?

TAB123	TABX	TAB/B
TA55	ABCDEF	TAB 29
TAB		

* * *

Acceptable table names are: TAB123, TABX, and TAB. Why are the others <u>not</u> acceptable?

TAB/B has a special character. TA55 does not follow the rule, "start with TAB". ABCDEF is wrong for the same reason. TAB 29 has a blank which is not a letter or a number.

2. Specify the "Length of Entry" in positions 40-42. If the entry is a numeric field, include the number of decimal positions in position 44.

3. Specify the "Number of Entries per Table or Array" in positions 36-39.

4. Specify the "Number of Entries per Record" in positions 33-35. For example, if each record containing table data was 100 positions long and each entry in that table was 5 positions long, there would be 20 "entries per record".

Use your Extension sheet to define the following tables.

Name	Type	Length of Entry	# of Entries Per Table or Array	# of Entries Per Record
TABXYZ	numeric	3	16	16
TAB2	alphameric	20	25	4
TABC	numeric	7 with 2 decimals	100	14
TAB19	numeric	15 with 9 decimals	10	6
		*	*	*

Your entries should look like this.

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			TABXYZ	16	16	3		0							
02	E															
03	E			TAB2	4	25	20									
04	E															
05	E			TABC	14	100	7		2							
06	E															
07	E			TAB19	6	10	15		9							
08	E															
	E															
	E															

Now describe a table that contains the names of the months of the year. Assume that all table entries will fit into one record. Make up your own table name.

* * *

If your coding is correct, you have TAB in positions 27-29. Positions 30-32 may be blank or have any letters or numbers (without a blank between them). Positions 34, 35 should contain a 12 and positions 38-39 should contain 12. The length of a table entry is always large enough to hold the longest word or phrase or the largest number to be stored in the table. Since the longest name is SEPTEMBER, your entry for positions 40-42 should be 9 in position 42.

The tables we have been describing up to this point are known as "compile-time tables". By this, I mean that the actual table data is included as a part of your RPG II source program. Suppose that you wrote a program that uses 2 compile-time tables. This is the order in which your coding would be keyed in or read in from disk for compilation.

1. File Description specifications
2. Extension specifications
3. Input specifications
4. Calculation specifications
5. Output specifications
6. a special record containing **b in 1-3 (b means blank)
7. all of the actual table data records for the first table as defined on the Extension sheet
8. a second special record (**b in 1-3)
9. actual table data records for the second table defined on the Extension sheet

During the process of compilation, these tables are stored <u>with</u> the program for future use.

What is one purpose of the Extension sheet? When are **b records a part of the source program? The Extension sheet is used to describe every table used in a program. The **b records are included in a source program whenever compile-time tables are required.

Well, how is it going? OK so far? Let's move on to a second kind of table known as a "pre-execution-time table".

Pre-execution-time tables are created some time before they are needed and they are stored on the disk as a file. A table of this type is also defined in the same manner on the Extension sheet, but we must include one more specification. We enter a filename in positions 11-18 known as the "From Filename". When this is the case, a compiled object program is loaded to run a job followed immediately by all pre-execution-time tables defined as files for it, and then the job starts to run.

RULE: When pre-execution-time tables are used in a program, each one must be described as an input table file on the File Description sheet and must include these entries.

1. a T in position 16 (code for a table file)

2. an E in position 39 (code for extension entries needed) to alert the RPG II program compiler to tie together the File Description entry and the Extension entry having the same filename, when used in the object program.

Note: This rule only applies to pre-execution-time tables.

Describe a pre-execution-time table file having the following characteristics on a File Description sheet and an Extension sheet.

1. The input table file has a record length of 54 characters and is stored on disk.

2. The table entries are numeric, six positions long with no decimals. There are 9 entries in each record and a total of 100 entries in the table.

Make up your own filename and table name.

* * *

Check your coding against these entries. If you do not know why yours does not agree, re-read this section about "Pre-execution-time" tables.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
02	F	INTABL	I	T					54							E	DISK													
03	F																													

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			INTABL		TABMOT	9	100	6		Ø							
02	E																	
03	E																	

Now that you have learned to describe either "compile-time" or "pre-execution-time" tables, let's consider how they are used in a program.

In general, table look-up is this kind of activity.

1. A "search" field value is compared against the first table entry.

2. If the table entry is what is being searched for, the table value is made available for processing and output.

3. If the first table entry is not what is being searched for, the program compares the search value against the next table value. Again, the question (in 2 above) is raised.

4. If no acceptable table entry is found during the search and the entire table has been searched, the next step following the look-up operation is begun.

In order to describe the table look-up activity, you must:

1. describe the tables to be used by making entries on the Extension sheet (and on the File Description sheet if needed),

2. describe an input field (or a constant) that contains (or is) the searching value,

3. describe the LOKUP (table look-up) operation including:

   a. the "search" field or constant as Factor 1,

   b. LOKUP as the operation,

   c. the table to be searched as Factor 2,

   d. another table which contains corresponding values as the Result Field (optional),

   e. at least 1 Resulting Indicator (in positions 54-59).

Here is an example of table look-up as coded on a Calculation sheet.

RPG CALCULATION SPECIFICATIONS

C Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators: Not (9-11)	Indicators: And, Not (12-14)	Indicators: And, Not (15-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field: Name (43-48)	Result Field: Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus / Compare 1>2 / Lookup(Factor 2) is High (54-55)	Resulting Indicators: Arithmetic Minus / Compare 1<2 / Lookup(Factor 2) is Low (56-57)	Resulting Indicators: Arithmetic Zero / Compare 1=2 / Lookup(Factor 2) is Equal (58-59)	Comments (60-74)
01	C					CODE	LOKUP	TABAAA	TABBBB						12	
02	C															
03	C															
04	C															

This description means, "search the table named TABAAA for a value equal to (indicator 12 is in positions 58-59) the value found in the search field named CODE. When you find an equal, also locate a corresponding value in the table named TABBBB".

Try to code these examples.

1. Search a table named TABRAT for a value equal to the value in the search field named RATE. When an equal is found, also locate a corresponding value in a table named TABAMT.

2. Search a table named TABTAX for a value equal to the value 175. When an equal is found, also locate a corresponding value in TABSTX.

* * *

You needed to assign an indicator in positions 58-59 to test the look-up operation for an equal condition in both examples. Here are my solutions. Your indicator entry may be different, but the other entries must be the same as mine.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators (Not, And Not, And Not)	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (1>2) (Plus)	Low (1<2) (Minus)	Equal (1=2) (Zero)	Comments
01	C			RATE	LOKUP	TABRAT	TABAMT						32	
02	C													
03	C			175	LOKUP	TABTAX	TABSTX						15	
04	C													
05	C													

Normally, the reason for doing a table look-up operation is to use the value found in the corresponding table (the one in the Result Field) for additional calculations. If no search table entry (the Factor 2 table) satisfies the problem requirement, such as "search for an equal value", we may wish to bypass all further calculations, or even stop the computer run. One reason for stopping is that if table data or the search field data or the constant is wrong, the operator should probably re-run the job after corrections have been made. Otherwise, if the job continues, all output will be garbage.

How would you bypass further calculations? How would you, as a programmer, stop the computer during a run?

You already know how to bypass calculation steps - simply include a GOTO operation. We haven't yet talked about how to stop the computer during a run. It is easy for system operators, they just press a stop key on the machine. The programmer uses a special "halt" indicator to do the same thing. Here is an example:

RPG CALCULATION SPECIFICATIONS

C				Indicators							Factor 1	Operation	Factor 2	Result Field			Resulting Indicators				Comments
Line		Form Type	Control Level (L0-L9, LR, SR, AN/OR)	And		Not	And		Not	And				Name	Length	Decimal Positions	Half Adjust (H)	Arithmetic: Plus	Minus	Zero	
				Not			Not			Not								Compare: 1>2	1<2	1=2	
																		Lookup(Factor 2)is: High	Low	Equal	
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	1	C									FIND	LOKUP	TAB1	TABX						39	
0	2	C		N39								SETON						H1			
0	3	C		N39								GOTO	END								
0	4	C																			
0	5	C																			
0	6	C																			
0	7	C																			
0	8	C																			
0	9	C									END	TAG									
1	0	C																			
1	1	C																			

Consider what happens in this example when the entire table (TAB1) has been searched and no value equal to the value in FIND can be located. First, indicator 39 is not turned on. Second, we tell the computer to "set on" a special halt indicator, H1. Third, we also direct the program to bypass all further calculations.

RULE: When a halt indicator is turned on, processing continues for all remaining steps including output for that record, and then the computer stops.

When did I include the GOTO instruction? Well then, what good is the H1 indicator when it turns on?

Using the GOTO assures me that all further processing will be bypassed, but the machine does not stop running! By including the SETON instruction and any one of the halt indicators (H1-H9), the machine will stop after all processing and output for that record has been completed. The operator then becomes aware of the fact that something has gone wrong in your program. You of course must provide a set of instructions to the operator for this situation so the job may either be completed with the error noted, or the job removed until the error can be corrected.

One more point. After the operator takes action including restarting the machine, the halt indicator is automatically turned off.

RULE: The SETON operation "sets on" any and all indicators specified in positions 54-59.

We have looked at examples in which the search took place for an "equal" condition. Look at your Calculation sheet at the titles over positions 54-59. For which conditions can table look-up be tested? Which entry is the controlling specification for a look-up operation?

* * *

The look-up activity may be tested for 3 conditions: High, Low and Equal. The controlling specification for this operation is the table named as Factor 2. Let's examine the situation where it is useful to have an indicator assigned to High (in 54-55) or in Low (56-57). Suppose we were looking for this condition.

> Find a value in a table that is the first entry that is higher than the search field value.

Obviously the entries in the table need to be in some sort of sequence or we could not find a "higher" entry. Also, the entries do not include every possible value. For example, let's say we have a numeric table with these values.

TABCUB

001
008
027
064
125
216

Now, when the look-up is done and we are searching for a "high" condition, the program compares the value in the search field (assume it is 025) against the first table value. Is Factor 2 (the first table value) high? No, because the first entry in the table is only 001. Now the program compares 025 against 008, the second table value. Is the table entry high? No, so it looks at the next table value. Is the table value high? Yes, 027 is high as compared to 025. What happens now?

The table look-up operation is complete and the indicator assigned in positions 54-55 for a high condition turns on. What will happen the next time if the new search field value is 220?

* * *

In this case, the indicator is not turned on because the table value is never higher than the search value. Will the indicator turn on if the search value is 216? No, because the table value is not higher.

Examine the following coding and then select a correct answer from the choices shown.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (LO,L9, LR, SR, AN/OR)	Indicators: Not	And	Not	And	Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: 1>2	1<2	1=2	Lookup(Factor 2)is: High	Low	Equal	Comments
01	C							SEARCH	COMP 216												17		17	
02	C		17						GOTO PASS															
03	C							SEARCH	LOKUPTABCUB												48			
04	C																							
05	C																							
06	C																							
07	C							PASS	TAG															
08	C																							
09	C																							

1. Indicator 17 will never be turned on.

2. If indicator 48 is turned on, SEARCH is less than 216.

3. If indicator 17 is turned on, indicator 48 will also be turned on.

* * *

The correct choice is number 2. When examining table data and possible search field values, consider the use of a comparison to bypass the steps that follow. You might also wish to halt processing under certain conditions.

RULE: If a table is to be searched for a "high" condition, table values must be in ascending sequence.

RULE: If a table is to be searched for a "low" condition, table values must be in descending sequence.

Now look at an Extension sheet. An entry is made in position 45 whenever tables are to be searched for either a "high" or a "low" condition. Specify code A for ascending or code D for descending table values.

RULE: At least one indicator must be assigned but no more than two indicators may be entered for a look-up operation.

In RPG II coding, this provides for five acceptable combinations.

1. High (54-55): search an ascending table until a value in it is higher than the search value.

2. Low (56-57): search a descending table until a value in it is lower than the search value.

3. Equal (58-59): search a table until an equal is found. This table may be in any sequence or it can be randomly arranged.

4. High (54-55) and Equal (58-59): search an ascending table until a value in it is either equal (tested first) or a high condition exists.

5. Low (55-56) and Equal (58-59): search a descending table until a value in it is either equal (tested first) or a low condition exists.

RULE: When 2 conditions are to be tested, the same indicator may be assigned to both places.

The RPG II Reference Manual contains additional rules and examples you should read about at this time. Ignore information about arrays as you read because we will present the topic of arrays when you return to this text.

* * *

You read about each point we discussed here plus one significant addition.

> "Related tables can be described separately or in alternating format."

What this means is that instead of describing two tables whose values are related by using two separate lines on an Extension sheet, one line is used to describe a combination table that alternatingly contains a value for one table followed by a value for the other table and so on. When the alternating format is used, fill in entries in positions 27-45 to describe the table values in the first part of each group and then specify a second table name and information using positions 46-57 for table values in the second part. Here is an example.

As Two Related Tables

Extension Specifications

E	Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
	01	E			TAB1	7	3	1			ØA						
	02	E			TAB2	7	7	9									
	03	E															
	04	E															

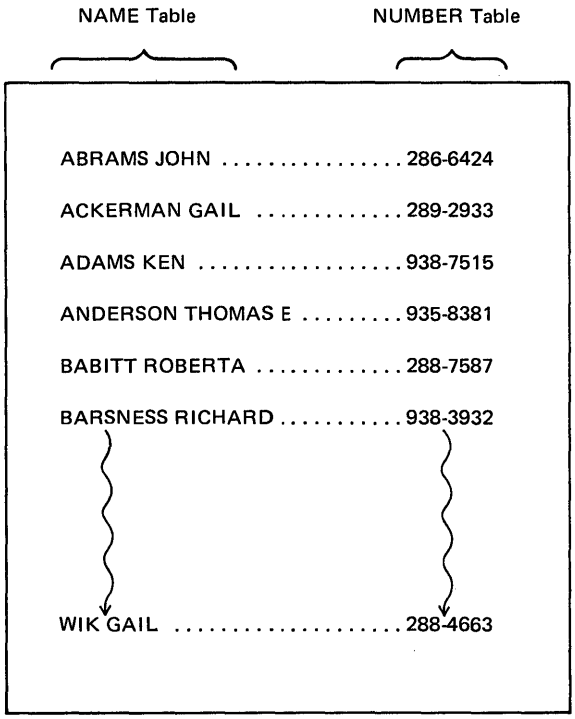
In Alternating Format

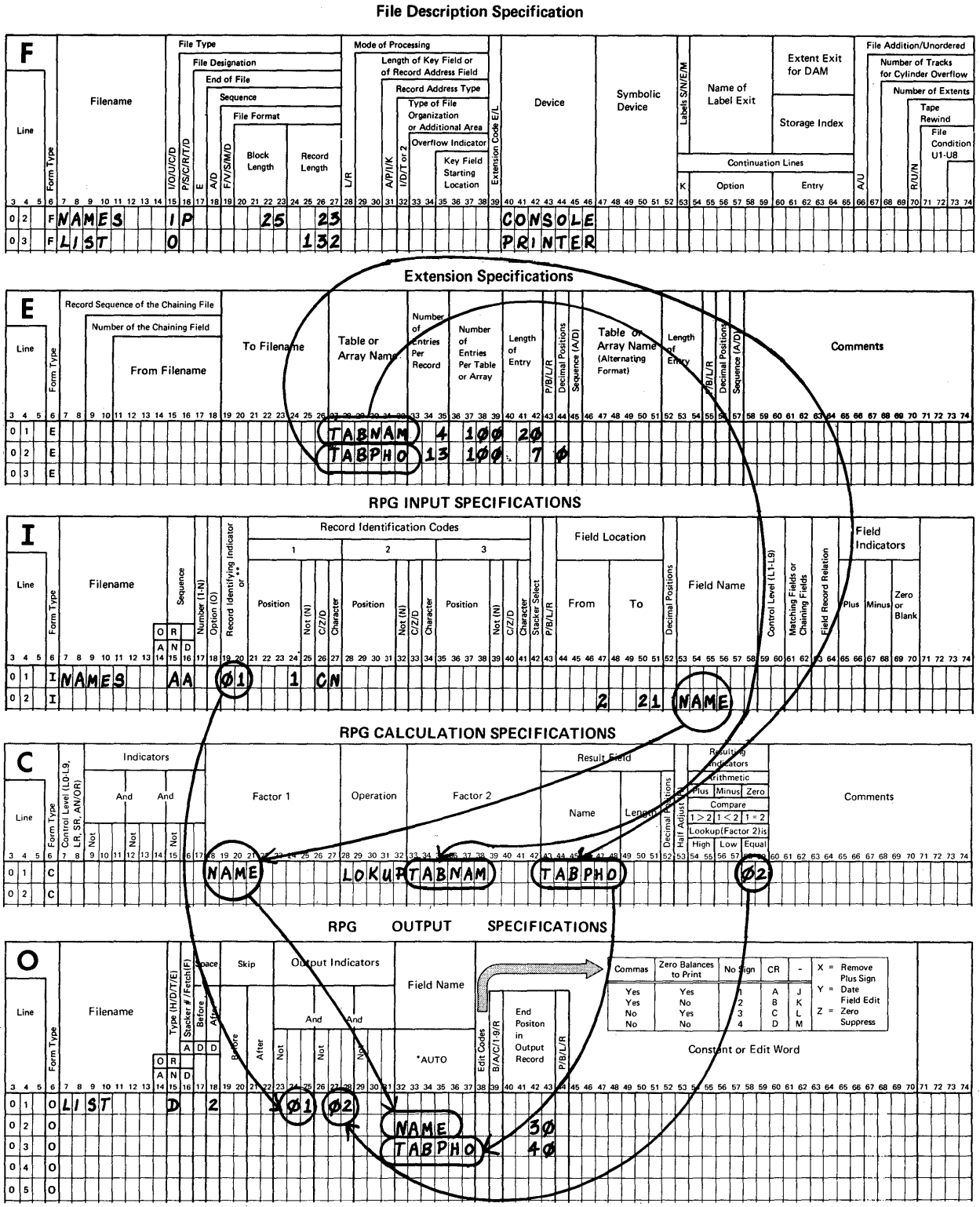
Extension Specifications

E	Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
	01	E															
	02	E															
	03	E															
	04	E															
	05	E															
	06	E			TAB1	7	7	1			ØA	TAB2	9				
	07	E															
	08	E															
		E															
		E															

Since RPG II can use tables described either way, you need to know how your company wishes to arrange table data before creating table data values and then describe them as created in your problem solutions.

To complete the presentation on Tables, I have included a coded solution to an example in which we wish to look up phone numbers and then print a list of them.




File Description Specification
F
Filename
File Type
File Designation
End of File
Sequence
File Format
Block Length
Record Length
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
Extension Code E/L
Device
Symbolic Device
Name of Label Exit
Extent Exit for DAM
Storage Index
Continuation Lines
Option
Entry
File Addition/Unordered
Number of Tracks for Cylinder Overflow
Number of Extents
Tape Rewind
File Condition U1-U8
NAMES IP 25 23 CONSOLE
LIST O 132 PRINTER
Extension Specifications
E
Record Sequence of the Chaining File
Number of the Chaining Field
From Filename
To Filename
Table or Array Name
Number of Entries Per Record
Number of Entries Per Table or Array
Length of Entry
Table or Array Name (Alternating Format)
Comments
TABNAM 4 100 20
TABPHO 13 100 7 0
RPG INPUT SPECIFICATIONS
I
Filename
Sequence
Number (1-N)
Option (O)
Record Identifying Indicator or **
Record Identification Codes
Position
Not (N)
C/Z/D
Character
Stacker Select
P/B/L/R
Field Location
From
To
Decimal Positions
Field Name
Control Level (L1-L9)
Matching Fields or Chaining Fields
Field Record Relation
Field Indicators
Plus
Minus
Zero or Blank
NAMES AA 01 1 CN
2 21 NAME
RPG CALCULATION SPECIFICATIONS
C
Control Level (L0-L9, LR, SR, AN/OR)
Indicators
And
Factor 1
Operation
Factor 2
Result Field
Name
Length
Half Adjust
Resulting Indicators
Arithmetic
Plus
Minus
Zero
Compare
1>2
1<2
1=2
Lookup(Factor 2)is
High
Low
Equal
Comments
NAME LOKUP TABNAM TABPHO 02
RPG OUTPUT SPECIFICATIONS
O
Type (H/D/T/E)
Stacker # /Fetch(F)
Space
Before
After
Skip
Output Indicators
Field Name
*AUTO
Edit Codes
End Positon in Output Record
Commas
Zero Balances to Print
No Sign
CR
-
X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress
Constant or Edit Word
LIST D 2 01 02
NAME 30
TABPHO 40

ARRAYS

Now for arrays. Many of the points just covered for tables are true for arrays. Here is a list for quick review.

1. An array is a table. That is, an array is an arrangement of data items having like characteristics.

2. Each array used in a program is described on the Extension specifications sheet.

3. There are "compile-time" arrays and "pre-execution-time" arrays. Both are used and specified in the same manner as are tables of these same two types.

4. A look-up operation (LOKUP) may be specified for calculations. There are five acceptable combinations for testing the results of an array look-up: High, High and Equal, Low, Low and Equal, or Equal only.

5. Related arrays can be described separately or in alternating format.

That was easy. You might be raising the question, "Why do we need to use arrays if they are just like tables? Why not just use tables?". Well, there are differences we want to bring out in this lesson so that you can determine in given data processing problem situations whether you will want to use tables or arrays. Here are some of those differences.

1. Only table names may begin with TAB. Array names can be any other field name you wish to make up.

RULE: An array name refers to the entire array.

2. There is a special kind of array called an "execution-time" array. In addition to describing this kind of array on an Extension sheet, you also describe it as an input record on an Input sheet. For example:

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E					ARAY		8	12		4							
02	E																	
03	E																	
04	E																	
05	E																	
06	E																	
07	E																	
08	E																	
	E																	
	E																	

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence (OR AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	XYZFILE	AA			01																								
02	I																				1	96		ARAY						
03	I																													

When this kind of array is described on the Extension sheet, no entry is made in positions 33-35 (Number of Entries Per Record) because the input field or fields will provide the data later on. What we described above is essentially this. First, we describe a storage space (Extension specification) for an 8 entry array in which each entry will have 12 positions of numeric data including 4 decimal positions. Second, we will fill the storage space of 96 positions (8 X 12 = 96) with data as it is read in from an input record during the job.

3. In addition to the use of LOKUP to perform a look-up operation on either table or array data, we have the use of 2 special operations to process array data.

a. XFOOT (crossfoot) causes the value in each entry of a numeric array to be added together and stored in the Result Field.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus / Minus / Zero; Compare 1>2 / 1<2 / 1=2; Lookup(Factor 2)is High / Low / Equal	Comments
01	C				XFOOTARAY		SUM	144				
02	C											
03	C											
04	C											

b. MOVEA (move array) causes characters from the left-most positions of Factor 2 to be moved to the left-most positions of the Result Field. Using this operation enables you to do one of three things:

1. move data in one array to replace data in another array, or
2. move part or all of the data in an array into a field, or
3. move part or all of the data in a field into an array.

RULE: Movement of data stops when the end of the shorter length item (the array or the field) is reached.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus / Minus / Zero; Compare 1>2 / 1<2 / 1=2; Lookup(Factor 2)is High / Low / Equal	Comments
01	C				MOVEAARR1		ARRA2					
02	C											
03	C				MOVEAARA65		FIELDX	4				
04	C											
05	C				MOVEAFIELD7		AR29					
06	C											
07	C											
08	C											
09	C											
10	C											

4. Although we didn't discuss editing of tables on the Output sheet, a table entry is edited just as a numeric field is edited.

   An array entry may be edited in the same manner. Specify an edit code in position 38 on Output and the array entry will be edited as desired.

5. An array may be edited in its entirety! If you enter an edit code in 38 and name the array, RPG II will cause the array to be edited so that each entry in the array will include the desired punctuation and 2 spaces will be included to separate each entry from the other when printed.

RULE: You must be sure to plan for enough space on the print line to include all punctuation characters as well as the 2 spaces between items.

Example:

ARRAYZ has 6 entries of 7 positions each with 2 decimals. Let's say we will specify edit code 1 to include commas and a decimal point, but no sign. How much space is needed to print the entire array?

Well, each entry will include 7 positions of data plus a comma and a decimal point plus 2 spaces for a total of 11 print positions. It will look like this on your plan:

ØØXX,XXX.XX

Since there are six of these entries in the array, we need 66 print positions on a line in order to print the array.

NOTE: You may use an edit word instead of an edit code to edit an entire array. In this case, your edit word must include the two blank spaces to separate results.

6. As you recall, we describe entire tables and arrays on an Extension sheet. Thereafter, we can search either a table or an array to find a particular item using the LOKUP operation.

   RPG II provides a unique facility for the referencing and use of particular array entries through the use of a scheme known as "indexing". In this case we:

   a. use an array name of 1, 2 or 3 characters, and

   b. include an "index" constant or field to refer to the desired entry.

Example:

Extension Specifications

E			Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
Line		Form Type																	
01		E					SL		4	6		2							
02		E																	
03		E																	

RPG CALCULATION SPECIFICATIONS

C				Indicators			Factor 1	Operation	Factor 2	Result Field				Resulting Indicators				Comments
Line		Form Type	Control Level (L0-L9, LR, SR, AN/OR)	And	And					Name	Length	Decimal Positions	Half Adjust (H)	Arithmetic: Plus	Minus	Zero		
				Not	Not	Not								Compare: 1>2	1<2	1=2		
														Lookup(Factor 2)is: High	Low	Equal		
01		C					FIELDA	ADD	SL,1	TA		72						
02		C																
03		C					XM	SUB	SL,4	REM		62						
04		C																
05		C																
06		C																
07		C																
08		C																
09		C																

What is the name of the array? How big is each entry? How many entries are there in the array?

* * *

The array named SL contains 4 numeric entries, each being 6 positions long with 2 decimals.

Notice that the first calculation step describes the additions of a field (FIELDA) to "something" (SL,1) to get a result (TA). Also, the second calculation step describes the subtraction of "something" (SL,4) from a field (XM) to get an answer called REM. In each case, the "something" included the name of the array followed by a comma and a constant number. Here is what each step means.

SL,1 refers to the use of data in the first entry in array SL for the addition step. What do you think SL,4 means?

It means, refer to the use of data in the fourth entry in array SL for the subtraction step.

What do these steps each mean?

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup (Factor 2) is (High / Low / Equal)	Comments
01	C			SL,1	MULT 6		X					
02	C											
03	C			SL,3	LOKUPTAB5		TAB6				29	
04	C											
05	C				MOVEAAR		SL,2					
06	C											

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E) (O / A / R / N)	Stacker # / Fetch (F)	Space: Before (A) / After (D)	Skip: Before / After	Output Indicators: And / Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
01	O							SL,1	1		48	
02	O											
03	O											
04	O											
05	O											
06	O											

*AUTO

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

Constant or Edit Word

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

* * *

Remember, the use of an index value and an array name references a particular entry in that array and treats it as though it were a field of data. How did you interpret each example?

1. Multiply the value in the first entry (SL,1) of array SL by 6 to get X.

2. Search the table named TAB5 for a value equal to the value in the third entry (SL,3) of array SL. If an equal is found, locate the corresponding value in TAB6.

3. Move the characters in array AR in the left-most positions of the second entry (SL,2) of array SL. Stop the movement of data when either the second entry of SL is filled or when all characters in array AR have been moved.

4. When producing output, include the first entry (SL,1) of array SL as a field using the edit code 1 to include commas. The right-most character shall be in output position 48.

We are not limited to using a constant as an index value. The following example shows the use of a field as the index. Take time to examine what is happening in this series of calculation steps.

Extension Specifications

E			Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
Line		Form Type																	
0 1		E					AR			5		1							
0 2		E																	
0 3		E																	

RPG CALCULATION SPECIFICATIONS

C				Indicators						Factor 1	Operation	Factor 2	Result Field			Resulting Indicators			Comments
				And	Not	And	Not	And	Not				Name	Length	Decimal Positions	Arithmetic			
Line		Form Type	Control Level (L0-L9, LR, SR, AN/OR)													Plus	Minus	Zero	
																Compare			
																1>2	1<2	1=2	
																Lookup(Factor 2)is			
															Half Adjust (H)	High	Low	Equal	
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 1		C									SETOF					20	30		
0 2		C									Z-ADD	1	Y	10					
0 3		C								RET	TAG								
0 4		C								AR,Y	COMP	7						20	
0 5		C		20							GOTO	END							
0 6		C								5	SUB	Y	TEST	10				30	
0 7		C		30							GOTO	END							
0 8		C								Y	ADD	1	Y						
0 9		C									GOTO	RET							
1 0		C								END	TAG								
1 1		C																	
1 2		C																	

Here is what I hoped you found out when you examined the coding.

1. The array AR has 5 entries each containing 1 numeric character.

2. On the fourth calculation step, we refer to a particular entry of array AR. The particular entry is determined by the value in the index field Y.

3. The first time through, Y is given the value 1 as a result of the Z-ADD operation. This means that we will use the first entry of array AR on the first try for the COMP step.

4. The value of Y is incremented (increased) by 1 each time the loop is completed. When either the value in an entry of the array is 7 or the value of the index is 5, this activity is finished.

What you have just examined is the coding used to scan the contents of a field, one position at a time. In order to do this, the field was described as a 5-entry array and then a loop was programmed to examine the first, the second, the third, the fourth and the fifth entry for a particular character. In our example the scan was being made for the presence of the character 7.

Tables and arrays are alike by definition, but arrays can be processed in a variety of ways not available for processing table data.

One more very important point about using tables and arrays.

RULE: Entire tables and arrays can be written out under control of RPG II at the end of a job when the last record indicator (LR) is on.

To specify that you want to write out a table or an array:

1. describe an Output file on the File Description sheet to hold those tables and/or arrays,

2. include the filename as an entry on the Extension sheet in positions 19-26 for every table and array to be written out. Positions 19-26 are titled "To Filename".

3. describe a total-time output record (Type code T in 15) under control of the last record indicator (LR). Include the name of the array or array entries or table to be produced as output in positions 32-37 (Field Name).

Take some time to read about arrays in the RPG II Reference Manual and read about the use of the operations LOKUP, XFOOT and MOVEA. When you are ready, return to this text.

* * *

Chapter 6: Summary

When table data is placed in storage, its contents may be searched so that corresponding information (in a second table) can be extracted and used to help solve a data processing problem.

In System/32, table searching starts with the very first element and continues until the desired element is found, or until all elements have been examined yet none satisfy the search requirement as specified.

Arrays may also be searched. In addition an array element may be identified directly through the use of an "indexing value". Also, an entire array may be moved to another storage area (MOVEA), and the contents of every element in a numeric array may be added together in one operation (XFOOT).

Now that you have completed chapter 6 you should be able to:

1. describe tables and arrays,
2. use them in calculations, and
3. print out their contents at the end of a job.

New coding entries encountered in this chapter include:

FILE DESCRIPTION

16	File Designation (T for table file)
39	Extension Code (E)

EXTENSION

11 - 18	From Filename (pre-execution-time tables)
19 - 26	To Filename (to write out table values at end of job)
27 - 32	Table or Array Name
33 - 35	Number of Entries Per Record
36 - 39	Number of Entries Per Table or Array
40 - 42	Length of Entry
44	Decimal Position
45	Sequence (Ascending)
46 - 51	Table or Array Name (Alternating Format)
52 - 54	Length of Entry

INPUT

No specific new entries, but execution-time arrays are described here as fields.

CALCULATION

18 - 27 Factor 1 (use of array names with an index)

28 - 32 Operation (LOKUP, SETON, XFOOT, MOVEA)

33 - 42 Factor 2 (use of array names with an index)

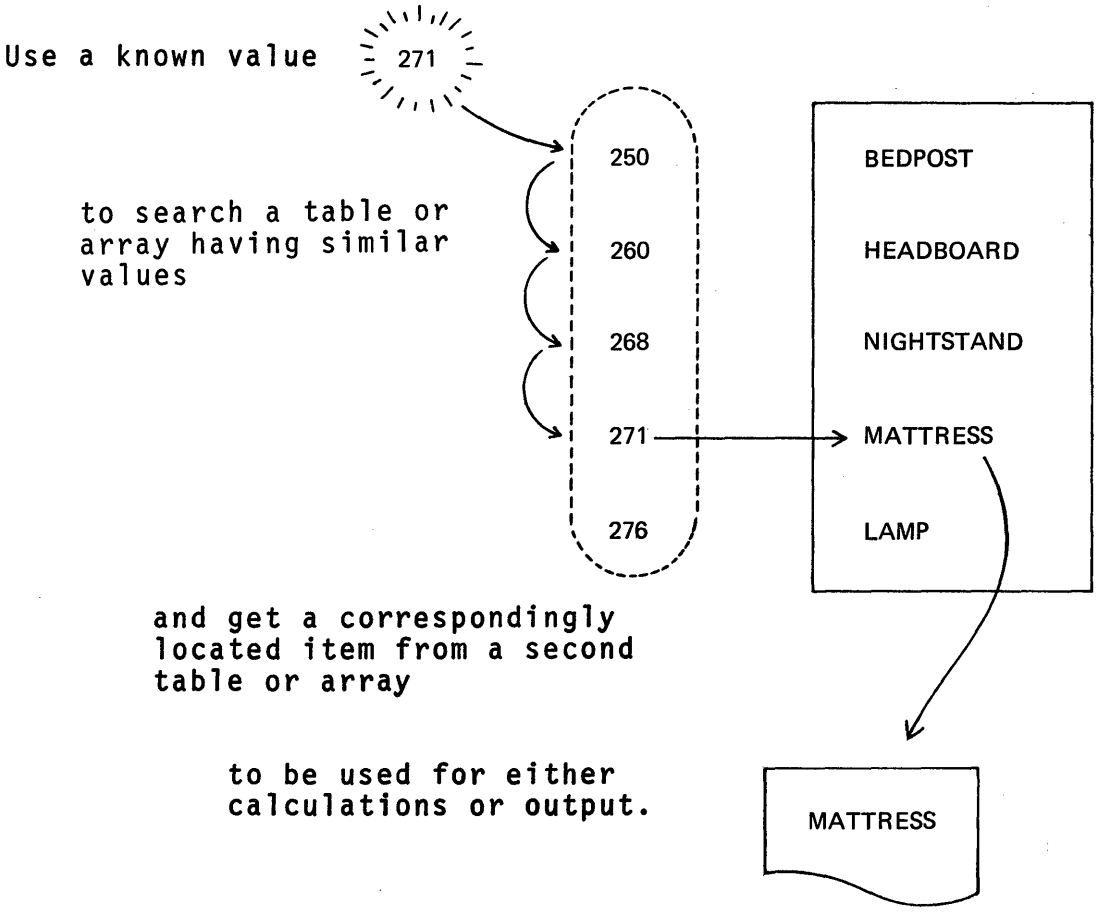
54 - 59 Resulting Indicators (H1 for SETON, all for look-up)

OUTPUT

32 - 37 Field Name (use of array names with an index)

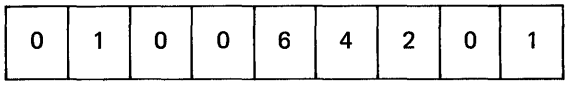
The following illustrations deal with the operations that may be used with table data (LOKUP) or array data (LOKUP, XFOOT, MOVEA).

1. Lookup (LOKUP)



2. Crossfoot (XFOOT)

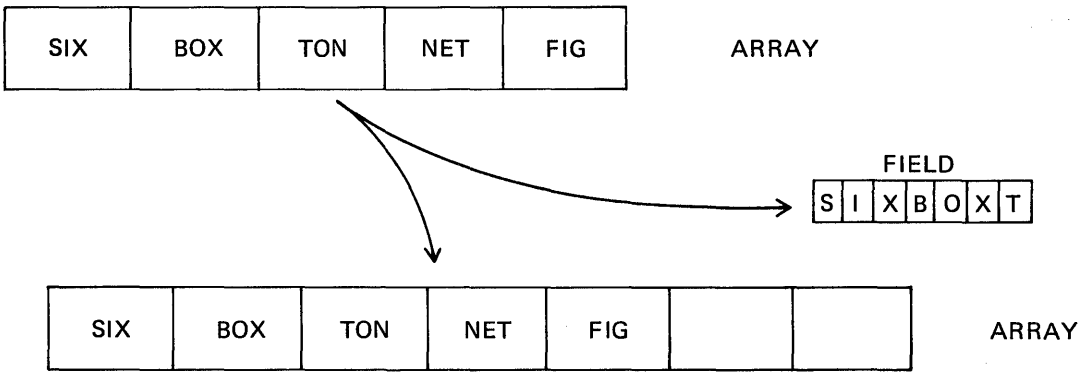
   Add the value in each array element to find the total.



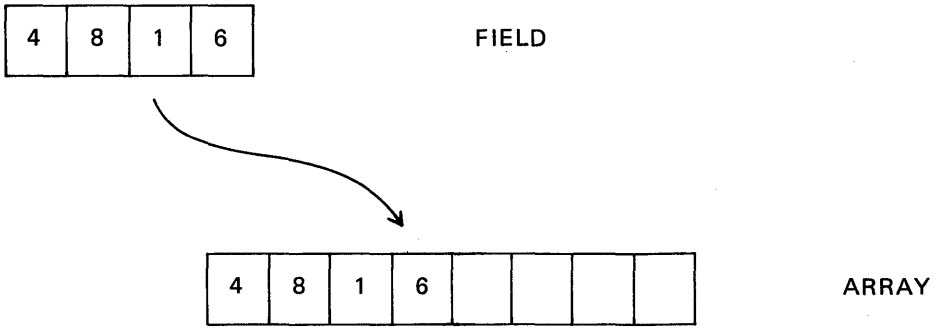
0 + 1 + 0 + 0 + 6 + 4 + 2 + 0 + 1 = 14

3. Move an array (MOVEA)

   a. Move the contents of an array to a field or to another array.



   b. Move the contents of a field to an array.

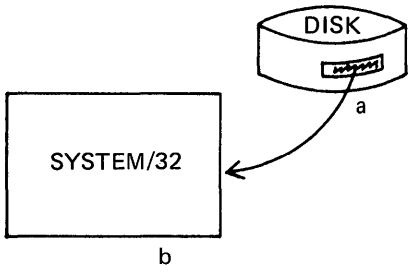


These illustrations relate types of tables or arrays.

1. Compile-time Tables or Arrays

   a. Load the program that <u>includes</u> the table or array, then

   b. start the job.



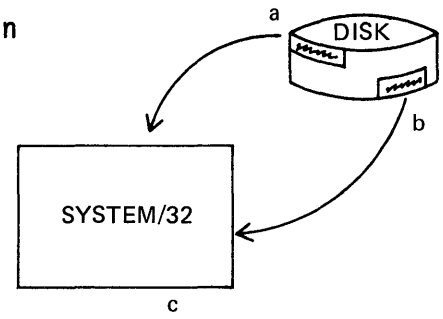
---

2. Pre-execution-time Tables or Arrays

   a. Load the program that

   b. loads the table or array, then

   c. start the job.



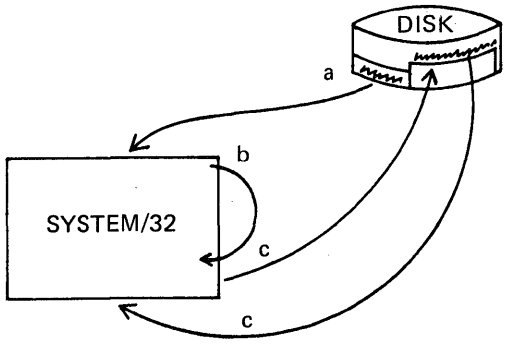
---

3. Execution-time Arrays

   a. Load the program to

   b. start the job, that

   c. reads array data as input records.



SELF TEST: Chapter 6

Try to answer each question from memory. Use a piece of scratch paper to keep track of your answers so you can check them.

1. What is the rule for naming tables?

2. What is the rule for naming arrays?

3. What does the operation LOKUP accomplish?

4. Which 5 combinations of entries for Resulting Indicators are acceptable with the LOKUP operation on the Calculation sheet?

5. How do I reference a particular entry of an array?

6. In which case of editing output will RPG II provide separator spaces for array entries?

   A. When an edit code is used.

   b. When an edit word is used.

7. An index for an array may be either a(n) _____ or a(n) _____.

8. What is the difference between "compile-time", "pre-execution-time" and "execution-time" arrays?

ANSWERS: Chapter 6 Self-Test

1. A table name must begin with TAB. It can be from 3 to 6 characters long.

2. An array name must start with a letter. It can be from 1 to 6 characters long. If it is to be used with an index, it can only be 1 to 3 characters long.

   The first 3 characters in an array name cannot be TAB.

3. It initiates a search of table or array entries to find an acceptable "match" based upon the condition(s) tested by Resulting Indicators. When a search is successful, the particular entry of the table or array is stored for use in further calculations or output.

4. High
   High and Equal
   Low
   Low and Equal
   Equal

5. I use an index value to reference a particular array entry.

6. Separator spaces are included when an edit code is used, choice "A".

7. An index for an array may be either a field or a literal.

8. Compile-time arrays are entered with a source program and included with the resulting object program during compilation.

   Pre-execution-time arrays are defined as input files and read in for use just after the program is loaded, but before other input is made available for processing.

   Execution-time arrays are defined as input records. They are read in as records when the program calls for input.

If you wrote essentially what I did, count it as correct. Allowing 5 points for question #4 and 3 points for question #8, you should have scored at least 10 points. If you did not, you may wish to re-read this chapter.

Since your reason for studying this course was to prepare for attendance in an IBM System/32 RPG II Workshop class, you have completed preparation for it.

Chapters 7 and 8 contain information that will be referenced in the workshop.

These two chapters are not prerequisite to attendance in the workshop, but I know it will be of benefit to continue if you have the time.

Chapter 7: Keyboard, KEY and SET (1 to 2 hours)

When disk files are created, input records are keyed in from the console keyboard on the System/32. In the examples you studied up to now, we specified the device name CONSOLE when describing such input records.

Another device name, KEYBORD, is acceptable for a special approach to keying input records. In both cases, the console keyboard and display screen function as one unit. Here's a chart that alerts you to differences about which you will learn as you study this chapter.

CONSOLE	KEYBORD
Input records are keyed at "normal" input time.	Input records are keyed when a KEY operation is encountered during calculations.
When a complete record has been keyed in, it is processed.	One KEY operation is needed to enter each field of data.
Field names as specified on the Input sheet become the prompts.	Prompts are specified as Factor 1 or included as a 2-digit code number with the KEY operation.
Record length is calculated by adding the number specified as an input field "to" entry to the number of fields specified plus 1.	Record length is simply the length of the longest field to be keyed in. It cannot exceed 40 characters.
To designate that all records in the input file have been keyed in, the operator presses the command key (CMD on the keyboard) followed by pressing the special character "/".	To designate that all records in the file have been keyed in, the operator enters whatever number, letter or special character the programmer has directed on the program run sheet.
RPG II completes all "last record" processing and ends the job.	RPG II completes whatever end-of-job operations and output the programmer has specified. The programmer is responsible for turning on the last record indicator (LR) if he wants RPG II to terminate his job.

Now that you have an inkling as to the differences, you need to know that another operation called SET may be used when KEYBORD is specified as the device name. This operation permits the operator to press one, two or three command keys, which the program checks and in turn sets on corresponding command key indicators. Prompts are included with the SET operation in the same manner as described for the KEY operation.

CHAPTER OBJECTIVES

After studying this chapter you should be able to use the operations KEY and SET in solving problems requiring keyed input data during calculations. For these problems you would specify the device name KEYBORD.

A self test is included in this chapter.

Keyboard and KEY

You have already learned to define and use interactive data entry input files. As you recall, the device name for this kind of file is CONSOLE. Also, the record length is computed as one more than the sum of the highest field end position and the number of input fields to be keyed. The block length must be at least two more than the record length.

When you wish to (or need to) key in data during calculations rather than as input fields prior to the first calculations, you can define a primary input file on the File Description sheet and specify the device name as KEYBORD. That spelling is funny looking, but it is correct.

When you have defined a KEYBORD input file, RPG II provides the following:

1. One field of data may be entered whenever a KEY operation is encountered on the Calculation sheet. Numeric fields may be 15 positions long with 0 - 9 decimal positions while in this case alphameric fields may be up to 40 positions.

2. When the KEY operation is encountered, the program waits for the operator to key in one field of data.

3. After that field is entered, the program becomes active and continues until another KEY instruction appears, or the job ends, or a halt occurs.

When the data is keyed, it is automatically shown on the display screen.

OK. Now here is what you have got to code when doing this kind of job.

1. Specify an input, primary, KEYBORD file.

2. Specify one KEY instruction for each field to be keyed into the system.

3. Specify an indicator to identify the type of record being keyed.

4. Turn on the last record indicator when the job is to be ended, that is, when all records to be keyed have been keyed.

RULE: To describe a keyboard input file on the File Description sheet, enter KEYBORD as the device name and specify the length of the longest field to be keyed as its record length.

RULE: To describe the KEY operation on the Calculation sheet, enter KEY in 28-30; enter the name of the field that shall hold the keyed data as the Result Field; specify the field length and number of decimals if this field has not been defined elsewhere in the program.

For convenience to the operator, specify a constant, as Factor 1. This constant will appear on the display screen on the line above the keyed data to prompt the operator.

Here are some examples in which KEY is used.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators	Comments
01	C			' '	KEY		LOC	17				
02	C											
03	C			'RATE'	KEY		PRATE	4	4			
04	C											
05	C			'AMT XX.X'	KEY		AMOUNT	3	1			
06	C											
07	C											
08	C											

What is the size of the field to be keyed in each example? What else will be shown on the display screen in each example? What is the longest constant you can describe in Factor 1?

* * *

The field lengths are: LOC is 17 alphameric positions; PRATE is 4 numeric positions with 4 decimals; AMOUNT is 3 numeric positions with 1 decimal position.

Since the first example has a 1-position blank entry in Factor 1, no prompt will show, only the 17 keyed characters. In the second example, we will show the prompt RATE above the keyed data. The third example will include the prompt AMT XX.X above the keyed data. Factor 1 may contain a constant of up to 8 positions.

Assume that only these three KEY operations shown were included in one program. What record length entry should be specified for the KEYBORD file on the File Description sheet? Why?

* * *

If you remembered the rule, it is 17. The rule is, "record length is equal to the length of the longest field to be keyed".

Next, let us consider the use of keyboard input files from another point of view. The file is described on a File Description sheet and the particular fields to be keyed as input are described on the calculation sheet. No entries are made on the Input sheet for this kind of input file because every field to be keyed in is described on the Calculation sheet! Remember, the RPG II program will pause to let the system operator provide input fields as needed during calculations.

Here is a brief summary of what we have learned about keyboard input files as described in RPG II.

1. An input primary file is described on the File Description sheet. The device name must be KEYBORD.

2. No Input sheet entries are made for this kind of input file.

3. Each field to be keyed must be described on the Calculation sheet as a Result Field, one field per KEY instruction. Since the keyed data will be displayed as it is keyed, a "prompt" should be included as Factor 1. This prompt will also display during the keying activity.

There is more. When we described input records on an input sheet for DISK or CONSOLE files we included the assignment of a Record Identifying Indicator in 19-20. For a KEYBORD file, we make no input sheet entries, so how can a programmer establish a similar indicator relationship to the records keyed in during calculations?

* * *

There are a number of ways to do this. I hope you thought of the SETON operation as one obvious method. Another would be to compare the contents of a keyed field of let's say one position to a code character. If they are equal, turn on one indicator. If not, turn on a different one. You may have thought of other ways and could be correct.

Why are we concerned about establishing an indicator like this? Well, when it is time to produce output records, we can assign the appropriate indicator to control specific lines of output or specific items within such a line.

One more question. How will this programmer-assigned indicator be turned OFF? If you remember this you have a good memory. The programmer must make sure it is turned off. This can be done by:

1. using a SETOF operation, or

2. repeating the compare instructions the next time around.

Now for the last dilemma. The operator will continue to key in data records whenever the program calculations are interrupted by a KEY instruction.

In normal jobs, the special "last record" in the disk file or the special keyed in "last record" in a CONSOLE file causes the Last Record indicator (LR) to be turned on and the job terminates when all processing and output is produced.

How can a similar action be accomplished with KEYBORD files? You guessed it - the programmer has to somehow make LR turn on. Here are some possibilities.

1. Using the COMP operation, check for a special code that you can tell the operator to key in at the end; when it is keyed in, SETON the last record indicator.

2. If you wanted the operator to key in only 25 records, use a counter and keep track of how many were keyed. When you get to 25, set on LR.

Now that we have talked about KEYBORD files and the KEY instruction, examine the following example and mentally review all of the important points to see how the coding satisfies the rules.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	R/U/N	Tape Rewind	File Condition U1-U8
02	F	VALUES	I	P					7						KEYBORD													
03	F	LIST	O						132						PRINTER													
04	F																											
05	F																											
06	F																											

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators And	Indicators And	Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions	Result Field Half Adjust (H)	Resulting Indicators Arithmetic Plus	Minus	Zero	Compare 1>2	1<2	1=2	Lookup(Factor 2)is High	Low	Equal	Comments
01	C				'CODE'	KEY		CODE		1											
02	C				CODE	COMP	'2'											01	01	LR	
03	C		LR			GOTO	STOP														
04	C				'FIRST'	KEY		VAL1		40											
05	C				'SECOND'	KEY		VAL2		72											
06	C				'THIRD'	KEY		VAL3		61											
07	C				VAL1	ADD	VAL2	VALUE		82											
08	C				VALUE	ADD	VAL3	VALUE													
09	C				STOP	TAG															
10	C																				
11	C																				
12	C																				

RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E) O A	R N	D	Stacker # /Fetch(F) A	D	D	Space Before	After	Skip Before	After	Output Indicators And Not	Not	And Not	Not	Field Name *AUTO	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
01	O	LIST			D				2				01								
02	O																VAL1	1		30	
03	O																VAL2	1		40	
04	O																VAL3	1		55	
05	O																VALUE	1		68	
06	O																				
07	O																				
08	O																				

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Constant or Edit Word

You should know just a bit more before we move to the next topic. I told you to include a constant as Factor 1 to provide a "prompt" on the screen. You are to use one of these as Factor 1 for a KEY operation.

1. an alphameric constant of 1-8 characters,

2. a numeric literal of 1-10 digits,

3. a field of 1-40 characters; the contents of that field are the prompt,

4. a table element (one entry), or

5. an array element.

RULE: Prompts cannot exceed 40 characters.

<u>SET</u>

This operation is frequently used in combination with a KEY operation. First, we will consider its use alone and then in combination.

One purpose of the SET operation is to designate which of the 16 available command keys on the system keyboard may be pressed by the system operator during calculations. If the operator presses one of the designated command keys, its associated Command Key Indicator is turned on. These 16 indicators are KA, KB, KC, KD, KE, KF, KG, KH, KI, KJ, KK, KL, KM, KN, KP, KQ. For example, if the instruction

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus (1>2, High)	Minus (1<2, Low)	Zero (1=2, Equal)	Comments
01	C			\ /	SET						KA	KB		
02	C													
03	C													
04	C													

were encountered in a program, the operator may press command key 1 which will cause the indicator KA to turn on. He may press command key 2 instead, in which case command key indicator KB will turn on. There are two more possibilities.

Press command key 1 <u>and</u> press command key 2.

Don't press either command key.

As you suspected, both KA and KB would be turned on in the first case. In the second case, neither would be turned on. How does the operator know what to do? You guessed it, the programmer must provide directions on a program run sheet so that the operator can press the correct key or combination of keys.

Note the blank constant, indicated by quote marks in Factor 1 (18-20). A SET operation in which command key indicators are included must have a Factor 1 entry or a user message number in 31-32. This rule also applies to KEY. More about user messages later.

Here is a more complete example.

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators High (Plus, 1>2)	Resulting Indicators Low (Minus, 1<2)	Resulting Indicators Equal (Zero, 1=2)	Comments
01	C		19	' '	SET						KL	KM	KN	
02	C		KL		GOTO	RT1								
03	C				Z-ADD	5	TALLY							
04	C		KM		GOTO	RT2								
05	C		KN		GOTO	RT3								
06	C				SETON						H1			
07	C													
08	C													
09	C													

How many command keys could the operator press when the SET operation is found? If none were pressed, what would happen? Which command key indicator controls the instruction to make the value of TALLY equal to 5? Which specific command keys are made available during this program segment? Suppose that the command key indicator KM was turned on the first time thru the program and the second time thru indicator 19 was off. Which instructions would be performed the <u>second</u> time through?

* * *

If indicator 19 is ON, the operator could press 1, 2 or 3 command keys. If none were pressed, two things would happen: first, TALLY would be set to a value of 5 and then halt indicator H1 would be turned on. No command key indicator controls the Z-ADD operation, but if KL is turned on, that step will be bypassed. The specific keys that control KL, KM and KN are numbers 12, 13 and 14 respectively so these are the ones that are available during this program segment.

Since the setting of indicator 19 determines whether or not KL, KM and KN may be turned on, any command key indicator turned on the first time around remains on during the second time around.

RULE: Command key indicators are turned on if the operator presses the corresponding command key. These indicators are turned off when a SET instruction that contains the same indicator is used or when the SETOF operation is used.

If no additional programming steps are included for this example, we may have difficulty during test data runs. How could this problem be eliminated? If you suspect this segment of coding to be a problem area during a test data run how could you check it out?

* * *

The SET operation using only Factor 1 coding may be used to display a series of prompt lines. In this case, the program does not pause until another SET with command key indicators or a KEY operation is encountered. Remember that up to 6 lines of data may be displayed and that the KEY operation uses 2 lines: one for a prompt and one for data as it is keyed.

The problem can be eliminated nicely by inserting this step between step 1 and 2.

RPG CALCULATION SPECIFICATIONS

C				Indicators							Factor 1	Operation	Factor 2	Result Field				Resulting Indicators				Comments
Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)		And			And							Name	Length	Decimal Positions	Half Adjust (H)	Arithmetic: Plus	Minus	Zero		
				Not		Not		Not		Not								Compare: 1>2	1<2	1=2		
																		Lookup(Factor 2)is: High	Low	Equal		
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 1	C			N19				SETOF										KL	KM	KN		
0 2	C																					
0 3	C																					
0 4	C																					

For operator convenience, you may specify a constant, a numeric literal, a field, a table element or an array element in Factor 1 of the SET operation. What does this do? It causes the information in Factor 1 to be displayed as a prompt to the operator just as it does when used in the KEY operation.

The SET operation provides for a special function when used in a program that has an interactive data entry (CONSOLE) input file. By specifying a larger than necessary record length entry on the File Description sheet for a CONSOLE file, we reserve a storage space called a "buffer" into which more than one record can be keyed. This permits an operator to key a new record while the computer processes the prior one.

If during processing it is determined that invalid data is in the buffer, you may wish to clear it out and have the operator re-key the correct data. To do this, specify the SET operation, the CONSOLE filename as Factor 2 and the word ERASE in 43-47.

It looks like this.

RPG CALCULATION SPECIFICATIONS

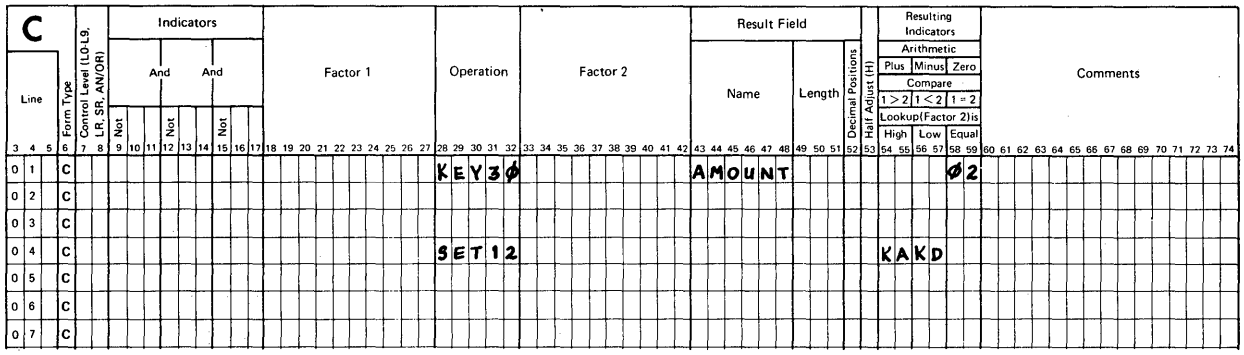
Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
01	C		28		SET	IDEFILE	ERASE					
02	C	*										
03	C	*	IF INDICATOR 28 IS		TURNED ON, AN	INVALID	DATA					
04	C	*	CONDITION EXISTS.		ERASE	THE CONSOLE	BUFFER	AREA.				
05	C											
06	C											

Using KEY and SET

On the IBM System/32 there is a provision for storing messages which can be used in programs to direct the operator to take action just as the prompts do when used in Factor 1.

If your installation has provided for these "user messages", you should become familiar with them so that when it is convenient to use one or more of them as prompts, you will know how to select the ones you need. Each user message is stored along with a 2-digit message identification code for all programmers to use. You may include the use of these messages as prompts by specifying the code digits in positions 31 and 32 along with either the KEY or SET operation. For instance:

RPG CALCULATION SPECIFICATIONS



In the first line, user message 30 becomes the prompt when an amount field is to be keyed into the system. After keying the amount, a test for zero balance is made. If the value of AMOUNT is zero, indicator 02 is turned on.

In the second line user message 12 is the prompt and command keys 1 and 4 may be pressed by the operator. If 1 is pressed, KA is turned on. If 4 is pressed, KD is turned on.

RULE: If Factor 1 contains a prompt and positions 31-32 also include a user message identification code, the prompt in Factor 1 will be displayed and the code in 31-32 will be ignored.

Both the SET and KEY operations interrupt the normal RPG II cycle. When KEY causes the interrupt, the operator is to key in data and then restart the system by pressing the ENTER key on the system keyboard. Likewise, when SET causes the interrupt, he is to press one or more command keys and then restart the system by pressing the ENTER key. When the ENTER key is pressed, calculations continue from the point of interruption.

From this information, you can see that the ENTER key must be pressed following each KEY or SET operation.

RPG II provides a special way of handling a pair of these instructions such that the operator can press one or more command keys and enter data before pressing the ENTER key.

RULES:

1. The SET operation must immediately precede the KEY operation.

2. Both operations must be conditioned by the same indicators, if any, in 7-17. If more than 1 indicator is used, they must be in exactly the same order for each instruction.

3. If a message identification code is specified in 31-32, it must appear in both operations.

Code this example.

You are to make command keys 2 and 5 available. If 2 is pressed, branch to a routine named TWO. If 5 is pressed, branch to routine FIVE. If both are pressed, branch to routine DUBL. Also, you need to have data keyed into a 7-position alphameric field called INFO before branching to any of the three routines. The prompt to the operator for this segment is found in user message number 45.

* * *

Your solution should be like this one. However, the order in which the last three steps appear is not significant. Did you include a SETOF instruction for KB and KE? Why or why not?

* * *

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Not (9)	Indicator (10-11)	Not (12)	Indicator (13-14)	Not (15)	Indicator (16-17)	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / High / 1>2 (54-55)	Minus / Low / 1<2 (56-57)	Zero / Equal / 1=2 (58-59)	Comments
01	C									SET45						KB	KE		
02	C									KEY45		INFO	7						
03	C			KB	N	KE				GOTO	TWO								
04	C		N	KB		KE				GOTO	FIVE								
05	C			KB		KE				GOTO	DUBL								
06	C																		
07	C																		
08	C																		

I did not include a SETOF instruction because the SET instruction does not have any controlling indicators specified in 7-17. Therefore, KB and KE are reset before that operation is done each time. Suppose that I did not include the entry in 12-14 for lines 3 and 4, what would happen?

If only KB <u>or</u> KE were on everything is OK. <u>But</u> if both KB and KE are pressed, the program would branch to routine TWO because KB is on when both are on.

RULE: The first conditioned instruction that is satisfied will be performed. If that instruction is a branch (GOTO), be sure that it is properly specified.

Summary: KEYBORD, KEY and SET

When the KEY operation is used, an input KEYBORD file is described and no input specifications are described for that input file.

The KEY operation causes a running program to pause during the time when calculation steps are being processed. The system operator must respond to this pause by entering a field of data, or by pressing the "ENTER" key on the keyboard without keying a field of data (in this case the field defined to receive keyed data is set to zeros or blanks).

Processing then continues until another pause by either encountering another KEY or SET operation, or until the calculation steps have all been completed.

When command key indicators are specified, the SET operation also causes a running program to pause during calculations. The system operator responds either by selecting 1, 2 or 3 command keys or by pressing the ENTER key on the keyboard. Processing then continues until another pause occurs or until all calculations are completed.

The SET operation may be used to erase the buffer area reserved to hold input data from a CONSOLE file.

When either KEY or SET is specified without a message identification code (in 31, 32) a prompt must be entered as Factor 1 (18-27).

The KEY and SET operations may also be used in pairs. When such a pair is described, the SET operation must be first; both operations must be controlled by the same indicators, if any; and if a message identification code is used in positions 31-32 with an operation, it must be used in both. When a pair of these operations is encountered, the system pauses only once rather than twice.

Chapter 7: Summary

The System/32 is a computer that makes use of keyed data in order to create and update files. In RPG II language we have the facility to describe keyed data entry in two ways: one, input-sheet-specified interactive data entry using the device name CONSOLE and input fields and two, calculation-sheet-specified interactive data entry using the device name KEYBORD and the operations KEY and SET.

Now you should be able to specify the use of either method. To make the prompts meaningful, you need to specify meaningful field names for CONSOLE files and to specify meaningful entries as Factor 1 for KEYBORD files.

New coding entries presented in chapter 7 include:

FILE DESCRIPTION

40 - 46	Device (KEYBORD)

EXTENSION

No new entries.

INPUT

No new entries.

CALCULATION

9 - 17	Indicators (KA-KN, KP, KQ)
28 - 32	Operation (KEY, SET)
33 - 42	Factor 2 (ERASE - use with SET to erase a CONSOLE buffer area)
54 - 59	Resulting Indicators (KA-KN, KP, KQ)

OUTPUT

23 - 31	Output Indicators (KA-KN, KP, KQ)

SELF TEST - Chapter 7

Use a piece of scratch paper to record your answers and check them after you've finished.

1. When KEYBORD is specified as the "Device Name", what operation is used to enter input data?

2. In which operation may you include a "user message identification code" to identify a desired operator prompt?

3. What is the longest field of data that may be entered when KEYBORD is the device name?

4. What must a programmer do in order to cause a job to end when keyboard data is used in a program?

5. How many command key indicators are available for programmer use?

ANSWERS - Chapter 7 Self Test

1. KEY

2. (1 point) KEY or SET may be used

3. 40 characters of alphameric data

4. To end such a job, the programmer must include a test for a special end of job condition. Normally, he will assign LR as the indicator for this condition.

5. 16 (KA-KN, KP, KQ)

RATINGS

5 Great

4 Good

Chapter 8: Indicators, Comments, DEBUG and Subroutines (1 to 2 hours)

All programs generated by RPG II are controlled by the setting of program switches known as "indicators". As an RPG II programmer, it is your responsibility to assign appropriate indicators and use them at various points in order that the processing of all records is properly initiated, controlled and terminated.

The topic of "indicators" is presented here in order to review their definition and usage as well as introduce a number of new ones to you. As you know, an indicator controls "when" activities in the generated program are to take place. You will be examining the use of indicators in relation to the various specifications sheets.

An RPG II source program is a collection of descriptive statements on a variety of coding specifications sheets. Some entries such as Filename (REGISTER) and Field Name (ITEM) or Operation (ADD) are actual words that you and I use in everyday conversation. Other entries such as "I" for input, "132" for record length or "LR" for last record indicator are not. RPG II includes a facility to insert comments throughout a program in order to eliminate guesswork when looking at a program created in the past or by some other programmer.

A special operation named DEBUG is a part of the RPG II language. It is used to help you locate program errors during test data runs so that corrections can be made prior to use of the program with real data. You can specify its use on any line on a Calculation sheet. When encountered, it temporarily stops the data run in order to print a list of all indicators that are ON at that time as well as the value stored in any one field if desired. When this information has been recorded, the test run continues. Any number of DEBUG statements may be included in a program, but they are generally placed at strategic points such as before a GOTO or TAG operation.

CHAPTER OBJECTIVES

After finishing chapter 8 you should be able to state the purpose of an indicator, and in particular, describe when indicators 1P (first page) and LR (last record) are set ON and OFF. You should also be able to specify the use of comments statements and the DEBUG operation, and you should be able to describe solutions in which RPG II subroutines are used.

A self test is included in this chapter.

INDICATORS

In an earlier chapter we learned about the "Generated RPG II Program Logic". We broke it down into three parts: Initialization, the steps used to start a job; the Main Program, those steps that deal with the reading, processing and producing of output records; and the Last Record step that deals with activity needed to terminate a job.

Indicators are the controllers in every job. They determine "when" events are to occur in a program. They are usually turned on or off by a condition such as reading a particular type of record, or as a result of performing a test. You can also turn certain indicators on by using the SETON operation or turn them off by using the SETOF operation. Other indicators provide for special functions and are controlled by RPG II. An example of this type is the first page indicator 1P.

RULE: An indicator must be "defined" before it can be used.

By "defining" an indicator, I mean that I assign it to a particular condition in my program. You defined control level indicator L1 in an earlier chapter to one of the input record fields. When you made this assignment, we say you have "defined" indicator L1. You could then use it to control <u>when</u> certain output records should be produced.

Examine a blank coding sheet of each type and note in which positions we can "define" an indicator. Also note in which positions we can make use of an indicator after it has been defined.

* * *

I created a chart that identifies the places you might have found. If you didn't see all of these, re-examine the sheets so that you become familiar with them.

Specification Sheet	Positions Where Indicators Assigned	Indicators Used
File Description	33-34 Overflow	----
Input	19-20 Record Identifying	----
	59-60 Control Level	
	65-70 Field	
Calculation	54-59 Resulting	7- 8 Control Level
		9-17 Indicators
Output	----	23-31 Output

In general then, indicators may be defined on File Description, Input or Calculation sheet. Indicators already defined may be used on Calculation or Output sheets.

There are a few more places on the sheets that you should be aware of because we can define and/or use indicators you do not know about as yet. First, look at positions 71-72 on the File Description sheet. The eight special purpose File Condition indicators (U1-U8) are used to control when an entire file is to be used for a job. Here is what I mean.

Let us say that in a certain application we normally include the printing of a report while doing a master file update with transaction records. At other times, we do not need to print that report even though the file update activity is to be done.

For the one job then we want three files active: the master file to be updated; the activity file to provide the transactions with which to update; and the printer file. For the other job we want both the master file and the transaction file, but do not want to include the printer file.

If we know this in advance of describing the RPG II entries for both jobs, we can describe one program that uses a File Condition indicator. All you need to do is "define" the indicator on a File Description sheet for the printer file and then include it as an indicator to control every calculation or output step that relates to the printed report. Naturally, you are the person who must tell the operator (on the run sheet) which File Condition indicator is to be turned on for which run.

Here is another place to examine indicator entries. Look at an Input sheet, in positions 63-64. Indicators defined here provide for a Field-Record relationship. Here is what I mean by that.

Suppose there are two record types in one input file that contain four fields that are alike but only one of these record types has a 5th field used in the same program. We would define these record types and their fields in this way, assigning a different Record Identifying Indicator (in 19-20) to each.

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank
01	I	FILDATA	AA			Ø1	1		C	K																				
02	I																				7	9		AA						
03	I																				12	18		BB						
04	I																				19	22		CC						
05	I																				29	39		DD						
06	I		BB			Ø2	1		C	L																				
07	I																				7	9		AA						
08	I																				12	18		BB						
09	I																				19	22		CC						
10	I																				29	39		DD						
11	I																				46	5Ø	1	AMT						
12	I																													

For programmer convenience in this kind of problem we may use an entry in the Field-Record Relation columns. We also specify the use of an "OR" condition, because the fields are present in type Ø1 or type Ø2. Here is how:

RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	AND/OR	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	Position (Code 1)	Not (N)	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	FILDATA		AA			Ø1	1		C	K										
02	I		OR				Ø2	1		C	L										
03	I											7	9		AA						
04	I											12	18		BB						
05	I											19	22		CC						
06	I											29	39		DD						
07	I											46	5Ø	1	AMT			Ø2			
08	I																				

The result? Less coding for you with the same results for the RPG II program! The key lies in having record types in one file that contain many identical fields and only a few that are different.

NOTE: You may have more than one field related to its record type using these entries.

Input Field Indicators, positions 65-70 provide a means for testing the status of an input field before it is used for calculations or output. What four conditions may be tested? Why would this be desirable?

* * *

A numeric input field may be tested for a zero value or to determine its sign. One reason for testing a numeric field for a zero value is that the division operation (DIV) cannot be performed if Factor 2 has a zero value. There are a number of cases where the use of negative values would cause errors in calculated answers. By testing the record's field as it is read in, we can either bypass (GOTO) calculations or make a change to the sign (Z-SUB) before performing the steps.

The fourth condition applies to alphameric fields. We can determine whether or not an alphameric field is completely blank before using it, for example with a MOVE instruction. Sometimes you must be sure that it is blank, other times you must be sure that it is not blank.

In addition to assigning unused indicators (from Ø1 to 99) in these Field Indicator positions, we may use a halt indicator (H1-H9). Remember that halt indicators do not turn off nor does the system stop running the job until all processing and output for that record have been produced.

Suggestion: Assign only the indicators you need to do your job. Using additional indicator assignments uses computer storage and run time to test for the condition, yet may not be of benefit to your program.

Next, examine the Calculation sheet, position 7-8. We may have assigned up to 9 control level indicators on the Input sheet for use in controlling "total-time" calculations. However, as you look at the title over these columns it includes LØ and LR.

The LØ indicator (level zero) is never defined by the programmer. It is a special "total-time" indicator provided for your use by RPG II so that you may perform total-time calculations and produce total-time output records even though no control breaks occur. This indicator is turned on at the beginning of every program and remains on throughout the entire run.

The LR indicator (last record) is also provided by RPG II and normally turns on at the end of a job, that is, when the special last input record is detected. You may specify LR as:

1. a Record Identifying Indicator on the Input sheet (19-20) for identifying special end of job records,

2. a Resulting Indicator on the Calculation sheet (54-59) to identify a condition that shall terminate a job.

There is a chapter on Indicators in your RPG II reference manual. Take time to read it and then return to this text.

* * *

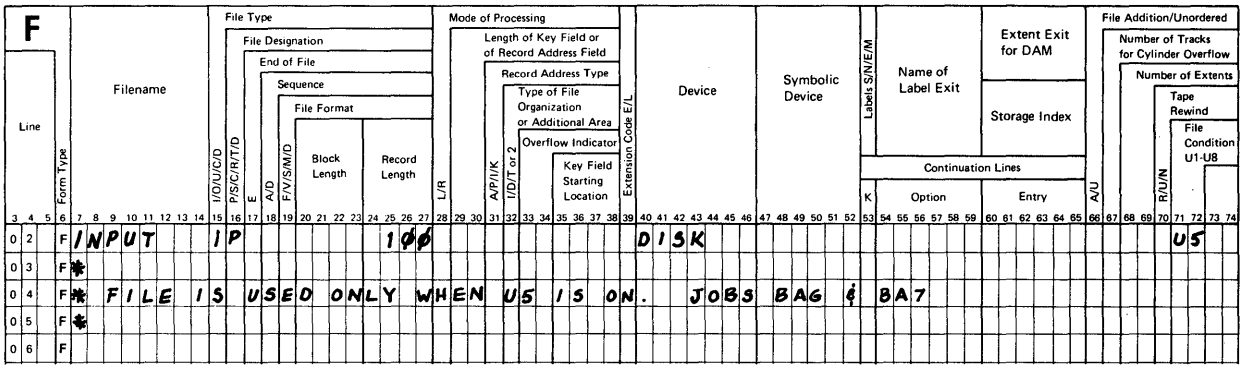
Now let's see how a programmer can get RPG II to help keep track of indicator assignment and usage. Here are two ways:

1. include "comments" statements when needed to clarify assignment or usage, and

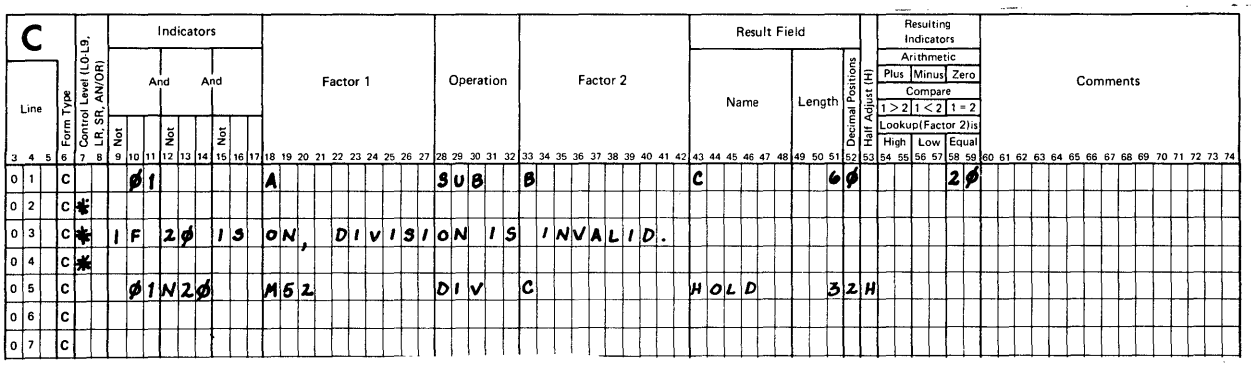
2. insert the special operation DEBUG for use when running test data to check the accuracy of your program before it is released for general use in your company.

A comments statement may be inserted at any point on any coding sheet we have used. These comments do not become a part of a program, they are merely notes for your reference (or someone else's) when reviewing the coded solution. All you do to code a comment statement is enter an asterisk in position 7 and then print your comment in the remaining spaces on that line. Here are some examples.

File Description Specification



RPG CALCULATION SPECIFICATIONS



When an asterisk is all that appears, it simply acts as a line spacer for convenience in reading the comments.

If a variety of indicators has been assigned and used in a large program with many calculations, it is sometimes difficult to keep track of their settings, especially when there are a lot of branching instructions (GOTO & TAG) included.

The DEBUG operation may be used at any point on the Calculation sheet to provide for a printout of indicator settings. We use it during test runs to verify indicator settings where we suspect (or have found out during a previous test run) that improper coding exists. Since the computer can run at high speed, it is a very nice way to trace data through calculations as opposed to desk checking by hand. Here is an example in which DEBUG and comments have been specified.

File Description Specification

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	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	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
			I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D			L/R	A/P/I/K	I/O/T or 2											K / Option / Entry	A/U		R/U/N	
02	F	LIST	O						132								PRINTER										
03	F																										
04	F																										

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: 1>2	1<2	1=2	Lookup(Factor 2)is: High	Low	Equal	Comments
			Not / Not / Not	Not / Not / Not																	
01	C					DEBUGLIST															
02	C*																				
03	C*				TEST INITIAL INDICATOR SETTINGS																
04	C*																				
05	C				LOOP	TAG															
06	C				AX	MULT	B	AXB		62								38			
07	C		38		AXB	ADD	AX	B											40		
08	C		38	N40		GOTO	LOOP														
09	C					DEBUGLIST															
10	C*																				
11	C*				TEST SETTINGS AFTER LOOP COMPLETED																
12	C*																				
13	C		38	40		GOTO	END														
14	C		N38	40		Z-SUB	B	AXB													
15	C				END	TAG															
16	C																				

Notice that the operation DEBUG must include the filename as Factor 2. This directs RPG II to write out to the specified file a list of all indicators that are on when the DEBUG operation is encountered during a test run. Every DEBUG statement in a program must refer to the same output file.

Read about the DEBUG operation in the RPG II Reference Manual to answer these items. What else can be printed in addition to the indicators list? How does a programmer eliminate the use of these debug statements after the data test runs prove satisfactory and all necessary corrections have been made?

* * *

The debug operation must include an entry for Factor 2 to identify the output file needed to list indicators that are turned on at the time. If an alphameric literal is entered as Factor 1, that literal will identify the particular DEBUG statement when it prints.

An alphameric literal is coded on the Calculation sheet exactly like a constant is coded on the Output sheet.

1. start with a single quote,

2. code up to 8 identifying characters in any combination <u>except</u> a quote,

3. end with a quote.

<u>NOTE</u>: If a quote mark is desired as a part of a literal such as M'S, you must code 2 quote marks for that one and still start and end with another. It would look like this:

'M''S'

If a field is specified as the Result Field, the contents of that field will also be printed. So, a DEBUG statement could look like this:

RPG CALCULATION SPECIFICATIONS

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / 1>2 / High (54-55)	Minus / 1<2 / Low (56-57)	Zero / 1=2 / Equal (58-59)	Comments (60-74)
01	C			'F89'	DEBUG	TESTRUN	RATE							
02	C													
03	C													
04	C													

As a result, the printed output would be identified by F89 and the contents of the field named RATE would also be printed along with the list of indicators that are on at that time.

To eliminate the use of the debug statements after a successful test run, the programmer does not change them. Instead there is a special entry on the "header" statement that takes care of the situation. Look at a File Description sheet. We have not examined the top form type (letter H in column 6) called the control or header specification.

Position 15 (DEBUG) is used to signify whether or not debugging is to be performed in the program.

1. If the number 1 is coded in position 15, a program will be compiled such that all DEBUG statements on the Calculation sheet will be active when the compiled program is run.

2. If position 15 is blank when the program is compiled, all DEBUG statements on the Calculation sheet will automatically be treated as though they were comments statements and will not be active when the program is run.

The discussion of indicators would not be complete if we did not mention their use in a program where Auto Report is included. Here are the rules:

1. Define indicators as you do in standard RPG II statements, and

2. use those indicators in the normal manner.

.... OR ....

1. Omit the Overflow Indicator in 33-34 on File Description and use *AUTO headings on Output.

   Auto Report will fill in an available overflow indicator for you.

2. Define control level indicators to Input fields in the normal manner and then enter on the Output sheet, position 39, a number from 1 to 9 to indicate for which level of total a particular value is to be calculated and printed. If a value is to be calculated and printed at last record time, enter the code R in 39.

   For example, if a total for the field named AMTT is to be printed in a total line for a control break at level 2, I code a 2 for the field named AMTT on the Output sheet.

The rule we pointed out earlier still applies.

RULE: An indicator must be defined before it can be used.

To summarize:

1. An indicator controls "when" an event occurs in a program.

2. Indicators may be defined on a File Description, Input or Calculation sheet.

3. An indicator must be defined before it can be used.

4. Some special purpose indicators are controlled by RPG II while others may be controlled by you by using SETON and SETOF operations.

5. Control level indicators are associated with groups of records.

6. When a control level indicator such as L4 is turned ON because of a control break, all control level indicators less than it are also turned on automatically.

7. When a halt indicator turns on, processing and output for that record are completed before the computer stops running.

8. The DEBUG statement provides a listing of all indicators that are turned on at the time the DEBUG is encountered (if compiled for activity). It is used to check programs during test runs and actually interrupts the normal RPG II logic cycle in order to print the listing at a specific time during calculation steps. Contents of a field may be printed.

SUBROUTINES

RPG II provides a special facility for handling subroutines. A subroutine is a set of calculation steps which may be performed following any other calculation.

To describe an RPG II subroutine:

1. enter the letters SR in positions 7-8 on every line containing a subroutine step

2. enter a unique name as Factor 1 and the operation code BEGSR (begin subroutine) as the first entry in the series

3. code the calculations to be performed in the subroutine in the order desired to solve that part of a program

4. enter the operation code ENDSR (end subroutine) on the line following the last calculation.

a. If a GOTO operation in the subroutine is used to bypass all remaining calculations in that subroutine, you may direct it to the ENDSR step by specifying a label as Factor 1 in the ENDSR step and using the same label as Factor 2 in the GOTO step.

RULE: Each subroutine must have a unique name.

RULE: Subroutines must appear last in the set of all calculations for a job.

RULE: To use a subroutine during either detail-time or total-time calculations, specify the operation EXSR (execute subroutine) and the name of the desired subroutine as Factor 2.

Here is how a program containing a subroutine runs.

When an EXSR operation is encountered, the program branches to the designated subroutine, performs the calculations in it, and automatically returns to the main program to the next step to continue normal activity.

Sample RPG II Subroutine

RPG CALCULATION SPECIFICATIONS

C				Indicators			Factor 1	Operation	Factor 2	Result Field			Resulting Indicators				Comments
Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)		And	And					Name	Length	Decimal Positions	Half Adjust (H)	Arithmetic: Plus	Minus	Zero	
				Not	Not	Not								Compare: 1>2	1<2	1=2	
														Lookup(Factor 2)is: High	Low	Equal	
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
01	C						NEW	SUB	LAST	USE	50					20	
02	C			20				EXSR	METER								
03	C							Z-ADD	USE	KWH	50						
04	C																
05	C																
06	C																
07	C																
08	C																
09	C	SR					METER	BEGSR									
10	C	SR					NEW	ADD	100000	BIG	60						
11	C	SR					BIG	SUB	LAST	USE							
12	C	SR						Z-ADD	0	BIG							
13	C	SR						ENDSR									
14	C																
15	O																

1. If indicator 20 is ON, the program is to execute the subroutine named "METER" (EXSR). All subroutine steps have SR in 7-8.

2. The first subroutine step includes the subroutine name (METER) and the operation BEGSR (begin subroutine).

3. When the operation ENDSR (end subroutine) is encountered, the program is directed back to the calculation step that follows the execute subroutine (EXSR) operation.

One subroutine may be referenced from any number of EXSR steps in the main program. In any event, the following program steps are taken:

1. branch to the designated subroutine,

2. perform its calculations,

3. return to the next main program step.

When should a series of calculations be used as a subroutine? You can use them any time you want, but a good time to use one is whenever the same calculation steps are performed at at least two different points in the main program.

Take a few minutes to read about the use of RPG II subroutines in the reference manual before you continue in this text.

* * *

Chapter 8: Summary

The proper use of RPG II indicators is key to your success as an RPG II programmer. Do not hesitate to use the System/32 RPG II Language reference manual to remind yourself of the various indicators and their proper use.

To document the use of indicators in your programs you should consider the inclusion of comments statements when an indicator is assigned, or to make a list of those used in the program by category. Here's an example.

RPG INPUT SPECIFICATIONS

I	Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C,Z,D	1: Character	2: Position	2: Not (N)	2: C,Z,D	2: Character	3: Position	3: Not (N)	3: C,Z,D	3: Character	Stacker Select	P,B,L,R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
	3 4 5	6	7 8 9 10 11 12 13	AND/OR 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
	0 1	I	* INDICATORS USED IN C5P4																											
	0 2	I	*																											
	0 3	I	* PAGE OVERFLOW												OF															
	0 4	I	* RECORD IDENTIFYING												Ø1															
	0 5	I	* CALCULATION TEST - OR Ø												21	22	23	24												
	0 6	I	*																											
	0 7	I	* RPG II ASSIGNED												1P															
	0 8	I	*																											
	0 9	I																												
	1 0	I																												
	1 1	I																												

I would place these entries ahead of all other Input entries for ease in locating the information when I need it.

Having completed chapter 8 you should be able to discuss the purpose of various indicators and specify them to solve the data processing problems. You may wish to add comments to programs to document your coded solutions. Even though you can specify the DEBUG operation, you won't appreciate its full use until you run test data on the system for programs you have written. You should also be able to specify the use of RPG II subroutines and to recognize them in programs written by other programmers.

New coding entries in chapter 8 include:

FILE DESCRIPTION

7 - 74	Comment statement (* in 7 followed by comment)
71 - 72	File Condition indicators (U1-U8)

EXTENSION

7 - 74	Comment statement (* in 7)

INPUT

7 - 74	Comment statement (* in 7)
63 - 64	Field Record Relation
65 - 70	Field Indicators

CALCULATION

7 - 74	Comment statement (* in 7)
7 - 8	Control Level (LØ, LR, SR)
28 - 32	Operation (DEBUG, EXSR, BEGSR, ENDSR)
	Note: To make DEBUG active, a 1 must be specified in column 15 of the "control" sheet (H in 6).

OUTPUT

7 - 74	Comment statement (* in 7)
39	1-9, R for use with Auto Report totals

INDICATOR CHART		
Specification Sheet	Indicators Assigned	Indicators Used
File Description	33-34 Overflow 71-72 File Condition	-----
Extension	-----	-----
Input	19-20 Record Identifying 59-60 Control Level 65-70 Field	63-64 Field Record Relation
Calculation	54-59 Resulting	7- 8 Control Level 9-17 Indicators
Output	-----	23-31 Output

SELF TEST - Chapter 8

Use a piece of scratch paper to record your answers. You should try to answer these questions from memory. As an aid for reference, you may refer to blank coding forms for this self test.

1. In your own words, what is the purpose of an indicator?

2. Match the letter of the indicator with its use. An indicator may be used for more than one answer, and there may be multiple correct choices for certain uses.

Uses	Indicators
___ Controls headings on reports except for first page	A. Control Level
___ Used on Input to identify each type of record	B. Field
___ Associated with a group of related records	C. File Condition
___ Used to determine condition during a compare operation	D. Output
___ Specific use of an entire file.	E. Overflow
___ Controls production of output records	F. Record Identifying
___ 01-99	G. Resulting
___ L1-L9	
___ U1-U8	
___ OA-OG,OV	

3. Name two indicators that are "defined" by RPG II rather than by the programmer.

4. What kind of field may be tested for a "Zero or Blank" condition?

5. What is the rule for specifying a "comments" statement?

6. What is the purpose of the DEBUG operation?

7. When is DEBUG active?

8. Where does a programmer specify RPG II subroutine entries?

9. Name the three operation entries necessary for the use of subroutines.

ANSWERS - Chapter 8 Self Test

You can score as many as 26 points on this test. If you scored less than 19 points, you may wish to re-read this chapter.

1. An indicator controls "when" an event shall occur.

2. (14 points possible)

   E

   F

   A

   G

   C

   D

   B,D,F,G

   A

   C

   D,E

3. (2 points) Any two of these choices may be counted.

   1P (first page)

   LØ (level zero control)

   LR (last record)

   H1-H9 (halts)

4. (2 points)

   A numeric field may be tested for "zero".

   An alphameric field may be tested for "blank".

5. To specify a comments statement, enter an asterisk in column 7 of that statement.

6. DEBUG causes a display of every indicator that is "ON" when the debug statement is encountered. Also, one field's value may also be displayed at that time.

7. DEBUG is active if a code 1 was specified in column 15 of the header control sheet (H in column 6) when the program was compiled.

8. On a Calculation sheet.

9. (3 points)

   EXSR to execute a subroutine

   BEGSR to identify the first entry in a subroutine

   ENDSR to identify the last entry in a subroutine

Rate Yourself

26	Fantastic!
22-25	Super
18-21	All Right

Chapter 9: Using the IBM System/32 RPG II Reference Manual

1. RPG II Philosophy
2. Creating Output Records
3. Processing Disk Files
4. Auto Report
5. Control Breaks
6. Indicators
7. Multifile Processing
8. Operation Codes
9. Tables and Arrays
10. RPG II Generated Program Logic

This chapter is designed to familiarize you with the contents of the IBM System/32 RPG II Reference Manual so that you will be able to use it as your primary source of coding information on the job.

You should have your manual plus a copy of each of the different kinds of RPG II specifications sheets listed.

Control and File Description

Extension and Line Counter

Input

Calculation

Output

Here is the approach you will follow in this chapter. First, I will present a review of the topic as presented in another chapter. Next, I will describe facilities of RPG II that are related to the same topic but that have not yet been taught. Third, I will direct you to use the RPG II manual to find answers to questions about the topic, especially about these new facilities.

When you complete a topic, you should be able to use the facilities included in it. Take your time as you read, and examine each example and illustration you come across. Do not hesitate to re-read sections of the manual before continuing with a new topic.

You are not expected to be able to use every facility to its fullest as a result of reading about it, but you will know at least:

1. what it is
2. in general, what it does
3. where an example of its use is shown

A good way of studying this chapter is to complete one topic, take a break, study the next topic, take a break, and so on.

TOPIC: The RPG II Philosophy

An RPG II programmer describes files, records and fields to be used for input, processing and output. The descriptive steps, called a source program, are arranged for analysis and compilation by a program called the RPG II compiler. After a source program has been compiled and is ready to be run, it is known as an object program.

All object programs generated from a set of RPG II descriptive statements cause the computer to run in essentially the same pattern which is:

1. Read a record.

2. Perform designated total-time (control group) calculations, if any.

3. Produce designated output for a control group, if any.

4. Perform designated detail-time (individual record) calculations, if any.

5. Produce designated output for an individual record.

When I used the word "designated" in steps 2, 3, 4 and 5, I meant do the step if the indicators assigned to the step are turned ON (or OFF if so coded).

This 5-step list provides the basic RPG II object program philosophy. As we continue the study of this chapter, you will want to note how this basic philosophy can be modified through the use of special operations such as CHAIN, KEY and SET.

Another part of the basic philosophy is that the control of every step in a generated program is dependent upon the settings of program switches known as indicators. Some indicators are assigned and set by programmer instructions. Others are assigned by the programmer's entries but are controlled by RPG II. A third group of indicators are assigned and controlled by RPG II alone. One key to the success of an RPG II programmer is knowing which indicators fall into each category and then using each available group to advantage.

A key to your success in programming in RPG II is to adopt a similar approach. Code all of the input entries for one kind of record before coding any input entries for any other kind of record. Also, code all calculations that process one kind of record before coding calculations for another kind. And, code all calculations for one control level group before coding calculations for another group. As a result, you will be able to trace what is happening more easily than if you have mixed coding entries.

IBM System/32 RPG II includes a feature called Auto Report that helps you describe printed reports. When possible make use of Auto Report to minimize your coding efforts regarding the alignment of report titles, column headings and columnar data fields. Make use of its facility to accumulate control level group totals and final totals without the need for calculation specifications.

The RPG II language makes use of a variety of coding specification forms. Each form is used for a particular reason and you already know about most of them.

FORM		PURPOSE
F	File Description	describe each type of file to be used in a program
E	Extension	describe tables and arrays
I	Input	describe each type of record and its associated fields found in input files
C	Calculation	describe each calculation in its proper order
O	Output	describe each type of record and its associated fields and constants needed to create output files

Both the File Description sheet and the Extension sheet have space for other purposes. At this time, refer to the RPG II reference manual to find answers to these questions.

1. What is the purpose of a control statement (H) entry (above the File Descriptions)?

2. What is the purpose of the line counter (L) entry (below the Extension)?

3. What is the sequence in which specifications shall be arranged to make up an RPG II Source Program?

4. The main program cycles have been described as read a record, process it, produce its output and repeat. The very first and very last program cycles in a job are somewhat different. What are these differences?

END OF TOPIC

TOPIC: Creating Output Records

Creating output records requires the description of at least one output file. An output file may be created from a mixture of input data, calculated results and constants. An output file description includes the code name of the device used to contain its records.

PRINTER is the name of a device used to hold printed records. Each line of print is considered to be a record in that output file. If a line of printing is produced for every input record processed, the report is called a "detail printed" report or "listing". If a line of printing is produced for every group of input records processed, the report is called a "group printed" report. The longest record in printer files is based upon the length of the line that the particular printer is capable of printing.

DISK is the name of a device used to store output records on the surface of a disk. These records are recorded magnetically and retained for future use until no longer needed. Disk output file records are recorded sequentially in space set aside for them when the job is started. Some disk files are stored along with index information for use in future jobs. These files are called indexed disk files and include a specification about the field used as a source of the index information. This field is called the key field. If no key field is identified, no index is built and the file is called a sequential disk file. Disk records may be up to 4,096 characters long.

Key Point: Before a record is created as output, it must be assembled in computer storage.

A 132-character print record requires 132 storage spaces. A 4,096-character disk record requires 4,096 storage spaces. You may find that records of so great a length take up too much space in the computer and are too long to be handled conveniently - for example, to print a listing of 4,096-character records would require about 32 print lines for each record!

A more convenient size might be 256 characters or 512 characters per record.

The display screen may be used as an output device. Its records are visual only, so they should not be produced unless they are useful as temporary information to the operator.

Read about display screen output records and find answers to these questions.

1. What device name is used to identify a display screen output file?

2. What entries may be made for Record Length and Block Length for such a file?

3. What effect do entries for spacing and skipping (Output sheet) have on display screen records?

You have specified special words as field names in order to include the system date (UDATE) and a page number (PAGE) as output fields. Here is a list of all the special words that are available for your use in any System/32 RPG II program.

Special Word	Purpose
PAGE	Provides automatic page numbering.
PAGE1	Provides automatic page numbering.
PAGE2	Provides automatic page numbering.
UDATE	Makes 6-position system date available.
UMONTH	2-position system date month.
UDAY	2-position system date day.
UYEAR	2-position system date year
*PLACE	Permits repetition of fields specified prior to it.

Read about these "special words" used as field names and examine illustrations of their use in the reference manual.

* * *

END OF TOPIC

TOPIC: Processing Disk Files

To update a disk file means to change the contents of one or more fields in one or more records of that file. Update files must be further described using input and output specifications. The only fields that need to be described for an update file are those whose contents are to be changed.

One method used to update a file involves chaining. This method includes identification of the update file (code U in 15) as a chained file (code C in 16) on the File Description statement. It also includes the use of the CHAIN operation on the Calculation sheet.

The examples we learned about earlier requested you to code solutions in which records in either a sequential disk file or an indexed disk file were to be updated randomly using the chaining method.

While we are on the subject of updating disk files, it is appropriate that we consider the processing of data in any type of file. Position 28 on the File Description sheet is called "Mode of Processing". Take some time to read about modes of processing in the RPG II manual. Use the following questions to direct your attention to various modes.

1. What mode of processing is identified by the entry L? by the entry R?

2. If no entry is made in position 28, what will be done in the generated program?

3. What is the difference between processing sequentially and processing consecutively?

NOTE: Processing methods are related to file organization. We've used indexed files and sequential files in our examples throughout this course. Refer to the RPG II manual for items about position 32 on the File Description sheet, "Type of File Organization".

4. How many different types of file organizations are supported by the RPG II language?

5. Which type of file may be found on

   a. disk?

   b. printer?

   c. display screen?

6. Which type of file may be processed

   a. consecutively?

   b. sequentially?

   c. randomly?

RPG II can be used to perform a special activity - processing records in an indexed file consecutively! Whenever an indexed file has been created using an unordered load or records have been added to it, it may be desirable to process the file as its records are stored. To do so, simply describe the indexed file as a sequential file by leaving positions 28-32 and 35-38 on the File Description sheet blank. The generated program will treat these records as though they were in a sequential file.

Disk files may be processed using the matching records technique. When this is done, a "matching field" is identified on the Input sheet for each record type to be processed in each file. Record types without match fields may be included in files that are processed using this technique. If so, no match field entry is made for them. Record types without match fields are selected for processing before those with match fields as they become available for processing.

You should examine the examples of such processing in the RPG II reference manual. Keep in mind that while records from two or more files are "matched", only one record is processed at a time by the generated program.

END OF TOPIC

TOPIC: Auto Report

Auto Report is a function of the RPG II language. It can be used to reduce the time required to plan and code a variety of programs. When Auto Report statements are included in an RPG II source program, that source program is diagnostically checked during an Auto Report "pre-compilation" run on the computer.

If no terminating errors are found, the Auto Report statements and the standard RPG II statements are merged into an RPG II source program that is ready for RPG compilation. Following successful compilation, the object program is ready for test data runs.

As you recall, we learned about using two features of the Auto Report function: *AUTO and /COPY. *AUTO provides for centered report titles and data fields centered under column headings. It also includes the facility to accumulate totals for selected numeric fields.

/COPY provides for the inclusion of cataloged specifications into a source program. Any RPG II specifications, including table data and array data, and *AUTO entries can be cataloged for later inclusion via the /COPY feature.

One more feature needs to be learned. It is the "option" specification. It is used to select available options such as:

1. catalog the generated source program,

2. suppress the printing of a report date and page number on the report title line,

3. suppress the printing of asterisks next to generated totals.

OPTION: Catalog the generated source program.

The reason for doing this is it provides a means for examining and modifying the intermediate level source program that is created by Auto Report prior to RPG compilation. There are instances where it is more convenient to modify the intermediate level steps, and this feature provides the programmer the option of doing this.

OPTION: Suppress the print of a report date and page number on the report title line.

There are reports that are more useful if a report date or page number appears elsewhere on the page. This option permits the programmer to make that modification.

OPTION: Suppress the printing of asterisks next to generated totals.

Again, it is not always desirable to identify certain totals on a printed report. This option eliminates those identification characters.

Each of these three options may be coded in a special, 1-line statement by following these rules:

RULE: If one or more options is desired in a program, the "options specification" must be the first statement in the program.

RULE: The options specification is coded in this manner:

POSITION	ENTRY
6	U Required
7-18	CF1,nnnnnnnn Used when a source program is to be cataloged in the library. Positions 11-18 are used for the name under which this source program is to be cataloged.
27	N Used when date and page number are to be suppressed. To retain these items, leave position 27 blank.
28	N Used when asteriks are to be suppressed. To retain the asterisks, leave position 28 blank.

There are no specific questions to be answered for this topic, but I think you should take time to read about Auto Report and look at the examples to reassure yourself that you can make use of this facility in your programming activities. Many examples showing the relationships between the coding and the report results are included for quick reference.

END OF TOPIC

TOPIC: Control Breaks

RPG II programs read, process and produce output for one record at a time. In certain problems, you need to include processing and/or output for groups of records. These groups are known as "control groups". Control groups always contain one category of information that is identical for every record in the group. Consider these situations:

1. A company hires individuals. Most are assigned to work in a particular group such as a department; the department name or number is identical for every employee in the department. The department name or number in employee records is identified as the "control field".

2. A big example is evident when you group people geographically, let's say for purposes of recording population. A person is part of a family. A family is part of a city, town, or area. The city, town, or area is part of a state. A state is part of a country, and so on. If we recorded all this information about each person in the United States, how many control levels are included?

In example 2, we have four levels of control for purposes of grouping: family, city (town or area), state and country.

These four levels of control are arranged in ascending level of control in our example, that is, "family" is the lowest level, "city" is the second level, "state" is the third level and "country" is the highest level.

Whenever you solve a problem involving groups of records, you need to do the following:

1. designate the field or fields in the input records as "control fields" by coding the letter L in 59 followed by a number in 60 to designate its level of control.

2. include the appropriate control level indicator on all calculations for the group.

3. include the appropriate control level indicator on all output for that group.

RULE: Calculations controlled by control level indicators are called total-time calculations and occur in control level order from lowest to highest.

RULE: Total-time output (T in 15) produced by control level indicators occurs in control level order from lowest to highest after total-time calculations are completed.

RULE: Header output (H in 15) produced by control level indicators occurs after all total-time output is produced and occurs in reverse order, from highest to lowest.

Let's see if I can clarify that set of rules. Assume that we are in the middle of a job and a level 3 control break occurs. Here is what happens and the order in which it will happen normally.

1. Since a level 3 control break occurred, L3 is turned on and L2 and L1 are turned on. Remember that rule.

2. All total-time calculations controlled by L1 are done.

3. All total-time calculations controlled by L2 are done.

4. All total-time calculations controlled by L3 are done.

5. All total-time output controlled by L1 is done.

6. All total-time output controlled by L2 is done.

7. All total-time output controlled by L3 is done.

At this time, L1, L2 and L3 remain turned ON.

That takes care of all group information for the old group. Now we normally start working with the record from the new group that caused the control break to occur.

1. All detail-time calculations for this first record are done. These calculations may include a control level indicator in 9-17!

2. All header output controlled by L3 is done.

3. All header output controlled by L2 is done.

4. All header output controlled by L1 is done.

5. All detail-time output for 1 record is done. This output may be conditioned by control level indicators!

At this time indicators L1, L2 and L3 are turned OFF.

You will want to read about control breaks and processing of groups of records in the RPG II reference manual. Here are some places to look for in the index and table of contents in that manual.

Indicators

Control Field

RPG II Object Program Logic

LO and LR Indicators

Group Printing

Group Indication

Control Level

* * *

END OF TOPIC

TOPIC: Indicators

An indicator is a program switch. In some cases RPG II turns them ON and OFF. In other cases, the programs can do so. Here is a list of available indicators for your use in programming. Remember that the DEBUG operation when active prints a list of all indicators turned ON at the time the DEBUG is encountered.

INDICATOR	NAME	PURPOSE
01-99	----------	Use as needed by programmer
H1-H9	Halt	Stop processing after output for that record
1P	First Page	Provide header output prior to first input record
OA-OG,OV	Overflow	Control page overflow output
L1-L9	Control Level	Facilitate group processing
LO	Level 0	Provide total-time activity when needed
LR	Last Record	Conclude job
KA-KN,KP,KQ	Command Key	External operator control of run
U1-U8	External	External file control for job
MR	Match Record	Control multifile processing

Whenever you need information about indicators, refer to an RPG II reference manual, especially to the chart that summarizes their usage.

* * *

END OF TOPIC

TOPIC: Multifile Processing

Multifile processing refers to jobs in which two or more input files provide data. When used in RPG II, multifile processing applies to programs that read records from a primary file (P in 16 on File Description) and one or more secondary files (S in 16).

The most simple example of this is:

> "Print a listing of the contents of these three files: FILEAA, FILEBB and FILECC in that order."

To do this job, which file must be designated as the primary file? How can we be sure that FILEBB will be processed before FILECC?

The primary file must be FILEAA. The second file to be described must be FILEBB and FILECC must be last. Here is how that would be coded assuming that they are all sequential disk files.

```
FFILEAA     IP      100       DISK
FFILEBB     IS       80       DISK
FFILECC     IS      100       DISK
FLIST       O       132       PRINTER
```

RULE: When two or more files are designated as secondary, they will be processed in the order specified unless otherwise controlled by the programmer.

RULE: A program may contain one and only one primary file.

That was easy. Now let's look at multifile processing in which records from one file are compared against records from another. In RPG II this is called "match fields" processing or "matching records".

RULE: When records from two files have the same value in their matching fields, the primary file record is processed before the secondary file record.

RPG II will cause two records to be read before processing begins so that matching can take place. After the primary file record is processed, another primary file record is read and the matching process is repeated.

In order to specify that matching records processing is to take place, you need to make these entries for each file used.

FILE DESCRIPTION

Specify either A or D in 18 to indicate the sequence (ascending or descending) in which records are stored on that file. All input records from the file will be sequence checked.

INPUT

Specify M1-M9 in 61-62 for the field(s) to be used for matching. Assign these entries to the same fields in both files.

CALCULATION

Use the matching records indicator (MR) to control the processing of data. Normally, this indicator is used in combination with a record identifying indicator to restrict the processing to the proper record of the available two that match (or don't match).

OUTPUT

Use the matching records indicator (MR) to control the output as desired. Again, if necessary, include a record identifying indicator to produce the proper output record(s).

That should give you an idea of what can be done and how it is coded. When you encounter a problem in which multifile processing is to be used, refer to the RPG II reference manual for specific rules and examples.

There is one more File Description entry that relates to multi-file processing. This entry is in position 17, "End of File". Here's why it is important.

A program that performs multifile processing could reach the end of one file before reaching the end of the others. This program needs some indication of whether it is to continue reading records from the other files or end the program.

Enter code "E" in 17 if reaching the end of this input or update file shall cause the job to end.

NOTE: If no entry is made in 17 for any file, the job ends after all records in all files have been processed.

* * *

END OF TOPIC

TOPIC: Operation Codes

Operation codes direct the computer to do arithmetic operations, perform tests, move data, branch to steps, and look up table or array data. There are special codes that permit the normal calculation sequence to be interrupted so that desired input or output functions may be performed.

You have learned what each of the following codes is used for and how each is to be specified. Examine this list to be sure that you can recognize each item and are able to describe its purpose.

ADD	SUB	MULT	DIV	MVR	SQRT	Z-ADD
Z-SUB	MOVE	MOVEL	MOVEA	GOTO	TAG	COMP
CHAIN	EXCPT	DEBUG	KEY	SET	SETON	SETOF
LOKUP	XFOOT	EXSR	BEGSR	ENDSR		

There are other operations that are part of the RPG II Language. The intent of this short lesson is to make you aware of each of the remaining codes by discussing them in groups. After I've completed that presentation, you should read about the rules for coding them properly and look at examples in the RPG II manual. You are not expected to memorize the proper use and coding of all codes, but you should be able to find and apply them when you need them in solving problems.

GROUP: Moving Data and Testing Zones

The move operations provide a means of moving and re-arranging data for your convenience in processing data records. RPG II provides operations to:

1. move data from one field to another,
2. move data from one array to another, and
3. move a part of a character in a field to replace the same kind of part in another.

The third category of move instructions includes four operations known as "move zone" operations. Their function is to move the zone part of a single character. What is the zone part of a character? Let's take a look.

Example: A numeric field in storage includes a digit and a sign in the rightmost position. The sign may be either plus or minus. The rightmost character in a numeric field if printed without editing will show up as a letter or as a special character. Look at these.

FIELD	RIGHT-MOST CHARACTER	PRINTED
+1234	+4 or D	123D
-707	-7 or P	70P

If for any reason you need to manipulate the zone portion of a numeric character with a sign, you can make use of a move zone operation, because the sign is the zone portion of that rightmost character! Rightmost is also called "low order" position.

Since signed numbers are the equivalent of letters or even special characters, you can agree also that if for any reason you need to manipulate the zone portion of that character, you would use a move zone operation. I said earlier that there are four move zone operations. Here's the list.

MLLZO — Move the low order position zone of Factor 2 to the low order position zone of the Result Field.

MLHZO — Move the low order position zone of Factor 2 to the high order position (left-most) zone of the Result Field.

MHLZO — Move the high order position zone of Factor 2 to the low order position zone of the Result Field.

MHHZO — Move the high order position zone of Factor 2 to the high order position zone of the Result Field.

RULE: When the zone is moved, Factor 2 is <u>not</u> changed. The zone in the Result Field character is replaced.

There are certain restrictions to the use of these move zone operations. Look at the rules and examples in the RPG II manual to find out:

1. how are they coded,
2. what are the restrictions,
3. when are they useful.

* * *

A special operation, TESTZ (test zone) provides the facility to test the zone of the left-most character in an alphameric Result Field. One, two or three Resulting Indicators are assigned to this operation to determine the following:

1. If the zone portion of the character tested is the same as a zone in the letters A-I or the &, the indicator in 54-55 (+) will be turned on.

2. If the zone portion of the character tested is the same as a zone in the letters J-R or the -, the indicator in 56-57 (-) will be turned on.

3. If the zone portion of the character tested is the same as a zone in all other letters and characters, the indicator in 58-59 (0) will be turned on.

Remember that this operation tests the high order, left-most, character in an alphameric Result Field to determine which type of zone is present in that character.

To code this operation:

1. specify TESTZ in 28-32

2. specify an alphameric Result Field in 43-48

3. include 1, 2 or 3 Resulting Indicators in 54-59.

* * *

GROUP: Bit Operations

I have included three operations in this group. BITON, BITOF, and TESTB. Unlike the operations we've studied so far, these control the turning ON or OFF of a "bit".

Definition: A bit is a binary digit.

How does that tie in with what we know? Well, a bit represents a part of a character in storage. Here is what I mean. The computer is capable of storing and manipulating characters or parts of characters, such as zones or bits. RPG II provides operations to store and manipulate numbers, letters, special characters, zone-portions of a character, and can establish a bit-portion of a character. Look at this example.

In storage, the number 4 is a number made up of 8 bits arranged like this: 11110100. Remember, I said a bit is a binary digit.

Binary digits are either 0 or 1 as the example shows. Look at this list of other examples.

CHARACTER	BINARY EQUIVALENT
1	11110001
C	11000011
*	01011100

Now don't get excited. You are not supposed to remember these codes. I just wanted you to realize that every storage position can hold an 8-bit character.

* * *

Since there are eight bit-positions in one storage position and RPG II provides two operations to turn bits on (meaning make them a 1) and to turn bits off (meaning make them a 0), we now have the facility to use a storage space as though it had ON/OFF switches.

First, here is how to code the operations of BITON (turn on a bit) and BITOF (turn off a bit).

1. To turn on from 1 to 8 bits in a single storage position:

   a. enter BITON in 28-32.

   b. enter an alphameric constant or a 1-position field as Factor 2 to control which bit out of the eight shall be turned on; bit positions are numbered from 0 to 7 for the eight bits (starting from the leftmost position).

   c. enter a 1-position alphameric field as the result field to designate where bit manipulations shall take place.

Examples: (Calculation specifications)

```
C                   BITON'Ø1234567'FEELD    1
C                   BITON'25'      RETO
C                   BITON'Ø'       SET      1
```

What do you think is accomplished by each of these instructions?

* * *

The first example means turn on all 8 bits in FEELD.

The second means turn on bits 2 and 5 in RETO.

The third means turn on bit Ø in SET.

RULE: A 1-position field may be used as Factor 2. If so, bit positions that are on in the field in Factor 2 will cause the same bit positions in the Result Field to be turned on.

Now for BITOF. As you suspected, the same rules apply but the designated bit positions are turned OFF, that is, they will contain zeros.

Well, so far all we've accomplished is that we know how to turn bits ON and turn bits OFF in a 1-position alphameric field. That is like being able to turn on a light switch or turn it off. What good is that?

Suppose I change the sentence to read:

> That is like being able to turn on a program switch or turn it off.

Aha! Now I can use this 1-position alphameric field to store 8 program switches which I can use in combination with indicators to establish, test and reset a set of conditions.

BITON will establish the conditions, BITOF will reset them, and the third operation TESTB (test bit) will be useful in determining their condition during a program. Look at this calculation sheet example:

```
C                    TESTB'35'      HOLD  1    445566
```

Bits 3 and 5 are compared against the corresponding bits in the field named HOLD. Which indicator will be turned on, 44 or 55 or 66?

RULES: When the same bits are on in both Factor 2 and the Result Field, the indicator in 58-59 (=) is turned ON.

When the bits that are ON in either entry are OFF in the other, the indicator in 54-55 (+) is turned ON.

When there is a mixture, some that are on in one are off in the other, the indicator in 56-57 (-) is turned ON.

In effect then, the TESTB operation compares bit settings to determine a condition.

RULE: At least one Resulting Indicator must be specified in positions 54-59.

Take time to look at the RPG II manual for more examples in which these operations BITON, BITOF and TESTB are used. You may find them useful in future programming activity.

* * *

GROUP: Control of Input and Output

Normally, the generated RPG II program includes instructions that read in data prior to being processed. Likewise, RPG II provides instructions to write or print output records after all calculations have been performed.

In some data processing problems it is necessary to read in one or more records when needed during calculation steps. Also, it is sometimes necessary to produce output records during calculation steps.

You have already studied about some operations that can alter the normally generated RPG II calculations in order to perform input or output functions. Do you remember them?

OPERATION	PURPOSE
CHAIN	read a disk file record
KEY	read a keyed record
SET	turn on an indicator corresponding to a keyed entry
EXCPT	write out a record or display it on the screen
DEBUG	list the indicators presently turned ON; also, list the contents of a specified field

There are three other operations that may be used to control input or output functions during the calculation steps.

1. FORCE: Select the file during calculations from which the next record is to be read when the normal RPG II program logic calls for a record.

2. READ: Read a record immediately from a "demand" file so that it can be processed during calculations that follow.

Definition: A demand file is one that is either an input, update or combined (special) file type.

All demand files (D in 16 on the File Description sheet) except those assigned to the KEYBORD device must be processed using the READ operation.

3. SETLL: Set the lower limits of an indexed demand file being processed.

Using this operation allows you to process an indexed file sequentially by starting at a point other than its beginning.

The rules for coding these three operations are listed.

1. FORCE:

   a. specify FORCE in 28-32

   b. specify the filename of the file from which the next record shall be read as Factor 2

2. READ:

   a. specify READ in 28-31

   b. specify the name of the demand file as Factor 2

   c. specify an indicator in 58-59 to be turned on at the end of file for the demand file; this entry is optional

3. SETLL:

   a. specify SETLL in 28-32

   b. specify a field or literal that contains the value of the lower limit as Factor 1; its length must equal the key field length

   c. specify the indexed filename as Factor 2

Before you include any of these operations in your programs, read about them in more detail in the RPG II reference manual.

Key Point: RPG II generates input and output instructions from your descriptive statements of the total job. You may alter the normal pattern if you have the need to, BUT be sure your job requires it before you indiscriminately use these operations.

* * *

GROUP: Branching to External Subroutines

This is the last group of operations to be covered. The two entries, EXIT and RLABL, provide an RPG II programmer the facility to link his program to one subroutine written in another programming language called Assembler.

If such an external subroutine exists and is available for use, here is how the RPG II programmer calls for it in his program.

1. specify EXIT in 28-32

2. specify the name of the external subroutine as Factor 2

When the EXIT operation is encountered, the RPG II program branches to the Assembler subroutine for processing. When the subroutine is finished, branching again occurs back to the RPG II program.

The RLABL operation may be used so that the external subroutine may reference a field, table, array or indicator in order to do its processing. To use RLABL:

1. specify RLABL in 28-32 on the line immediately following the EXIT operation

2. specify either a field name, a table name, an array name or an indicator as the result field.

NOTE: If an indicator is to be entered, it must be entered as IN in 43-44 followed by the specific indicator in 45-46.

If the situation arises when you may have access to an Assembler subroutine, read about the EXIT and RLABL operations in the RPG II manual.

* * *

END OF TOPIC

TOPIC: Tables and Arrays

A table is an arrangement of data items having like characteristics. So is an array. Table names must start with TAB, array names can be any name except those beginning with TAB.

Tables and arrays may be searched, one entry at a time, using the operation LOKUP. The search always begins at the first table or array element and continues until the desired element is found or the entire table or array has been searched. There are five different Resulting Indicator entries that may be considered when conducting a search.

1. Search until an equal is found. Elements may be arranged in any order.

2. Search until an element is higher than the search field. Elements must be in ascending order.

3. Search until an element is either equal to or higher than the search field. Elements must be in ascending order.

4. Search until an element is less than the search field. Elements must be in descending order.

5. Search until an element is either equal to or less than the search field. Elements must be in descending order.

Two operations apply only to array processing. XFOOT may be used to add the value in every element of a numeric array. MOVEA may be used to move an array to another array; to move an array to a field; or to move a field to an array.

Individual elements of an array may be used for processing or output by including an "index" value or field when designating an array name in that operation or output statement.

When two tables or arrays are loaded, they may be arranged in "alternating format". If so, they must be described on the Extension sheet on one line. Also, the table or array named first must have its data appearing first in the set of alternating data.

Read about tables and arrays in the RPG II reference manual if you want to know more about them in regard to:

1. What is meant by a "compile-time" or a "pre-execution-time" table or array?

2. What is an "execution-time" array?

3. How can the contents of a table or array be modified?

4. How can I add entries to a short table?

* * *

END OF TOPIC

TOPIC: RPG II Generated Program Logic

The RPG II manual includes a description of this topic from two points of view. First, for relatively simply jobs involving one input file and second, for more complex jobs. Refer to both sections of the manual to answer these questions.

SIMPLE

1. In what order do input, processing and output occur in simple jobs in which at least one control level has been defined and used?

2. When are indicators turned off?

3. In what way is the first program cycle different from normal job cycles?

4. In what way is the last program cycle different from normal job cycles?

COMPLEX

5. When are pre-execution-time tables loaded?

6. When are calculations controlled by LO performed, during detail-time or total-time calculations?

7. Page overflow is controlled by one of the indicators OA-OG or OV. When is overflow output produced? When is the overflow indicator turned off?

Knowledge of the generated program logic will be helpful during the debugging and testing of programs as well as during coding activity. Refer to these sections frequently during the first few test runs to become as familiar as you can with this logic. RPG II is a descriptive and generative language. To make the most of its features, you need to know the logic of programs it generates from your descriptions.

END OF TOPIC

This chapter was written to provide some study in areas of the RPG II language which were not covered in the first part of the course. You can best learn to use RPG II by using it on the job. The reference manual should be near you and you should use it to advantage whenever you need it.

For those of you who will be describing data processing jobs that involve telecommunications, get a copy of the separate manual, "IBM System/32 RPG II Telecommunications Programming" form number SC21-7597. This manual provides information on:

1. definitions of basic telecommunications terms,

2. descriptions of IBM System/32 telecommunications capabilities, and

3. specifications rules for writing telecommunications programs.

Telecommunications specifications are coded on a separate form (Form Type T in column 6) which has the form number GX21-9116. These statements are placed just ahead of Input specifications when the source program is ready for compilation.

There is no self test for this chapter. Here's wishing you success in your programming activities using the RPG II language.

TOPIC/SUBTOPIC INDEX

TOPIC/SUBTOPIC INDEX

International Business Machines Corporation

General Systems Division
5775D Glenridge Drive N. E.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

IBM General Business Group/International
421 Boston Post Road
Port Chester, New York 10573
(International)

SR30-0017-1