GC21-7514-3
File No. S34-28

Systems

Introduction to RPG II and RPG III: Batch Processing with Program Described Files

IBM

Fourth Edition (August 1980)

This is a major revision of, and obsoletes, GC21-7514-2. Major changes include using the latest level specification forms and updating the terminology. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or addition.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, be sure you have the latest edition and any technical newsletters.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. Use this publication only for the purpose stated in the *Preface*.

Publications are not stocked at the address below. Requests for copies of IBM publications and technical information about the systems should be made to your local IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

This book is intended to help you learn to use RPG to write programs that produce printed reports. The book is designed for people with no previous knowledge of computers and programming and for people who already know a programming language but who want to learn about report writing in RPG.

After reading this book, you should not expect to be able to write complex RPG programs. You will be introduced to only a small part of RPG. Many important features in RPG are not discussed in this manual; they are discussed instead in the RPG reference manual for your system. However, this manual provides enough background knowledge so that you can readily learn–from reference manuals, classes, or IBM personnel–more detailed information that is required for writing programs for your system.

For most systems the RPG programming language is called RPG II or RPG III. As the Roman numerals indicate, these two RPG products differ somewhat. However, the basic concepts of batch processing with program described files apply to both RPG II and RPG III. Therefore, this manual refers to the RPG programming language as simply RPG. Note that this manual does not discuss interactive processing or externally described files.

How This Manual Is Organized

The first chapter describes, in general terms, how a system operates and what you must do to run a program. The information in that chapter answers such questions as:

- What are the parts of a data processing system?
- What is a program?
- What is a programming language?
- What is an RPG program?
- How is an RPG program run on a system?

The second chapter describes the RPG program cycle and the RPG specifications you must write to do a particular task. The material in that chapter provides a gradual development of concepts, from the simple to the more complex. Thus, it is important to read the material in sequence. Sample programs illustrate the concepts presented.

The third chapter explains an RPG programmer's job more fully. It shows the things a programmer must do from the start of a sample program to its completion.

RPG Coding Forms

Following is a list of forms used to code and debug RPG programs. Contact your local IBM branch office to order any of these materials:

- RPG Control and File Description Specifications, GX21-9092
- RPG Input Specifications, GX21-9094
- RPG Calculation Specifications, GX21-9093
- RPG Output Specifications, GX21-9090
- RPG Indicator Summary, GX21-9095
- RPG Debugging Template, GX21-9129
- Printer Spacing Chart, GX20-1816
- Record Layout Form, GX20-1702
- Disk File Layout Chart, GX21-9108
- Printer/Display Layout, GX21-9174

This page is intentionally left blank.

Contents

Chapter 1. Basic Data Processing and Programming Concepts

To some people, data processing seems complex and mysterious, and data processing machines seem even more so. Actually, data processing is logical and straightforward. There is no more mystery about it than about most familiar business activities. For example, the statement we receive from a department store, listing our charges and payments for the preceding month, is processed data. Data processing merely means performing a series of planned operations on data to achieve desired results. The machines are simply tools that handle the volume and repetition of data processing.

PARTS OF A DATA PROCESSING SYSTEM

All data processing systems include input devices, output devices, and a processing unit.

Input Devices

Input is the data to be processed. The devices used for getting that data into the system include card readers, magnetic tape units, magnetic disk units, diskette storage devices, and work station keyboards. Information about the input devices applicable to your system is in the RPG reference manual for your system.

Output Devices

Output is the processed data in usable form. The devices that produce the output include printers, card punches, magnetic tape units, magnetic disk units, diskette storage devices, and work station display screens. Information about the output devices applicable to your system is in the RPG reference manual for your system. Because this manual deals only with report writing in RPG, we refer only to printed output.

Processing Unit

The main part of a data processing system is the processing unit. The processing unit controls the operation of the system, performs the arithmetic/logic functions, and contains the system's memory area, called *storage*. Storage holds the data and the instructions that tell the system how to process the data. Storage capacity is measured in bytes, K-bytes, or megabytes. A byte is the representation of one character. One K-byte is 1024 bytes ($1024 = 2^{10}$). A system that has 32 768 bytes of storage is called a 32-K system. A megabyte is one million bytes. Thus, 8.6 megabytes are 8 600 000 bytes.

PROGRAMS AND PROGRAMMING LANGUAGES

By itself, the system cannot know how it should process your data. Therefore, whenever you want the system to process data, you must tell it:

- When to read each item of data to be processed
- What form the data will be in
- Where in the system the data is stored
- What calculations to perform on the data
- What form the calculated results will be in
- How to record the results for further use

In short, a data processing system requires a set of instructions that describe the data and that identify each successive step of processing to be performed on the data. These instructions are called a *program.*

To communicate with the system, you must use the system's language, or one that can be translated into that language. The system's language is called *machine language.* It consists of letters, numbers, and symbols that, when properly arranged, have a specific meaning to the system and that, when interpreted by the system, cause it to perform a desired function.

Because machine language is so very different from our own language, it is extremely difficult to use it to write a program. For this reason, programming languages have been created. A *programming language* allows the programmer to use familiar words and symbols to write instructions.

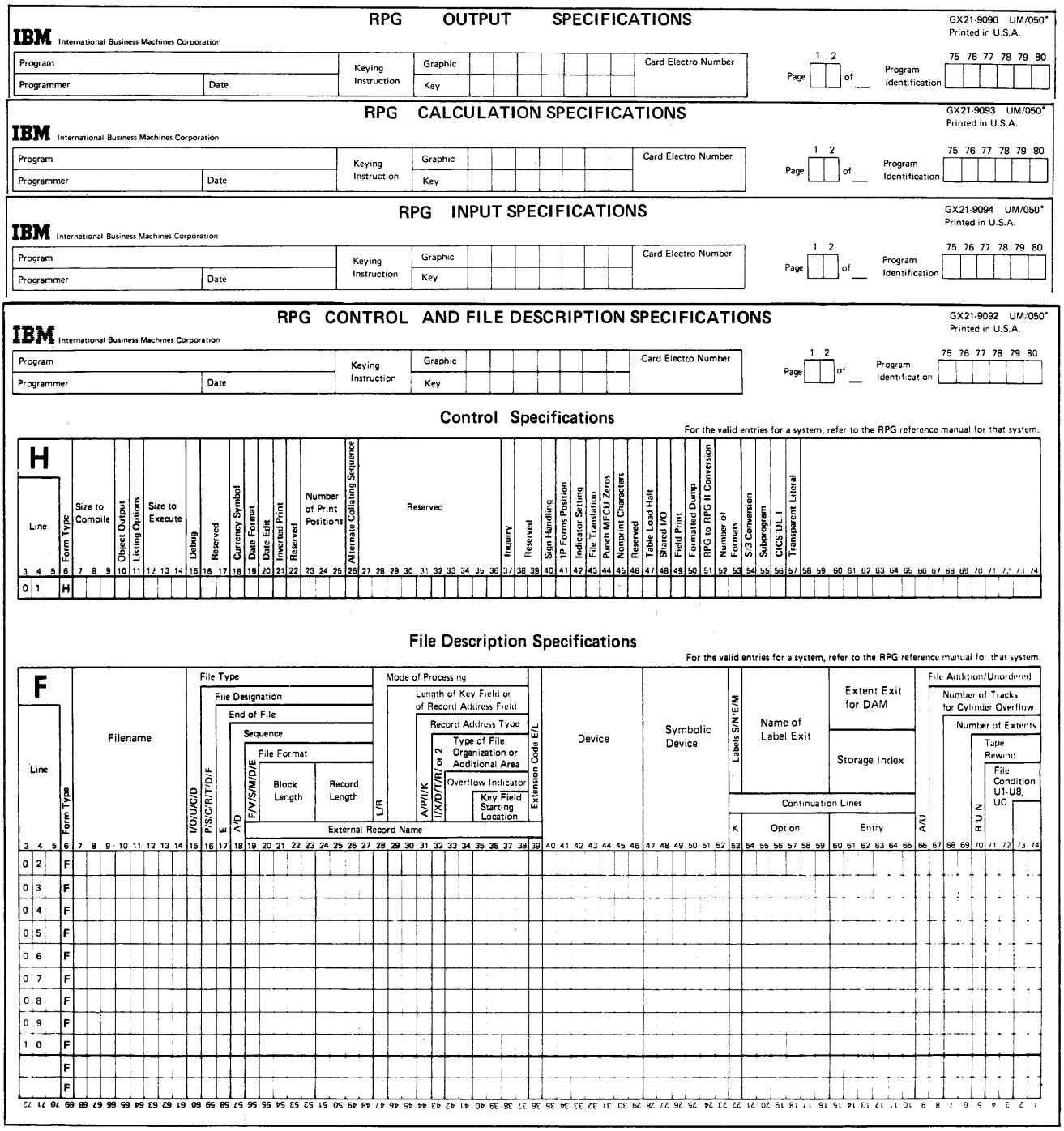
The RPG programming language is composed of letters, numbers, and symbols that you put together to form an instruction. When creating instructions in the RPG language, you must follow certain rules just as you would when constructing a sentence in English. You will learn about these rules in Chapter 2 of this manual.

The set of instructions you write is called a *source program.* This source program is translated, by a program called the compiler, into a machine language program called the *object program.* The data processing system uses the object program to process data. In fact, the system can use it over and over to process several sets of data.

Source Programs

The instructions you write for any program must describe the input, processing, and output requirements of the program. For example, one instruction might direct the system to read an input record, another might tell the system to add two numbers, and another might tell the system to print a line on the printer. Because not all programs are the same, you provide a different set of instructions for each program.

To write the instructions, you fill out RPG specification forms (see Figure 1). These forms have been specially designed to help you write instructions according to the rules of RPG language. The act of writing instructions on these forms is called *coding*; the entries you make on the forms are called *specifications*.



Notes:

1. The specification forms shown in this manual are used for RPG programs on several IBM systems. However, not all column headings refer to valid entries in those positions for all systems. Refer to the RPG reference manual for your system to determine which entries are valid for your system.
2. Column headings on other versions of these forms may vary slightly from those shown in this manual.

Figure 1. RPG Specifications Forms

To describe the input, processing, and output requirements of your program, you supply information on each form. For example, you have to describe your input data and specify what device (such as a disk unit) will read it. You also have to describe how the input data is to be processed. This includes specifying what type of operations (such as add or subtract) must be performed on the data. Finally, you have to specify what kind of output you want (such as a printed report), what information must be included in the output, and how that information should be arranged.

After you have coded the specifications forms, the next step is to get the coded information into the system. The system cannot read the coded forms, so you must put the specifications into a format that the system can read. Depending on your system, you key the specifications onto a diskette, into punched cards, or directly into the system.

Translating Source Programs into Object Programs

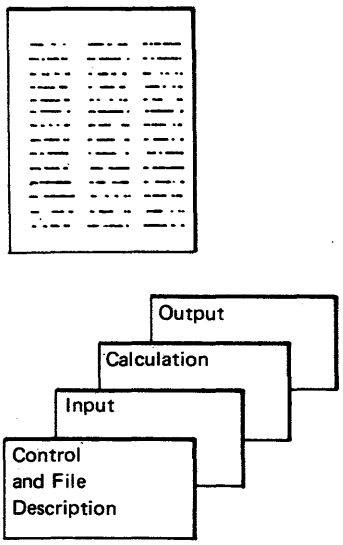
As we said earlier, the system understands only machine language. It cannot use a coded program (written in a programming language like RPG) directly. Any program you write in RPG must be translated into machine language. The translator is a program called a *compiler*. The RPG Compiler program is available from IBM.

The compiler translates your RPG specifications (source program) into machine language (object program). The translating it does is called *compilation*. Essentially, the compiler performs three functions during compilation:

- It determines what machine instructions the system needs in order to perform the job described by your RPG specifications.
- It translates your RPG specifications into a machine language program.
- It assigns storage locations to program instructions and data.

Summary

Figure 2 illustrates the steps you must go through to produce a report when you use RPG.



(1) Determine the requirements of your program. Define the input and printed output. Also decide what processing must be done in order to get the proper results.

(2) Write the source program on the RPG specification forms.

Figure 2 (Part 1 of 2). Steps in RPG Data Processing

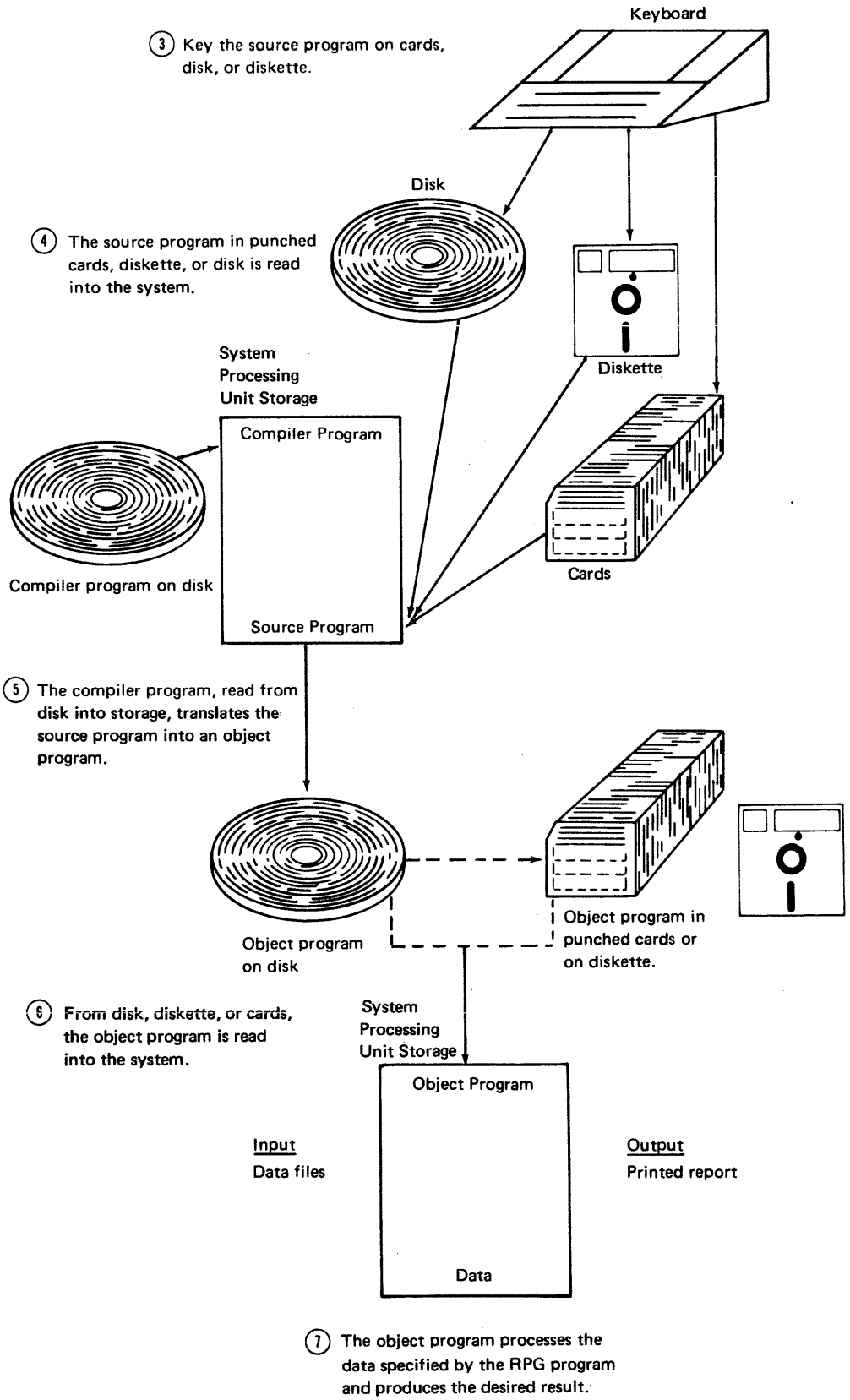


Figure 2 (Part 2 of 2). Steps in RPG Data Processing

DATA PROCESSING TERMS AND PROGRAMMING AIDS

Data Processing Terms

The basic unit of organized data is the record. A *record* can be defined as a group of related data items treated as a unit. The system reads and processes data one record at a time. A sequence of records containing the same kind of information is called a *file*. Within a record, each data item is called a *field*. Fields vary in length, depending on the number of characters each data item contains. For example, the field for a customer's name would be longer than the field for a date. Fields can be alphabetic (for example, a name field), numeric (for example, a credit limit field), or character (for example, an address field). Figure 3 illustrates the meaning of the terms *field*, *record*, and *file*.

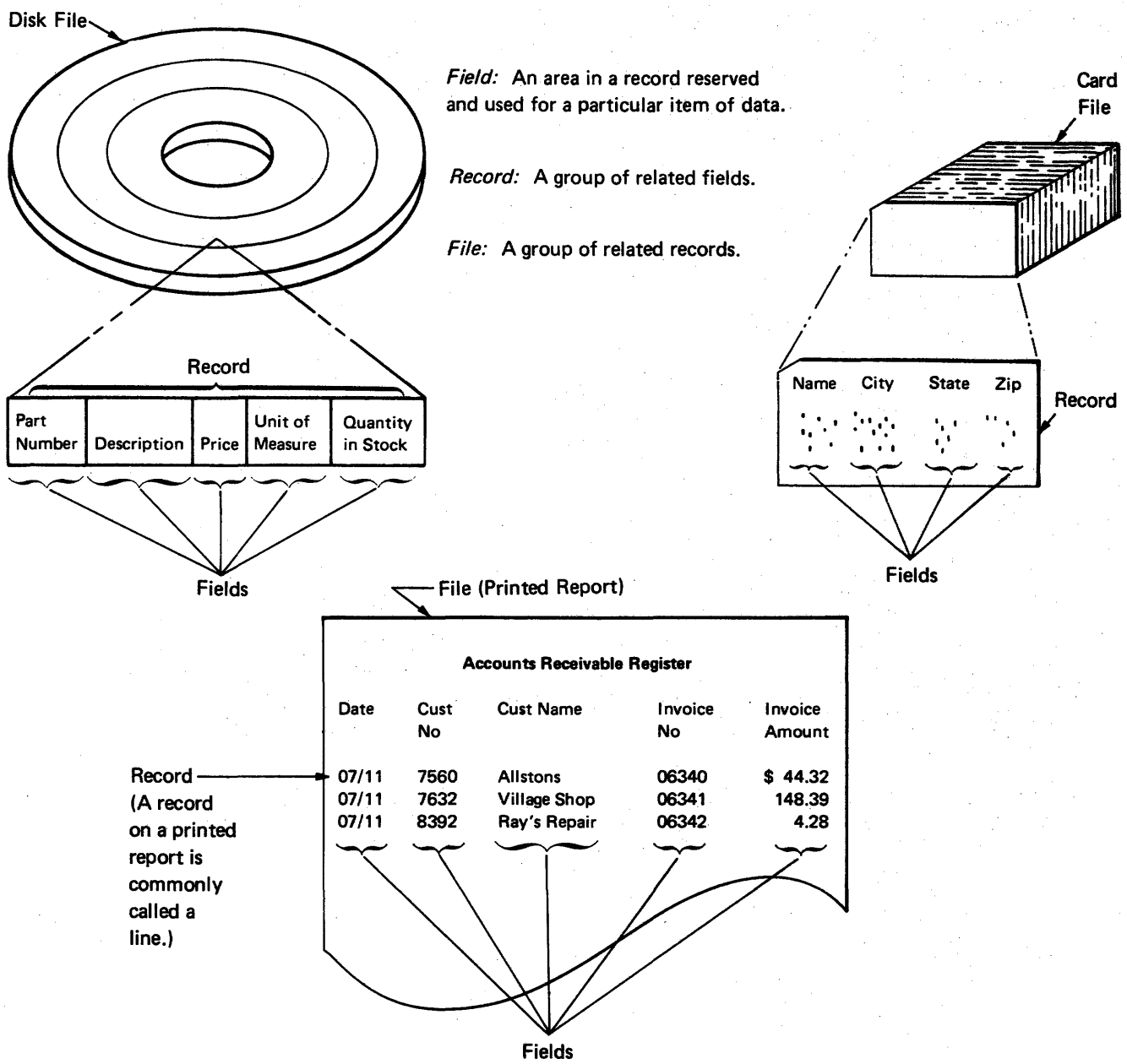
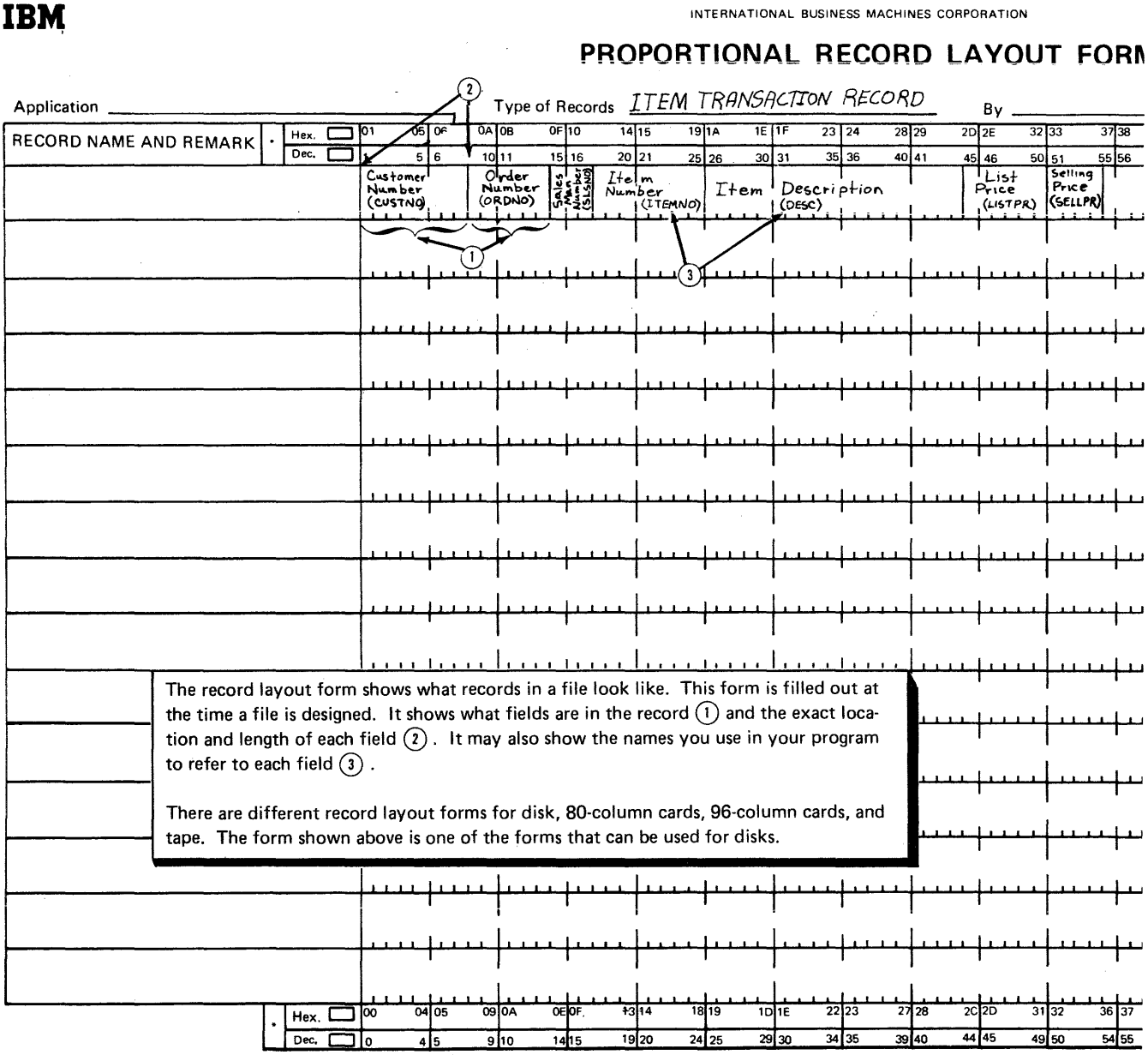


Figure 3. Basic Data Processing Terms

Programming Aids

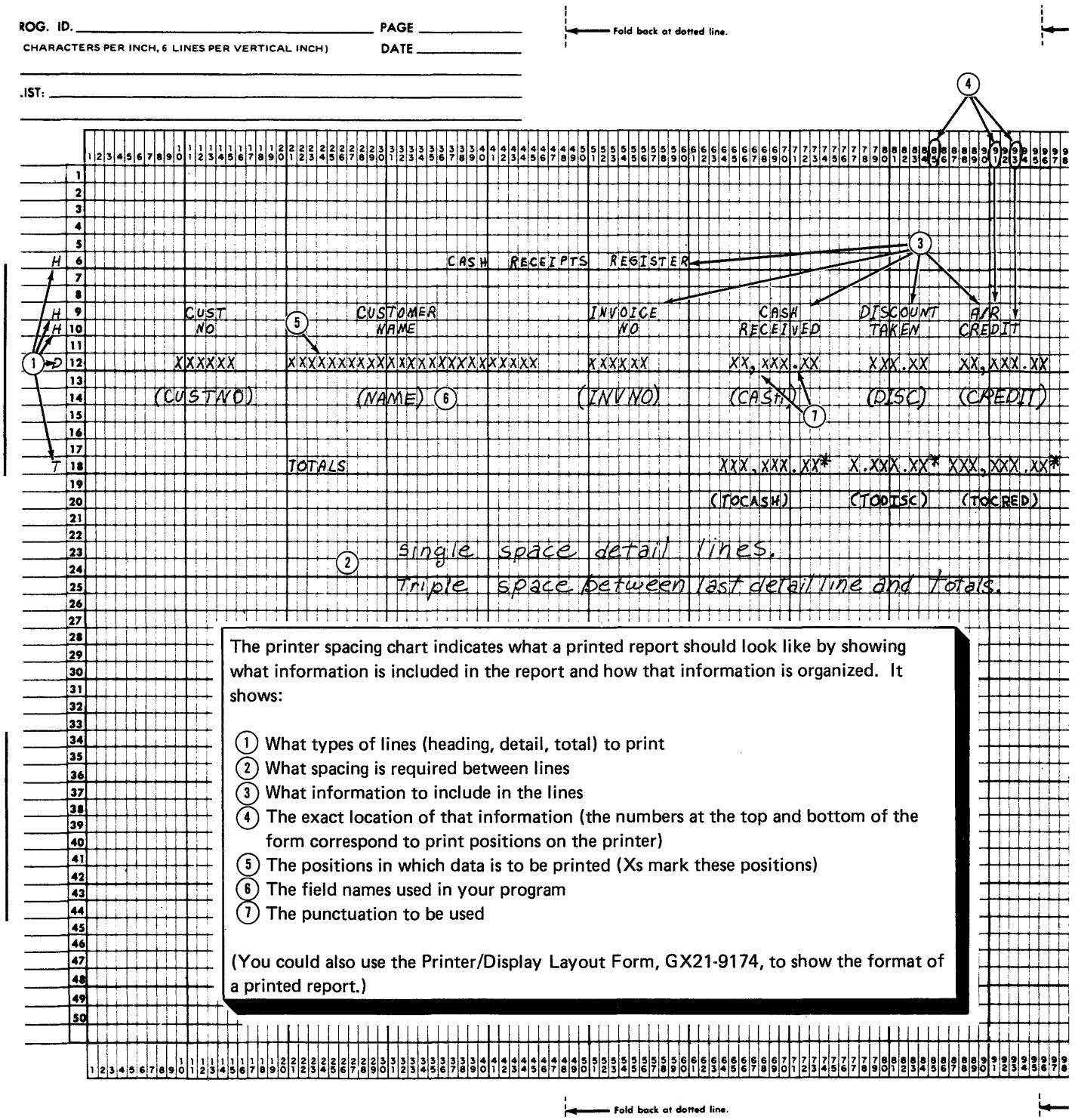
Figures 4 and 5 are examples of some forms to use when you prepare to write a program.



*Two numbering arrangements, each in hexadecimal and decimal notation, are shown. Select the arrangement and notation used by checking the appropriate box to the left.

†The number of forms per pad may vary slightly.

Figure 4. Record Layout Form



The printer spacing chart indicates what a printed report should look like by showing what information is included in the report and how that information is organized. It shows:

1. What types of lines (heading, detail, total) to print
2. What spacing is required between lines
3. What information to include in the lines
4. The exact location of that information (the numbers at the top and bottom of the form correspond to print positions on the printer)
5. The positions in which data is to be printed (Xs mark these positions)
6. The field names used in your program
7. The punctuation to be used

(You could also use the Printer/Display Layout Form, GX21-9174, to show the format of a printed report.)

Figure 5. Printer Spacing Chart

Chapter 2. RPG Programming Language

RPG Program Cycle

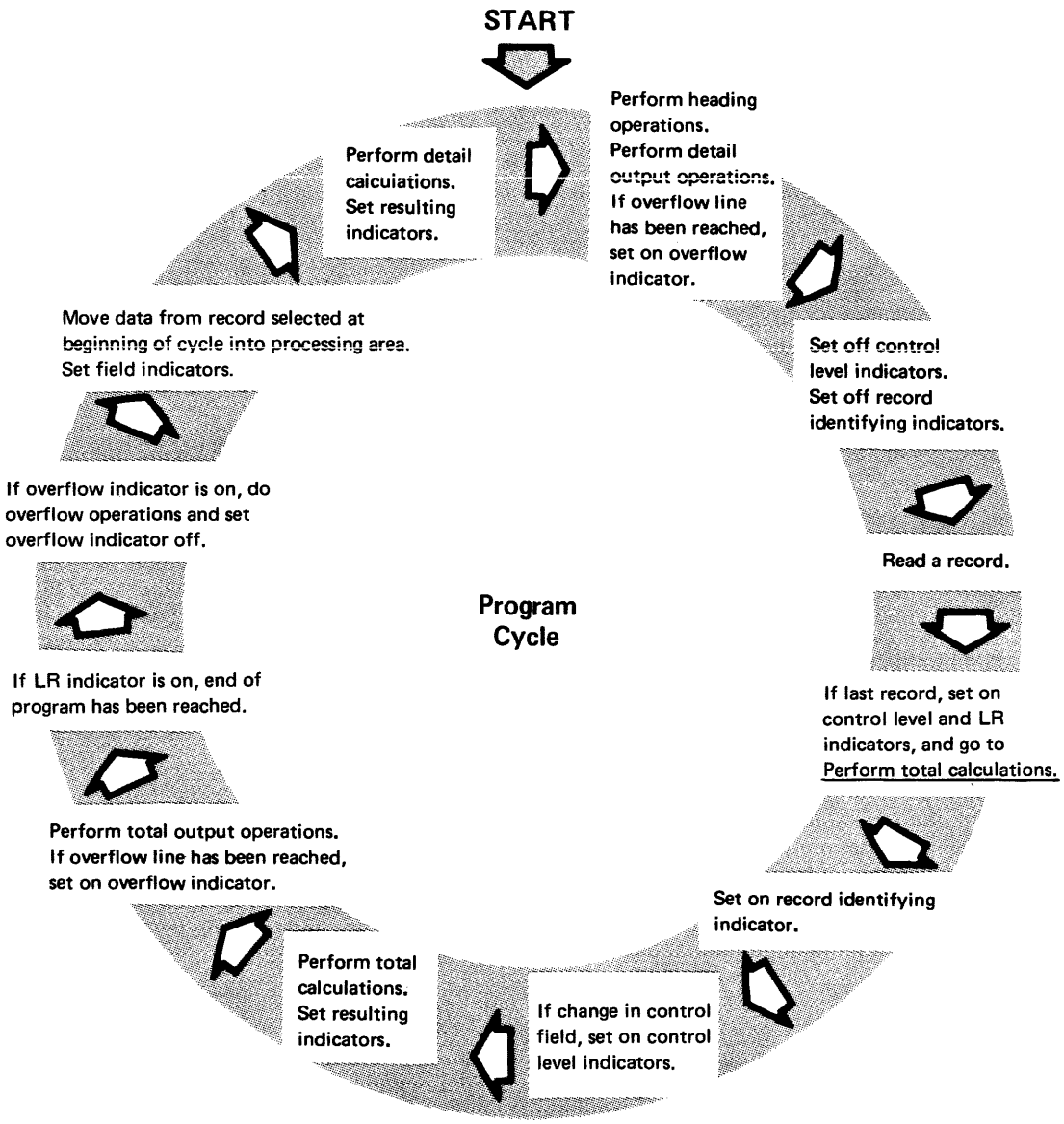
When a system processes data, it must do the processing in a particular order. This logical order for the processing is supplied by the RPG compiler and by your coding.

The logic the compiler supplies is called a *program cycle* (see Figure 6). The object program goes through this cycle of operations every time a record is processed. Depending on your specifications, the object program may or may not use a particular operation in the cycle. However, the program still goes through the complete program cycle every time. Because one program cycle is needed for each record read, many program cycles are required for every program.

It is important that you know the order of the operations in the RPG program cycle. This knowledge enables you to write specifications that use the cycle correctly. By knowing the order in which the operations in the cycle are performed, you can organize your program efficiently and save yourself unnecessary coding.

This chapter explains the operations in the RPG program cycle a few at a time. You will learn:

- Which operations are used for a particular function.
- Which RPG specifications you must write to use the function.



Notes:

1. The program cycle shown gives the general order of the operations. This cycle may vary slightly from the detailed cycle discussed in the reference manual applicable to your system.
2. You do not need to memorize the program cycle. The cycle is shown at this time only to give you an idea of the cycle of the operations. The operations will be discussed in greater detail later.

Figure 6. Program Cycle

Writing Specifications for Input and Output Operations

One of the simplest programs you can write is one that reads information from an input record, such as a card, and then writes that same information in the form of a printed report.

Program Cycle Operations

This simple program uses only the three most basic operations in the RPG program cycle. Figure 7 shows these operations.

Notice that two operations concern the basic requirements of a program: input (read a record) and output (detail output). The third operation is the movement of data inside the system.

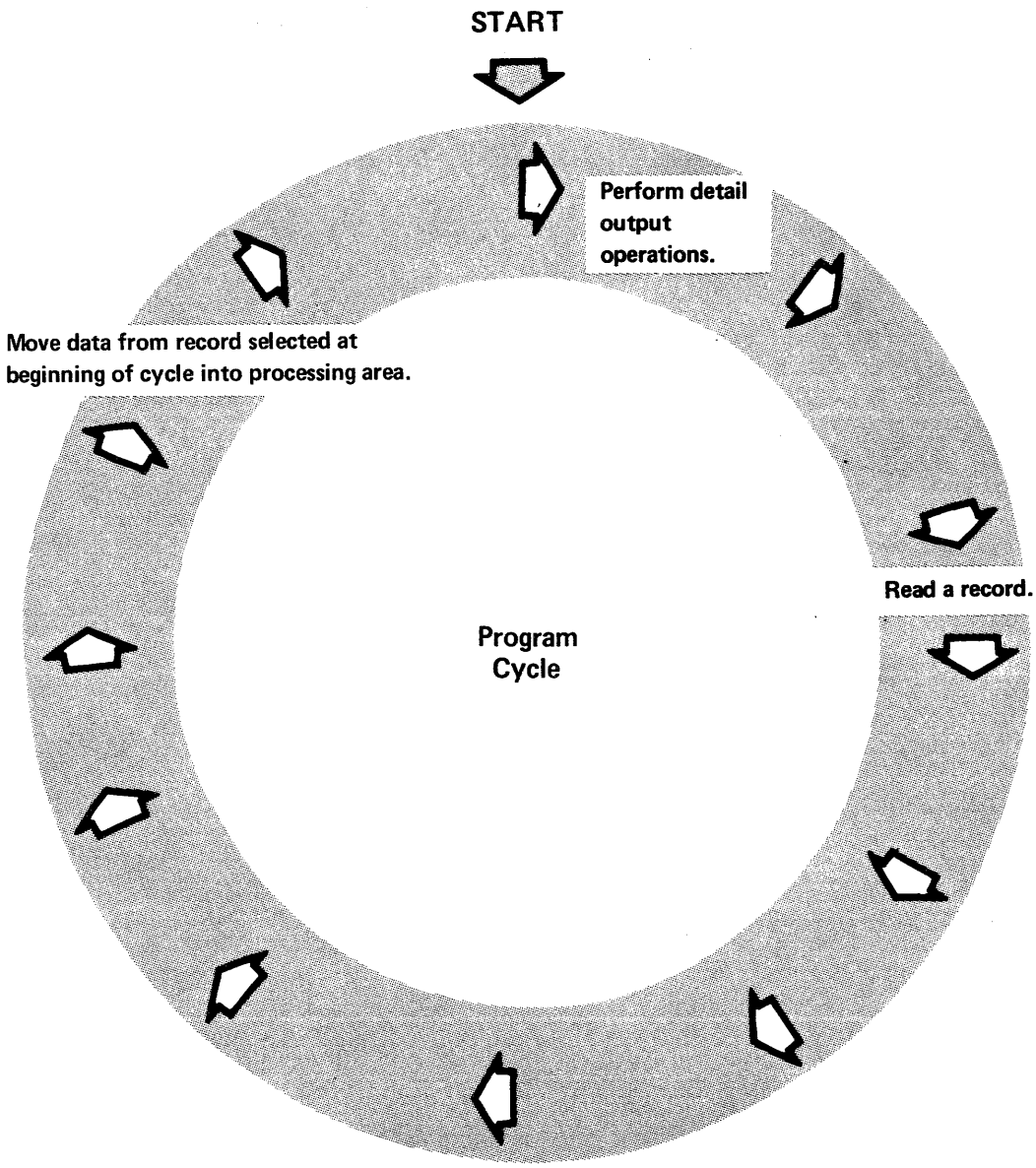


Figure 7. Three Basic Operations in the RPG Program Cycle

Data read by an input device must be transferred to the system's processing unit before it can be used. Moving data is a mandatory operation for every program. Because this operation is mandatory and is done exactly the same for every program, the compiler can supply instructions to do it.

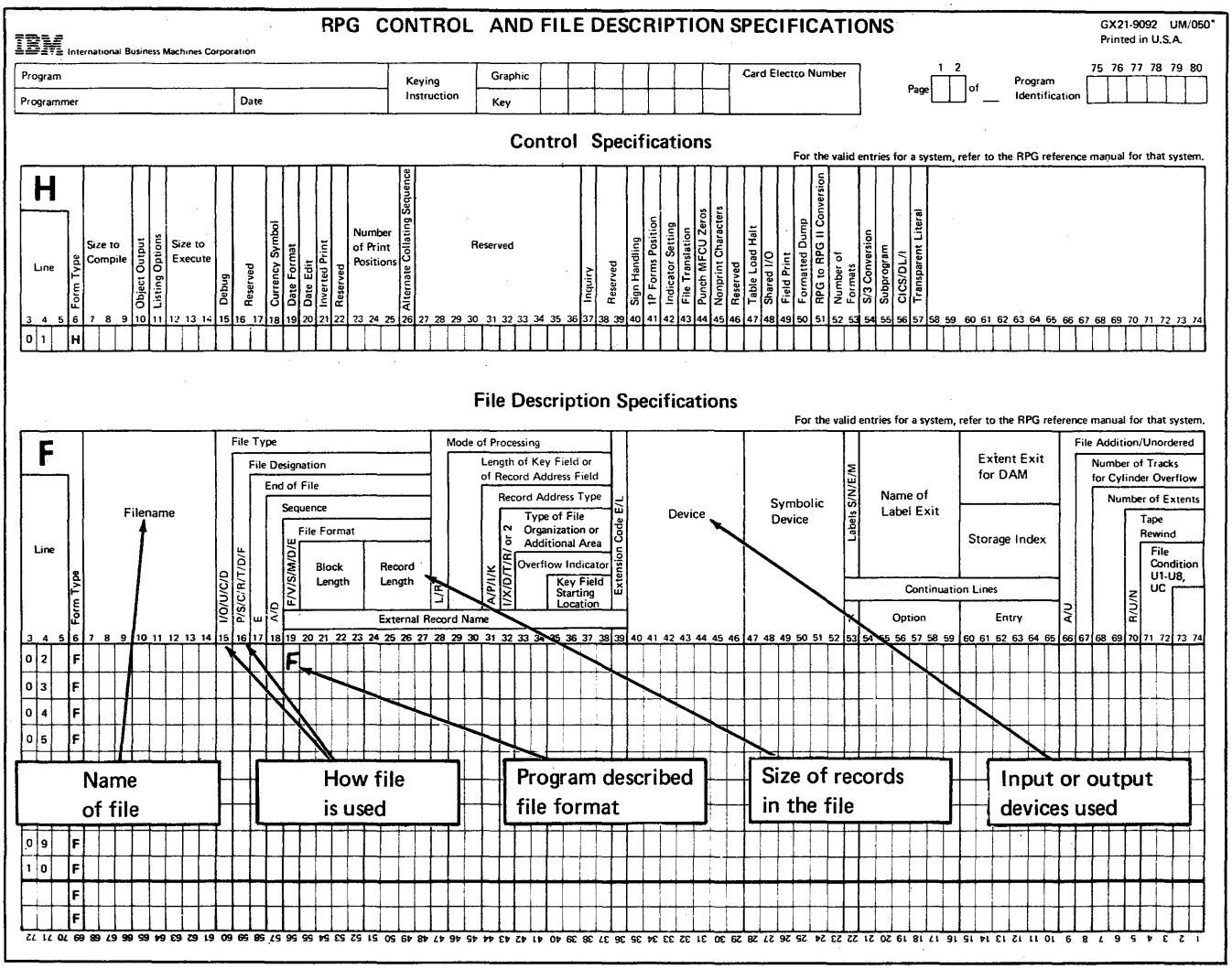
When the program is executed, the program cycle is repeated over and over. All three operations are used for every record in the input file. The term *detail output* in the cycle operation means output operations that are performed for every input record.

It may seem strange that detail output comes before a record is read. This occurs, however, so that headings can be printed on a report. If a program does not print headings, no information is printed during the first cycle.

To use these cycle operations properly, your specifications must describe the records in the input file and specify how the output records should be created. You must also indicate what devices are used in the program.

DESCRIBING THE FILES

The File Description Specifications form is used to describe all the files used by your program. This description includes the name of the file, the device used with the file, and information on how the file is to be used. You describe the files by filling out the indicated positions on the bottom half of the form.



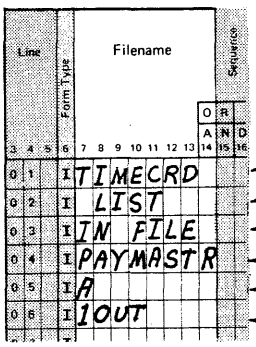
You must describe, on a separate line, every file used in your program. Many simple programs require only one input file and one output file. In the first programs we discuss, therefore, we use only one input file and one output file.

File Names

Every file used in a program must be named. The name provides you and the compiler a means of identifying the file. During compilation, the compiler associates the file name with other characteristics of the file. Thus, you can refer to that file by name throughout your program, and the compiler knows exactly which file you are referring to.

However, the compiler recognizes file names only if they conform to these rules:

- A file name must be from 1 to 8 characters long.
- The first character of a file name must be alphabetic. The remaining characters in the name can be either alphabetic or numeric.
- Blanks must not appear between characters in the file name.
- No two files used in the same program can have the same name. (Because some RPG compilers use only the first 7 characters of an 8-character file name, be certain, when using these systems, to make the first 7 characters unique; for example, use TRANSACT and TRANFILE, not TRANFILA and TRANFILB.)
- The file name must begin in position 7 on the specifications form.



← Valid file name.
← Invalid file name. Name must start in position 7.
← Invalid file name. A blank must not be used between characters.
← Valid file name.
← Valid file name.
← Invalid file name. The first character of the name must be an alphabetic character.

It is a good practice to assign meaningful file names. Meaningful names indicate something about the file, such as the type of records in the file or the use of the file. Because file names can be no longer than 8 characters, abbreviations may be necessary. But these, too, can be meaningful. For example, the abbreviation CUSTCHG might be assigned to an input file consisting of records for all customers having charge accounts.

Device Designation

You must also specify which devices your program uses for input and output. The ones you use, of course, depend on the system you have, the devices you have, and your program.

To indicate the device used for the file you named, enter the RPG code name for that device in positions 40 through 46. The name must begin in position 40.

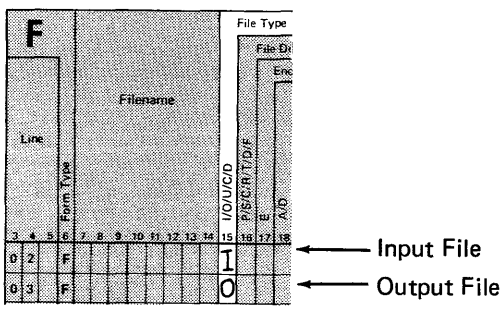
Note: The examples in this manual use shading in the device name positions rather than actual RPG code names for devices because code names differ for each system. The RPG reference manual for your system tells you which code names are applicable.

During compilation, the compiler associates the file name with the device name. When you use the same file name in the rest of your program, the system knows which device to use.

File Use

You must also describe how each file and its associated device is used in a program. Files can be used as either input or output. (Files with other uses are not discussed in this manual.) If records are read from a file, the file is an input file. If a new file is created during the program, the new file is an output file. Printed reports are the only output files discussed in this manual.

You specify file use by placing either an I (input) or O (output) in position 15:

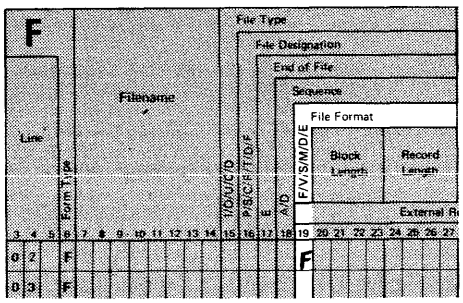


File Designation

Position 16 is used to explain more about the use of input files. In this manual we are discussing the use of only one type of input file, the primary file. For this, you place a P in position 16. Designating a file as a primary file in an RPG program indicates that the RPG program cycle supplies some of the program logic. For more information on primary file and other types of input files, see the RPG reference manual for your system.

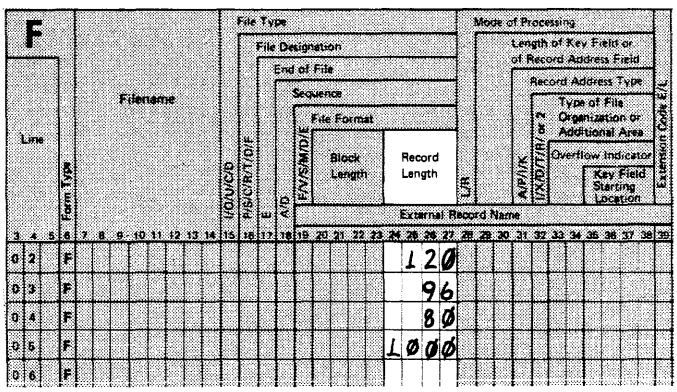
File Format

Position 19 is used to identify the format of records in the file and to indicate whether the records are defined in the RPG source program (on input and output specifications) or defined externally. This manual discusses only fixed-length program described records, which require an F in position 19. For information on other entries, see the RPG reference manual for your system.



Record Size

When describing files, you must specify the length (in characters) of records in the file. Record length is entered in positions 24 through 27. Enter the length so the last digit is in position 27:



The record length specification does two things:

- It tells the compiler how much storage space to set aside for a record (input or output).
- It specifies how many characters must be read to get a complete input record.

Record size for card files is easy to determine. It is either 80 or 96, depending on the size of cards you have. Maybe not all your cards have information in all columns, but all columns must be read to get an entire record. Blanks are placed in storage positions corresponding to unpunched card columns.

The size of records (lines) on the printer is also easy to determine. Printed records are limited by the size of your printer (the number of print positions in a line).

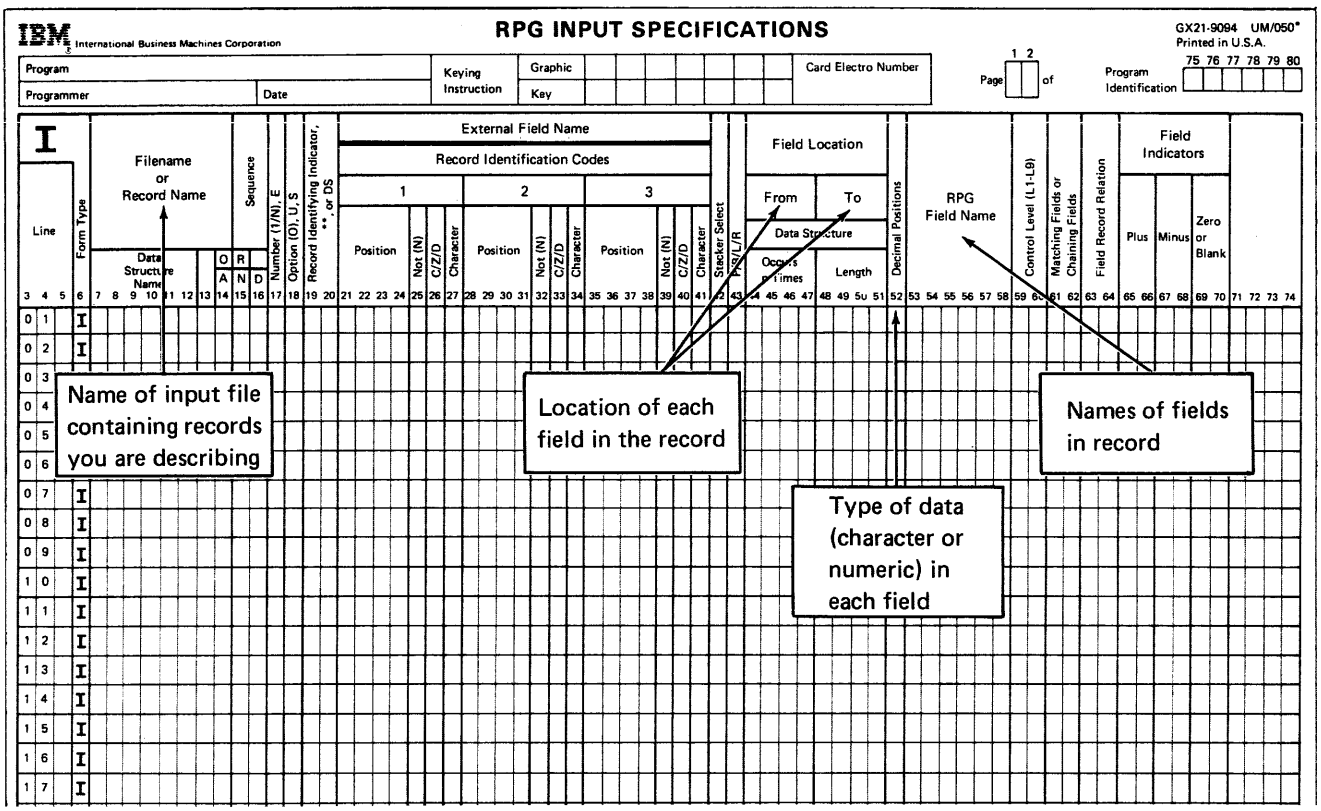
You may, if you wish, specify a record size smaller than actual printer size. If you do this, make certain that none of the lines to be printed are longer than the length you specify; otherwise the RPG compiler will give you an error message that requires that the program be corrected and recompiled.

Records on other files, such as disk, can be any size. The maximum size is limited only by the capability of the device. When you use one of these files for a program, make sure you enter the correct record size; that is, enter the one established at the time the file was created.

Other information may be required to describe how the input file is stored on the specific device in positions 29 through 38. See the RPG reference manual for your system for these entries.

DESCRIBING INPUT RECORDS

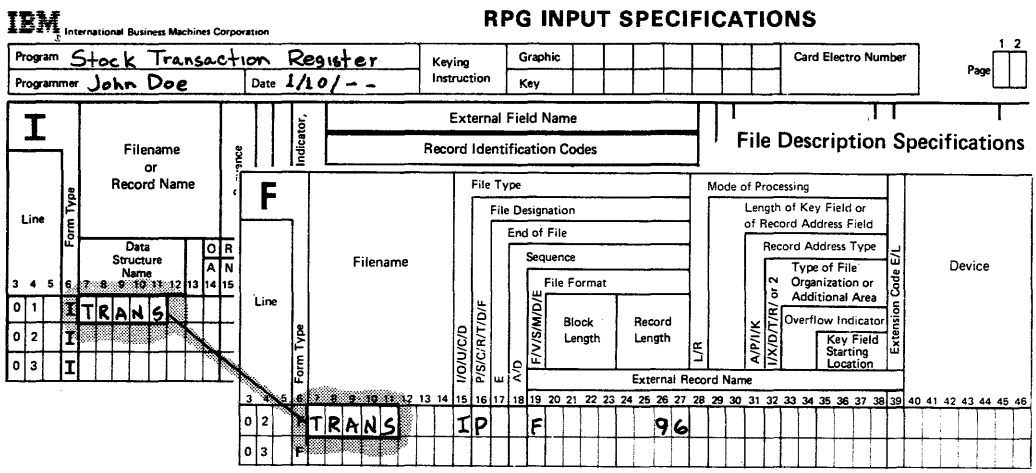
Besides describing files, you must describe the records in each file. The compiler needs this information to create an object program that will read records properly. Input records are described on the Input Specifications form. A description of the record in a file includes the name of the file containing the record, the name of each field in the record, and where each field is located in the record. To describe fields and data in the record, fill in the indicated postions:



To help in describing input records, you can use the record layout form described in Chapter 1. That form shows the location and length of all fields in the record.

File Names

To tell the compiler which records you are describing, enter in positions 7 through 14 the name of the file containing the records. The name must be the same (and spelled exactly the same) as the one you assigned to the input file on the File Description Specifications form:

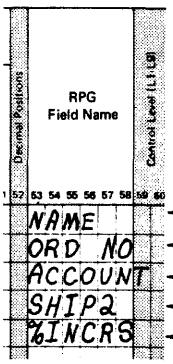


Field Names

To identify individual fields in the record, you must give each field a unique name. From information you placed on the File Description Specifications form the compiler determines the size of the storage area for each input record. The field names you supply on the Input Specifications form tell the compiler to divide this storage area into smaller sections so each can be addressed separately.

The rules for forming field names are as follows:

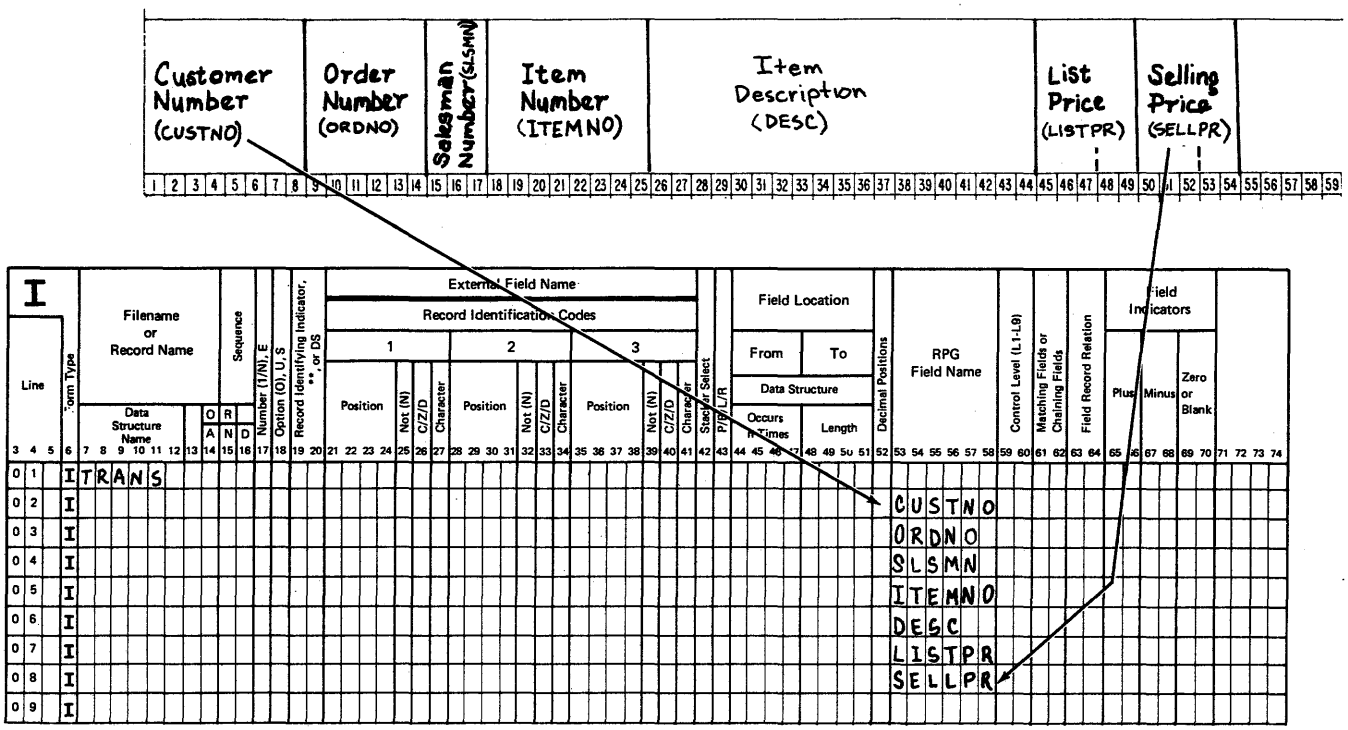
- The field name must be from 1 to 6 characters long.
- The first character must be alphabetic. Remaining characters can be either alphabetic or numeric.
- Blanks must not be placed between characters in a field name.
- The field name must begin in position 53 on the Input Specifications form.



←Valid field name.
←Invalid field name. A blank cannot be used between characters in the name.
←Invalid field name. A field name can be no longer than 6 characters.
←Valid field name.
←Invalid field name. The name must start with an alphabetic character.

It is a good practice to assign meaningful field names. For example, a field containing customer numbers would be more meaningful if it were called CUSTNO rather than FIELDA. CUSTNO indicates something about the data in the field.

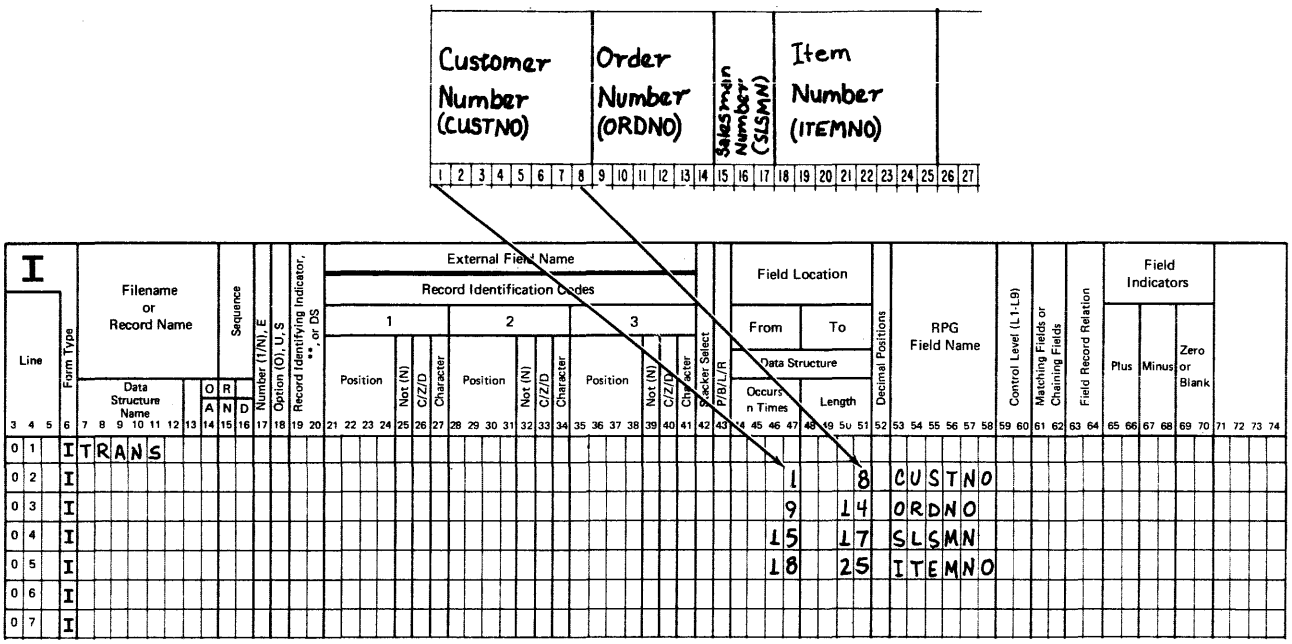
Enter field names one line below the file name, using a separate line for each field:



Be sure to name every field that contains information necessary for your program. If you need all fields on the record, name them all; if you need only a few, name only those you will need. The entire record is read, of course, regardless of how many fields you are using from that record. However, only information in the fields you name can be used by the program.

Field Location

After you assign a field name, you must tell where the field is located in the record. This enables the compiler to associate the field name with the right information. To describe the field location, you specify the position in the record at which the field begins and ends. Starting position is specified in positions 44 through 47 (From); ending position is specified in positions 48 through 51 (To). When a field is only one position, starting and ending position entries are the same. Field location entries can be easily determined from the record layout form:

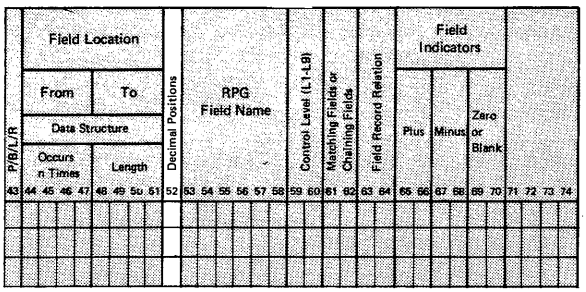


The compiler also determines field length from the To and From entries. The compiler needs field length to determine how many storage positions to allow for each field. If the field locations you specify indicate a field length of 6, the compiler allows six positions in storage for the field.

Type of Data

To complete your description of the input fields, the compiler checks position 52. This position indicates whether data in each field is character or numeric. A numeric field can contain only numbers; a character field can contain numbers, letters, and special characters.

If position 52 is blank, the compiler assumes the field is character. For numeric fields, position 52 must contain an entry. This entry indicates the number of decimal positions (digits to the right of the decimal point) in the field.



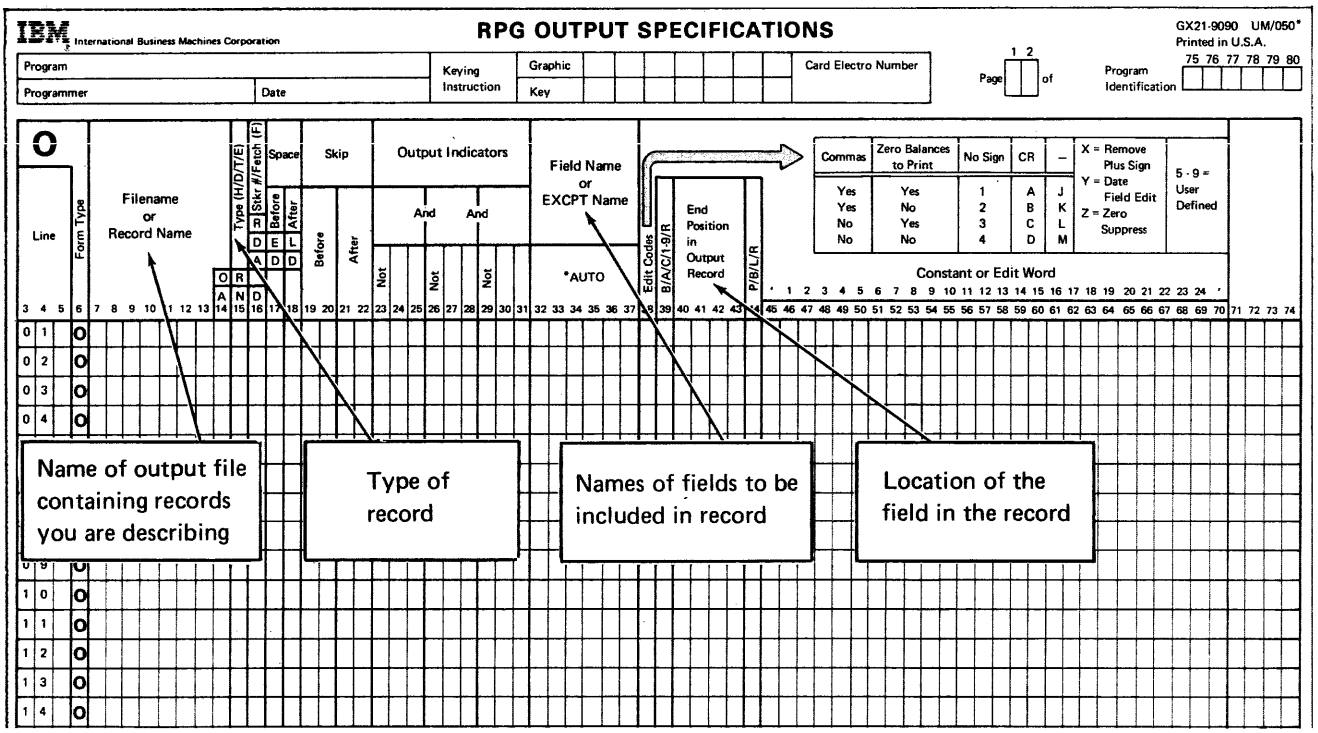
For character fields, leave position 52 blank. For numeric fields, enter in position 52 a number 0 through 9 to indicate the number of decimal positions in the field.

Although you do not include decimal points in fields in input records, you must consider them if you want correct calculation results and output data. By specifying to the compiler the number of decimal positions you know to be in a numeric field, you provide the information necessary to produce an object program that will handle numeric data with decimals.

Remember, any field used in an arithmetic operation (add, subtract, multiply, or divide) must be specified as numeric.

DESCRIBING OUTPUT RECORDS

Output records are described on the Output Specifications form. Information needed includes the name of the file containing the output record, the name of each field in the record, and the location where each field is to be placed in the record. To describe how the output records should look, fill in the indicated positions:

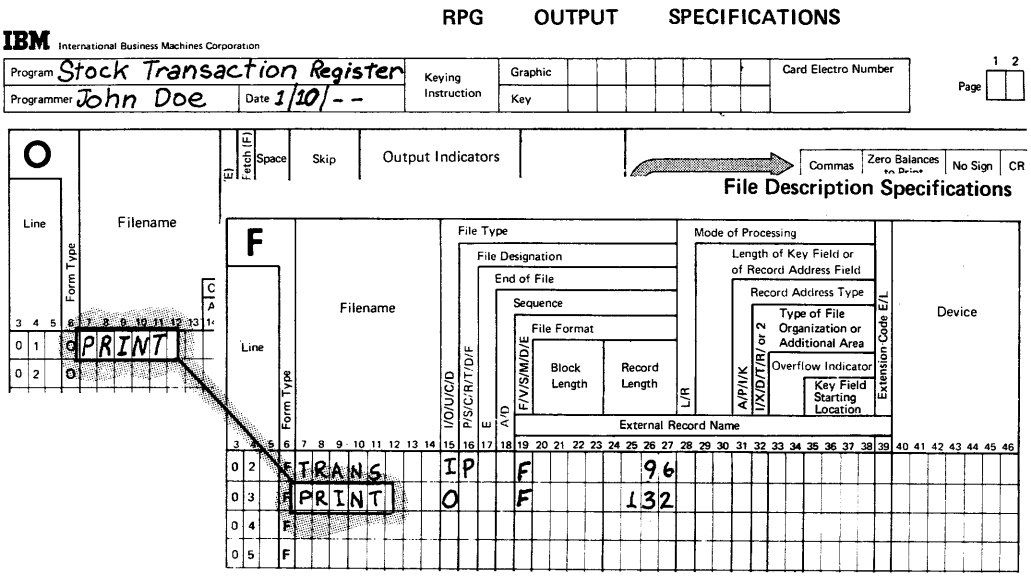


For a printed report, you make additional entries describing the format of the report; that is, entries indicating the spacing and punctuation you want. These entries are discussed later under *Printed Reports*.

The printer spacing chart and record layout forms are useful when you are writing output specifications. The record layout form shows the organization of fields on a card or disk record; the printer spacing chart shows the format of printed records.

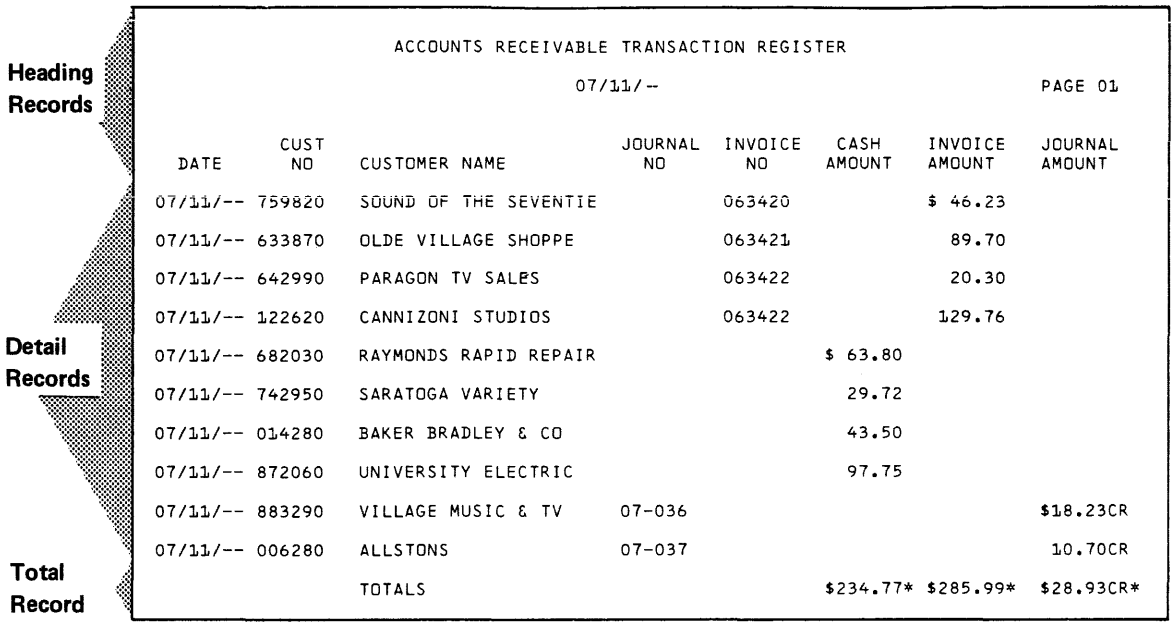
File Names

You indicate the output record you want created and the device you want to create the record by entering an output file name in positions 7 through 14. Make sure the name you enter is the same (and spelled exactly the same) as the name you entered on the File Description Specifications form for the output file:



Record Type

Three different types of records that can be specified on the Output Specifications form are heading, detail, and total. You usually find all three types in a printed report.



ACCOUNTS RECEIVABLE TRANSACTION REGISTER

07/11/-- PAGE 01

DATE	CUST NO	CUSTOMER NAME	JOURNAL NO	INVOICE NO	CASH AMOUNT	INVOICE AMOUNT	JOURNAL AMOUNT
07/11/--	759820	SOUND OF THE SEVENTIE		063420		$ 46.23	
07/11/--	633870	OLDE VILLAGE SHOPPE		063421		89.70	
07/11/--	642990	PARAGON TV SALES		063422		20.30	
07/11/--	122620	CANNIZONI STUDIOS		063422		129.76	
07/11/--	682030	RAYMONDS RAPID REPAIR			$ 63.80		
07/11/--	742950	SARATOGA VARIETY			29.72		
07/11/--	014280	BAKER BRADLEY & CO			43.50		
07/11/--	872060	UNIVERSITY ELECTRIC			97.75		
07/11/--	883290	VILLAGE MUSIC & TV	07-036				$18.23CR
07/11/--	006280	ALLSTONS	07-037				10.70CR
		TOTALS			$234.77*	$285.99*	$28.93CR*

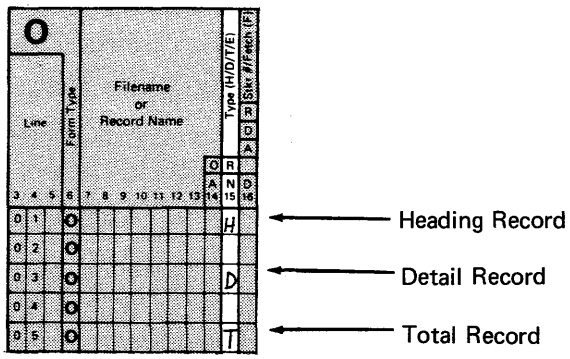
Heading records are printed at the top of a page. They include report titles, column headings, or any other information needed to identify the kinds of information found in the report.

Detail records contain information about an individual item. Information in a detail record is often taken directly from an input record.

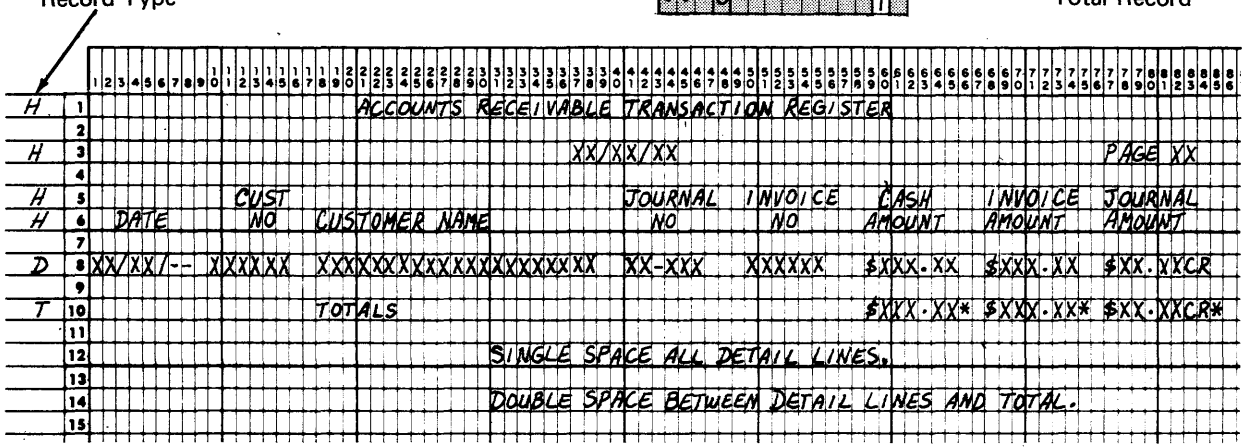
Total records are written after a group of detail records. They usually contain data that is the result of calculations on information in a group of detail records.

To specify record type, place an appropriate entry in position 15. If the printer spacing chart was properly filled out, you can refer to it to determine record type:

These entries specify the record type:

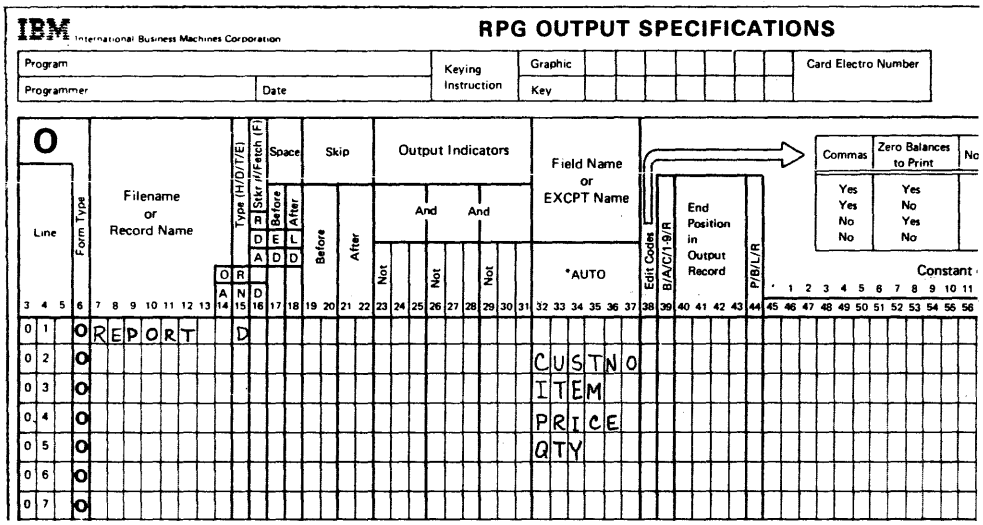


Refer to the printer spacing chart to determine the record type:



Field Names

To specify the information to be placed in each output record, you must name each field to be included. You specify these fields on separate lines of the Output Specifications form in positions 32 through 37. Begin the list of fields one line below the file name:

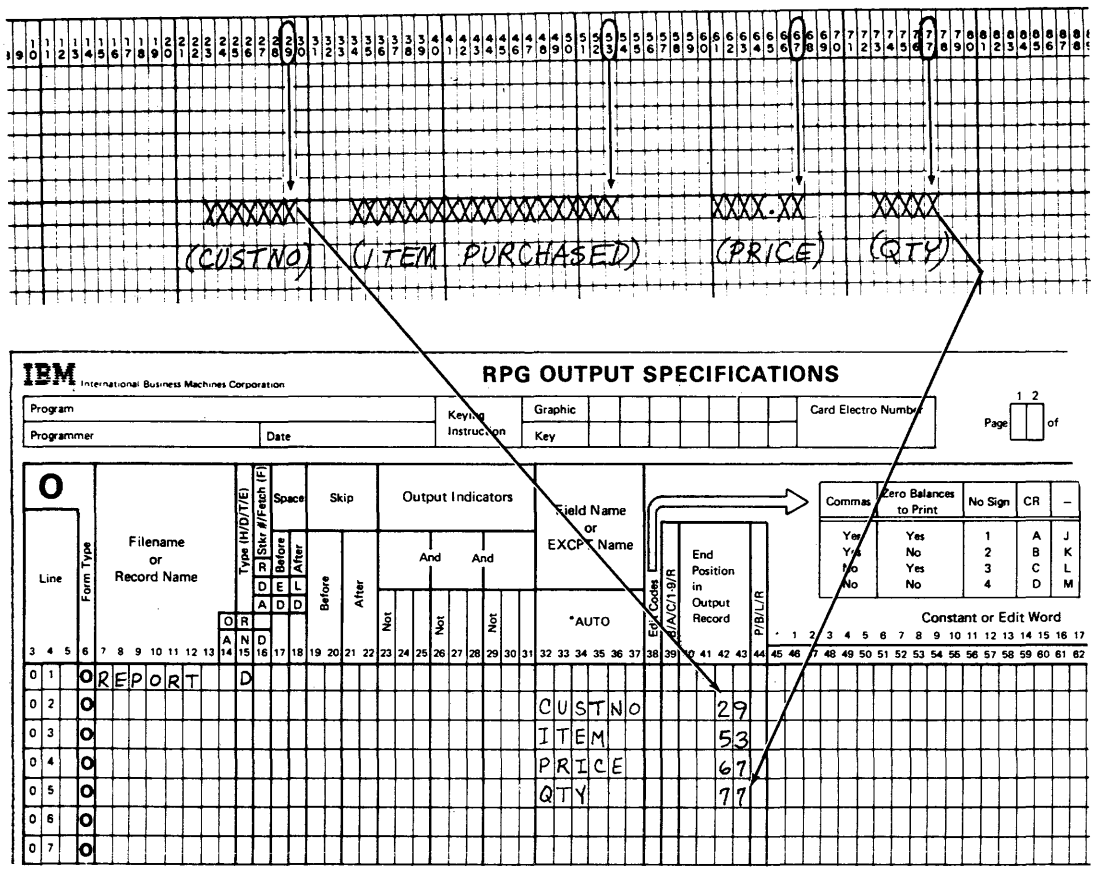


When listing the fields, make sure you enter a name that was previously given to a field (for example, a field named on the Input Specifications form). If the name you enter on the Output Specifications form is not one previously used, the compiler will not know what information you are referring to.

Field names are used to create the output record in the output storage area. Information is placed in the storage area one field at a time in the order you list them on the Output Specifications form.

Field Location

To specify where you want fields placed in the output records, you make an entry in positions 40 through 43 (End Position in Output Record). This entry must be the exact position where the last character in a field should be placed in the output storage area at the time the record is being created. The entry is easy to determine if you use a printer spacing chart or record layout form.



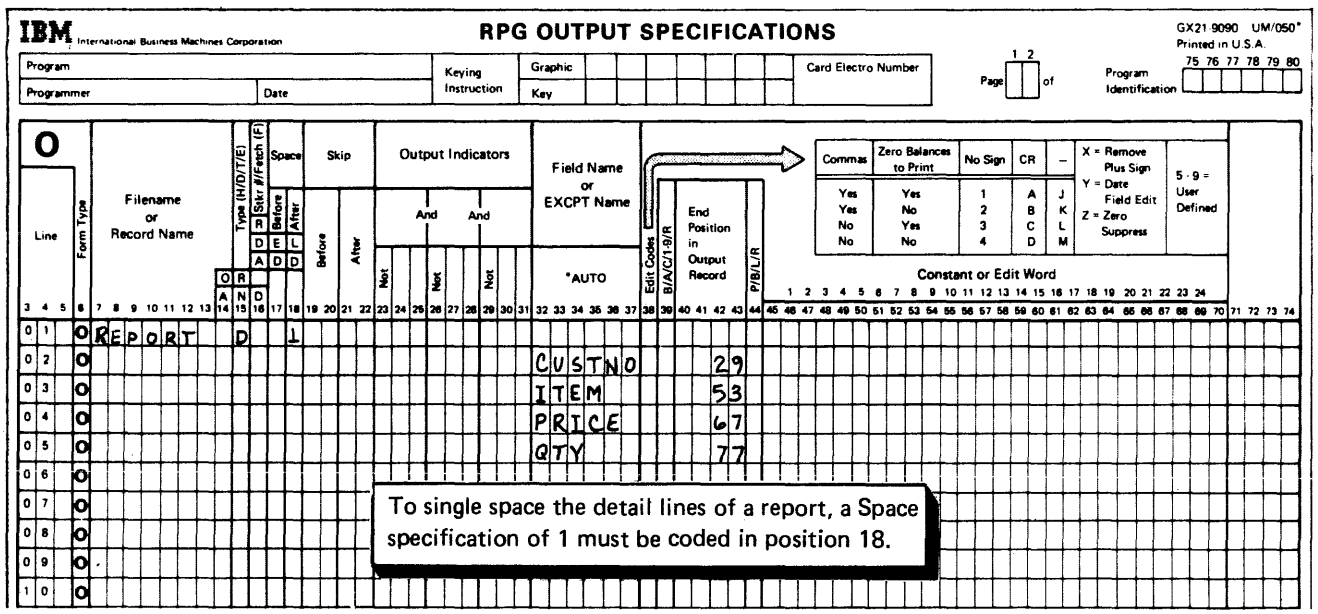
PRINTED REPORTS

When your output is a printed report, additional entries are needed on the Output Specifications form to make the report easy to read. Information must be neatly arranged in rows and columns with adequate space between items in a line and between lines.

Spacing

Your field location entries (positions 40 through 43) control space between fields, but to control spaces between lines you code positions 17 and 18 (Space).

You can have the printer single, double, or triple space between lines by entering the number 1, 2, or 3 in the appropriate positions. If you enter the number in Space Before (position 17), the printer spaces before printing the line; if you enter the number in Space After (position 18), the printer spaces after printing the line. You may enter numbers in both positions 17 and 18. By specifying three spaces before printing and three spaces after, you can have five blank lines between printed lines. The printer spaces three lines after printing a line, then spaces three lines before printing the next line. This next line is then printed six lines beyond the last line printed.

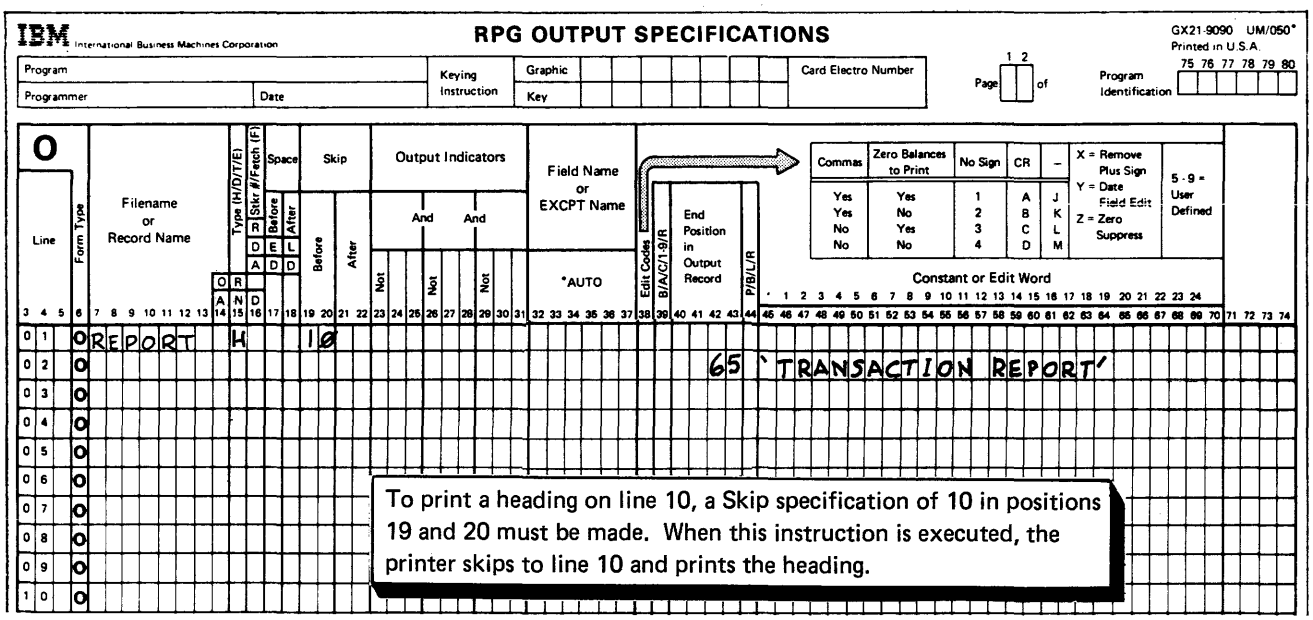


Skipping

You can use Skip entries in positions 19 through 22 to control spacing between lines on a page and to control printing the first and last lines on a page. A skip can be made before or after a line is printed. You indicate this by coding the skip in either positions 19 and 20 (Before) or 21 and 22 (After). The entry you place in these positions depends on the type of printer you have.

In this manual, we discuss and show examples only for printers with tapeless carriage control. If your system uses a carriage control tape, you will find the appropriate entries for skipping in the RPG reference manual for your system.

The Skip entry for a printer with tapeless carriage control can be any line number (1 through 112) on the printer paper. (The standard 11-inch printer paper has 66 lines per page when six lines are printed per inch.) A Skip entry in positions 19 and 20 indicates the line to which the printer skips before it prints the next line.



Printer paper is not rolled backward. If the printer is on line 50 when the program issues a skip instruction to line 10, the printer skips to line 10 on the next page.

Editing

Editing is a means of punctuating numeric fields by adding decimal points or commas or by indicating negative values (by a minus sign or a CR). Editing can also suppress the printing of leading xeros (in the number 00149, 00 are called leading zeros).

When a numeric field is read into storage, it contains no decimal point or commas; when an unedited numeric field is printed, it appears exactly as it is in storage. A large number printed without commas or decimals is hard to read.

Furthermore, an unedited field may not be meaningful when printed because of the way the system stores negative numbers. The system uses the last digit in a numeric field to indicate sign (plus or minus). If the field is negative, the system combines a minus sign with the last digit. When a negative number is printed unedited, the combination of digit and sign appears as a letter. For example, minus 6439 prints as 643R. On the other hand, a positive field has no sign (a numeric field that does not have a minus sign is assumed to be positive). A positive field, therefore, prints normally. Positive 6439 prints as 6439.

The compiler can provide instructions to edit in a number of ways. All you have to do is enter an edit code in position 38 of the Output Specifications form. Many codes are available, each indicating a different type of editing. Figure 8 shows the codes and the editing done for each. Figure 9 shows some examples of editing.

Note: When you edit a field, you often add characters to it. When printed, the edited fields require more space than they did on input records or in storage. When specifying the end position for an edited field, always take into account the spaces needed for the punctuation that will be added. The printer spacing chart shows the amount of space needed for the edited field.

Edit Code	Commas	Decimal Point	Sign For Negative Balance			Zero Suppress	Print Out On Zero Balance
			No Sign	CR	– (Minus)		
1	Yes	Yes	No Sign			Yes	.00 or 0
2	Yes	Yes	No Sign			Yes	Blanks
3		Yes	No Sign			Yes	.00 or 0
4		Yes	No Sign			Yes	Blanks
A	Yes	Yes		CR		Yes	.00 or 0
B	Yes	Yes		CR		Yes	Blanks
C		Yes		CR		Yes	.00 or 0
D		Yes		CR		Yes	Blanks
J	Yes	Yes			–	Yes	.00 or 0
K	Yes	Yes			–	Yes	Blanks
L		Yes			–	Yes	.00 or 0
M		Yes			–	Yes	Blanks
X[1]							
Y[2]						Yes	
Z[3]						Yes	

[1] The X code removes the plus sign of the field.

[2] The Y code is used for date fields. It suppresses only the leftmost zero and puts slashes in a three-digit to six-digit field according to the following pattern:

nn/n
nn/nn
nn/nn/n
nn/nn/nn

[3] The Z code removes signs and suppresses zeros.

Note: The edit codes shown in the first column are used in position 38 of the Output Specifications form to punctuate the field named on the same line. Only numeric fields can be edited. The decimal point is automatically inserted in the correct position.

Figure 8. Edit Codes

Edit Codes	Field Length and Digits	1769532	02	00	000	041345
	Field Characteristics	Positive Number with Two Decimal Positions	Negative Number with Two Decimal Positions	Zero with Two Decimal Positions	Zero with No Decimal Positions	Positive Number with Three Decimal Positions
1		17,695.32	.02	.00	0	41.345
2		17,695.32	.02			41.345
3		17695.32	.02	.00	0	41.345
4		17695.32	.02			41.345
A		17,695.32	.02CR	.00	0	41.345
B		17,695.32	.02CR			41.345
C		17695.32	.02CR	.00	0	41.345
D		17695.32	.02CR			41.345
J		17,695.32	.02-	.00	0	41.345
K		17,695.32	.02-			41.345
L		17695.32	.02-	.00	0	41.345
M		17695.32	.02-			41.345
X		1769532	0K	00	000	041345
Y		Must be used with a three-digit to six-digit field.			0/0	4/13/45
Z		1769532	2			41345

Note: This table shows the effect of editing on five different fields. It illustrates what will be printed by using each edit code on the fields.

Figure 9. Examples of Editing

EXAMPLE 1 (TRNREG): PRINTING A SIMPLE REPORT USING THE THREE BASIC CYCLE OPERATIONS

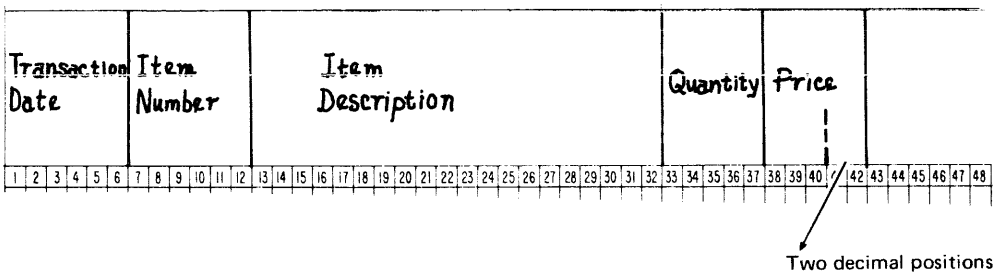
Program Definition

Print a report listing all items sold during a week. The selling of an item is known as a transaction, so the report is titled *Transaction Register*.

During the week, a transaction file is created. At the end of each day, transaction records are created from information obtained from order forms received during the day. To get the printed transaction report, you list the information from all input records on the printed report.

Program Requirements

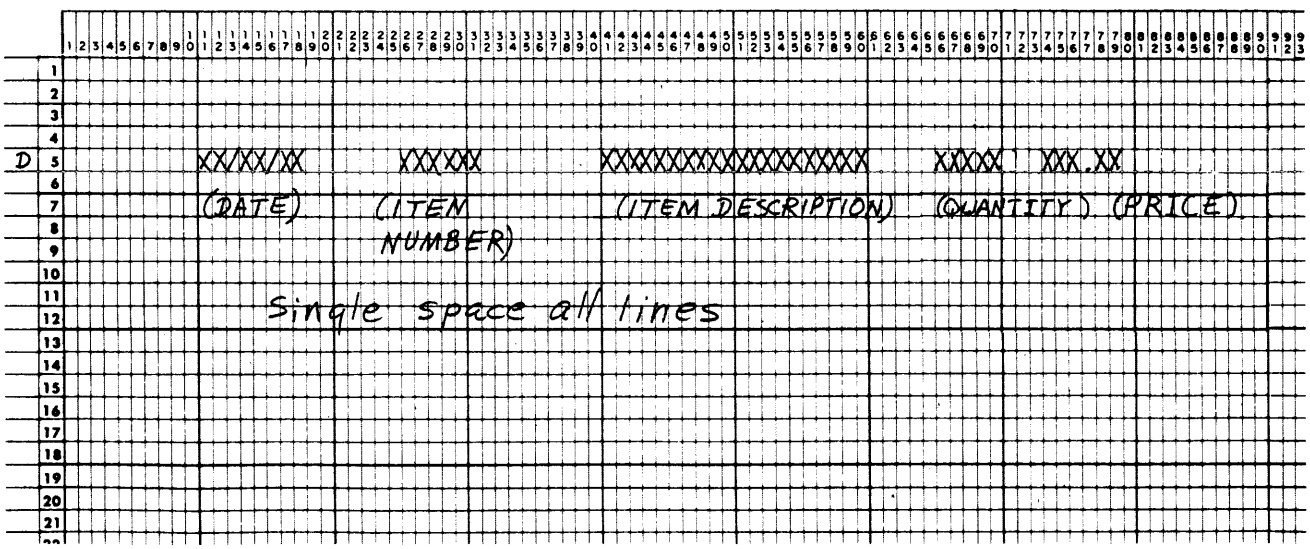
Input: Sales transaction file consisting of 96-character records. The format of the input records is shown on this record layout form:



Output: A printed transaction register:

7/23/--	413010	CH001 BOX 100A FLUSH	10	4.90
7/23/--	412146	CH148 BREAKER 15A	100	.89
7/23/--	411116	1500 TWIN SOCKET B	500	1.12
7/24/--	503029	MOTOR 1/2 HP 60 CYC	2	146.78
7/24/--	317802	TERMINAL CLIP	100	5.12
7/24/--	326917	TERMINAL BAR	100	4.12
7/24/--	411121	1506 SOCKT ADAPT BRN	400	.19
7/24/--	412997	CH173 BREAKER 30A	60	1.15
7/24/--	413088	CH176 BREAKER 60A	40	1.15
7/24/--	411174	C151 SIL SWITCH BRN	200	1.16
7/24/--	413090	CH005 BR BOX 150A	10	4.98
7/24/--	718326	FC803 FUSE 15A	200	.32

This printer spacing chart shows how the report is formatted:



RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9092 UM:050*
Printed in U.S.A.

Program	Transaction Register	Keying Instruction	Graphic								Card Electro Number	Page 01 of __	Program Identification TRNREG
Programmer	John Doe	Date 1/10/--		Key									

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	Reserved	Currency Symbol	Date Format	Date Edit	Inverted Print	Reserved	Number of Print Positior
01	H												

File Description specifications describe the files used: one input and one output. The input file, consisting of 96-character transaction records, is given the name TRANS. I in position 15 indicates input file; P in position 16 indicates primary file; F in position 19 indicates program described file; 96 in positions 24 through 27 tells how long the records are. The device used to read the file depends on your system. The output file is a printer. The file name is REPORT. The output records are also program described. The print line (record length) is 96 positions long.

Notice that entries have been made in positions 1 and 2 and in positions 75 through 80 in the upper right corner of each form. The page number (positions 1 and 2) and the line numbers (positions 3 through 5) help you keep your specifications in order. These numbers are keyed in the source program. Positions 75 through 80 identify your program.

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D/F	E	A/D	F/V/S/M/D/E	Block Length	External Record Name	Option	Entry
02	F	TRANS	I	P			F	96			
03	F	REPORT	O				F	96	PRINTER		

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

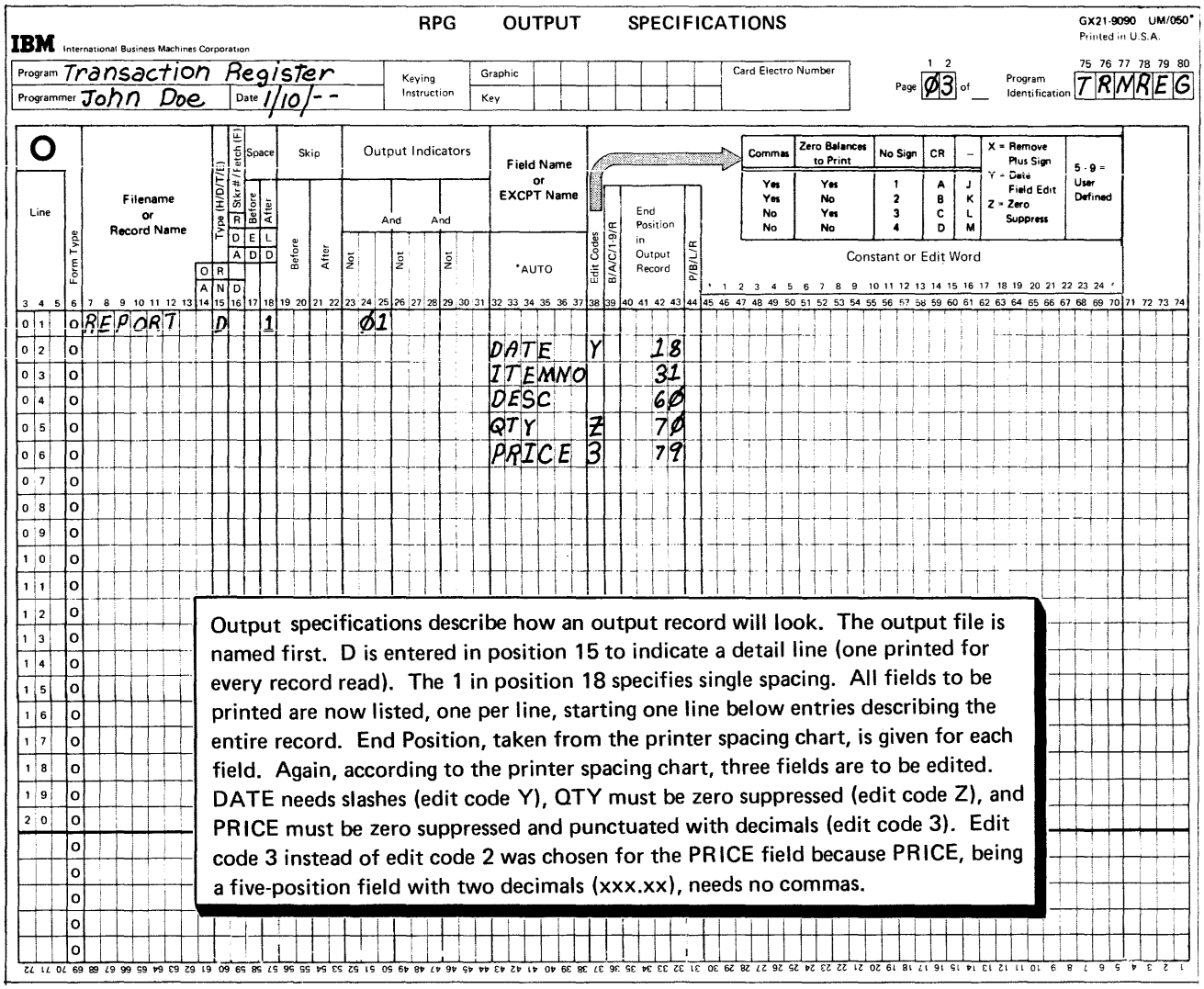
GX21-9094 UM/050*
Printed in U.S.A.

Program	Transaction Register	Keying Instruction	Graphic								Card Electro Number	Page 02 of	Program Identification TRNREG
Programmer	John Doe	Date 1/10/--		Key									

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / 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	Data Structure: Occurs n Times	Data Structure: Length	Decimal Positions	RPG 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	TRANS				01															1	6			0	DATE						
02	I																				7	12				ITEMNO						
03	I																				13	32				DESC						
																					33	37			0	QTY						
																					38	42			2	PRICE						

Input specifications describe the input records. The input file name must be the same as the name given to the input file on the File Description Specifications form. Fields on the input records are then described. Most of the information for describing fields is taken from the record layout form. The record contains five fields, all of which are needed for output. Therefore, all five fields are described, starting one line below the file name. Field description entries include field location, field name, and decimal position, which indicates type of data (character or numeric). Any field to be used in arithmetic operations or edited must be numeric. The Output Specifications (page 40) show that three fields are edited. Therefore, on the Input Specifications three fields have an entry in position 52 to indicate numeric fields.

Note: The record identifying indicator (entry 01 in positions 19 and 20 of the Input Specifications) is needed to get correct output. It will be explained later under *Example 5 (STOKST): Using Record Identifying Indicators to Process Different Record Types.*



Output specifications describe how an output record will look. The output file is named first. D is entered in position 15 to indicate a detail line (one printed for every record read). The 1 in position 18 specifies single spacing. All fields to be printed are now listed, one per line, starting one line below entries describing the entire record. End Position, taken from the printer spacing chart, is given for each field. Again, according to the printer spacing chart, three fields are to be edited. DATE needs slashes (edit code Y), QTY must be zero suppressed (edit code Z), and PRICE must be zero suppressed and punctuated with decimals (edit code 3). Edit code 3 instead of edit code 2 was chosen for the PRICE field because PRICE, being a five-position field with two decimals (xxx.xx), needs no commas.

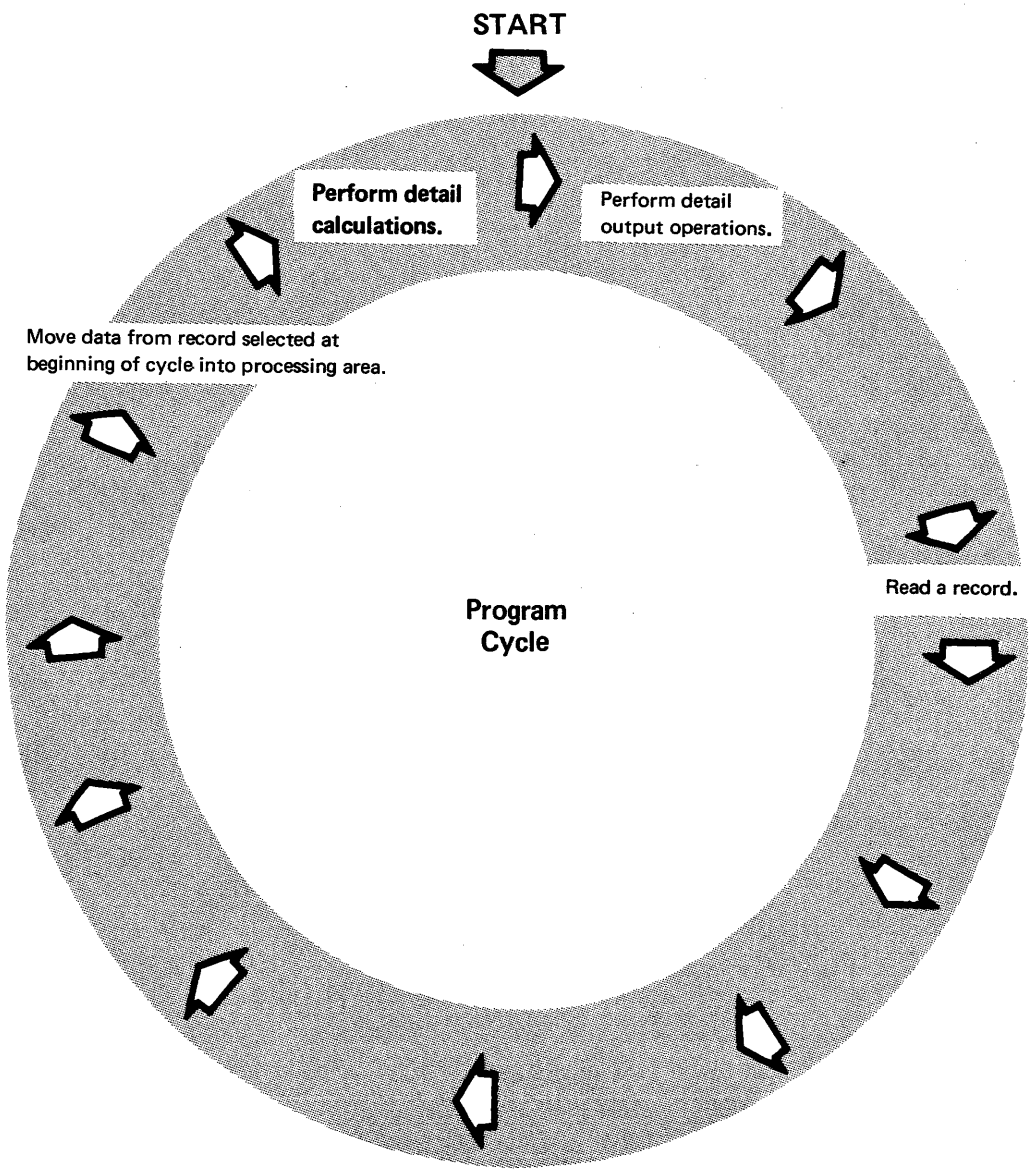
Note: The record identifying indicator (entry 01 in positions 24 and 25) is needed to get correct output. It will be explained later under *Example 5 (STOKST): Using Record Identifying Indicators to Process Different Record Types.*

Writing Specifications for Calculation Operations

Most programs require some processing. In RPG, processing can include calculating, comparing, moving, or changing data. In this discussion we consider only calculating arithmetic results (adding, subtracting, multiplying, and dividing).

Program Cycle Operations

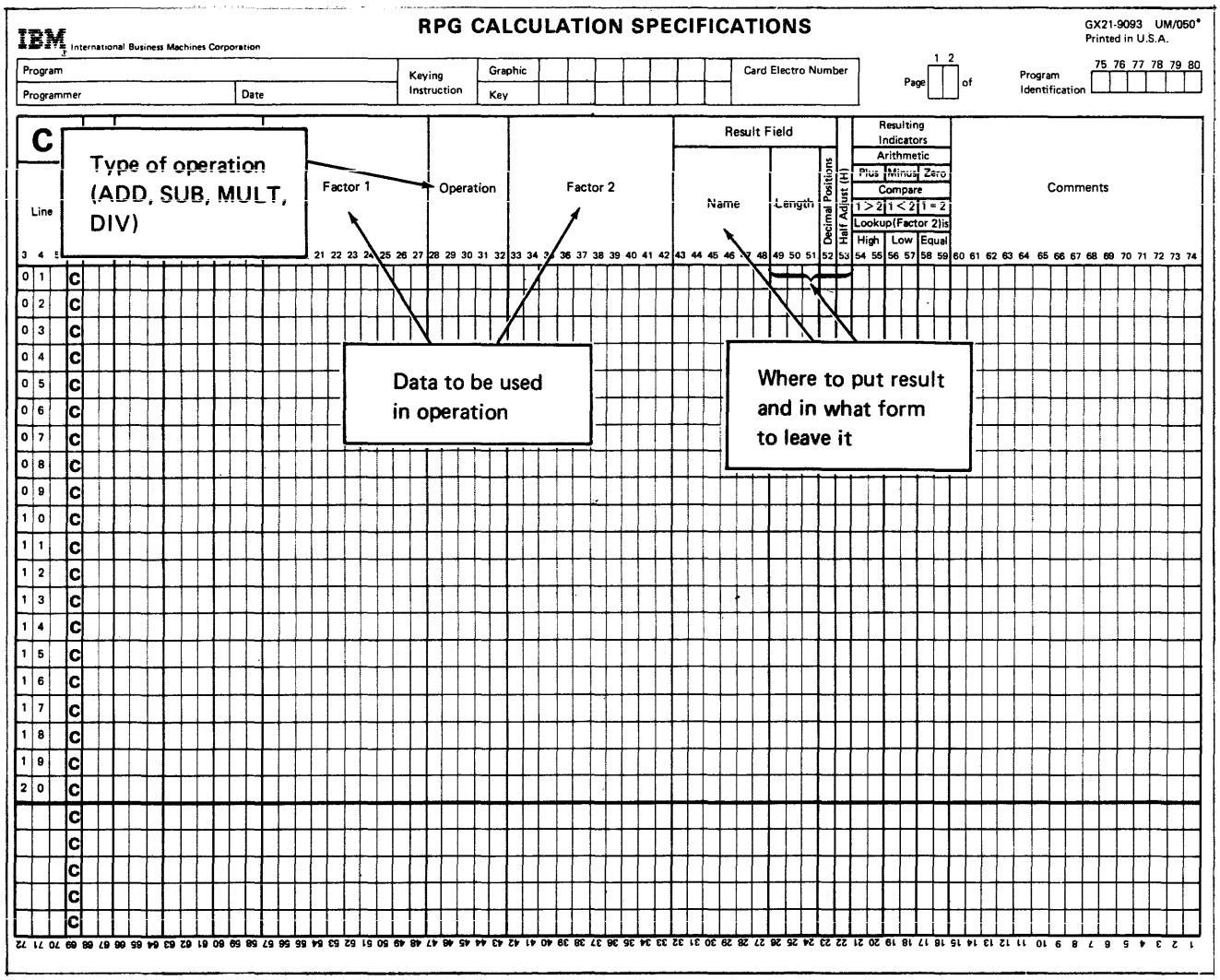
When you specify a calculation operation related to each input record processed, you are adding one more operation to the basic program cycle: the detail calculation operation (see Figure 10).



Note: This is a basic program cycle showing the addition of detail operations. Because this is a detail operation, it is performed during every cycle for every record read.

Figure 10. Program Cycle for Detail Calculations

The Calculation Specifications form is used to describe the operations you want performed. Information needed includes the type of operation to be done, the field(s) or constant to be used in the calculation, and the location where the result of the calculation is to be placed. You fill in the indicated positions:



Specify one operation per line. In each program cycle, processing steps are done in the order you specified on this form. If calculations must be done in a particular order, you must list the operations in that order.

DESCRIBING TYPE OF OPERATION

To indicate the type of calculation operation, you enter one of the following operation codes in positions 28 through 32 on the Calculation Specifications form:

ADD (add)

SUB (subtract)

MULT (multiply)

DIV (divide)

DESCRIBING DATA TO BE USED

After you have specified the type of operation, you must identify the data to be used. If you specified ADD, for example, you must tell the system what to add. You do this by naming the fields to be used in positions 18 through 27 (Factor 1) and 33 through 42 (Factor 2).

Instead of naming a field in Factor 1 or Factor 2, you can enter a *constant*; that is, you can enter actual data instead of the name of a field containing the data. For example, you can enter either of the following:

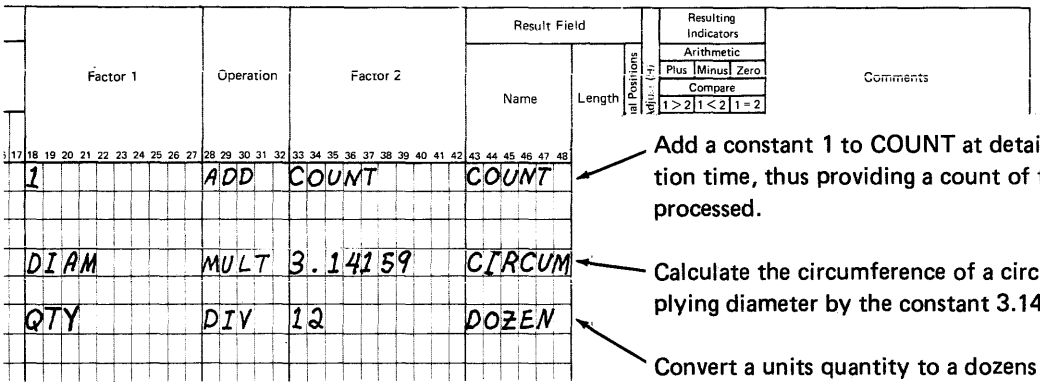
500 Constant (actual data)

AMOUNT Name of a field containing data

Constants can be either numeric or character, but here we discuss only numeric constants. The rules for using numeric constants are as follows:

- Constants can be up to ten numeric digits (0 through 9).
- Constants can have a sign and decimal point. The sign, if used, must be the leftmost character. The decimal point, if used, must be shown as part of the constant (for example, 4.12).
- The first character of the constant must be placed in the leftmost position of the Factor field.
- Constants cannot contain blanks.

The contents of a field can change during execution of a program, but constants do not. If you want to add, multiply, subtract, or divide the same number during every program cycle, you can use a constant:



Add a constant 1 to COUNT at detail calculation time, thus providing a count of the records processed.

Calculate the circumference of a circle by multiplying diameter by the constant 3.14159.

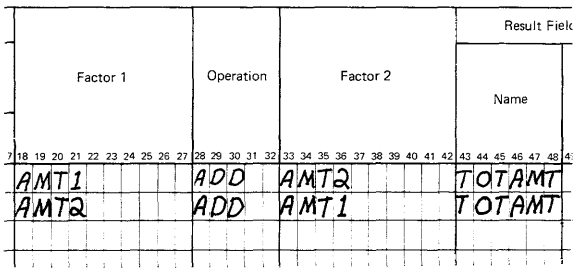
Convert a units quantity to a dozens quantity by dividing QTY by constant 12.

To the compiler, a constant is like a field name. During compilation, the compiler checks Factor 1 and Factor 2 for constants. If there are any, the compiler assigns a storage location for the constant and instructs the computer to put the appropriate constant in that location at the beginning of program execution.

Be sure to consider which fields you enter in Factor 1 and Factor 2, because the specified operation might affect the result. An easy way to remember where to enter the fields is to mentally replace the operation code with the corresponding arithmetic symbol. Thus, a subtraction is Factor 1 — Factor 2; a division is Factor 1 ÷ Factor 2.

ADD

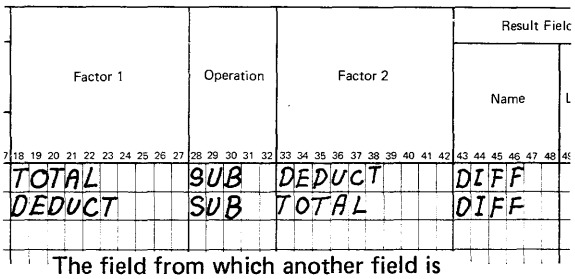
Factor 2 is added to Factor 1, and the sum is placed in the Result Field.



Either line adds the two amount fields. It makes no difference which amount field is entered in Factor 1 and which in Factor 2.

SUBTRACT

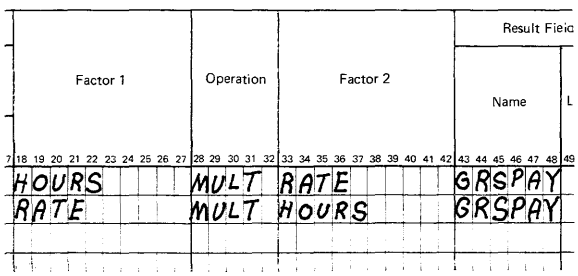
Factor 2 is subtracted from Factor 1, and the difference is placed in the Result Field.



The field from which another field is being subtracted must be entered in Factor 1. The field being subtracted must be entered in Factor 2. The bottom line will not produce the desired result.

MULTIPLY

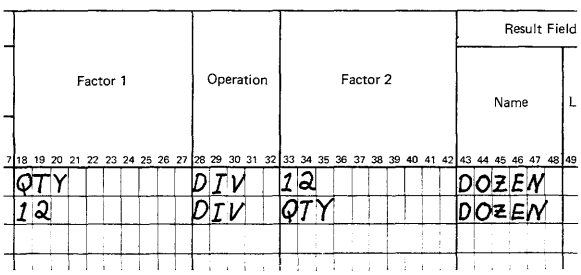
Factor 1 is multiplied by Factor 2, and the product is placed in the Result Field.



Either line multiplies the hours and rate to obtain the gross pay. It makes no difference which field is entered in Factor 1 and which in Factor 2.

DIVIDE

Factor 1 is divided by Factor 2, and the quotient is placed in the Result Field. Factor 2 cannot be zero.



The field being divided must be entered in Factor 1. The field by which Factor 1 is being divided must be entered in Factor 2. The bottom line will not convert a units quantity to dozens.

DESCRIBING THE RESULT FIELD

You must specify where you want the result of a calculation stored by naming that field in positions 43 through 48 (Result Field). The name you enter in the Result Field can be the name of a field already defined on the input or calculation specifications forms or a new field.

You would not need to name a new result field in these two situations:

- The contents of an input field are no longer required in your program, and the field is the correct size.
- The contents of an input field or a result field defined elsewhere on the Calculation Specifications form are to be replaced with a new value.

If you name a new field, you must specify field length (positions 49 through 51) and decimal position (position 52) so the compiler can assign adequate storage for the new field:

IBM RPG INPUT SPECIFICATIONS

Program	Graphic	Card Electro Number	Page
Programmer	Key		

Line	Form Type	Data Structure Name	Position	Not (N)	C/Z/D	Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Data Structure Occurs n Times	Data Structure Length	Decimal Positions	RPG Field Name
01	I	INFILE												
02	I								1	5				ITEMNO
03	I								6	20				DESC
04	I								21	25			0	QTY
05	I								25	30			2	COST
06	I													

RPG CALCULATION SPECIFICATIONS

	Keying Instruction	Graphic	Card Electro Number
		Key	

Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions
QTY	MULT	COST	COST		

The result field COST is not a new field because it is already defined by Input specifications. Field length and decimal position entries are not needed because the compiler already has this information and has set aside storage space for the field.

IBM RPG INPUT SPECIFICATIONS

Program	Graphic	Card Electro Number	Page
Programmer	Key		

Line	Form Type	Data Structure Name	Position	Not (N)	C/Z/D	Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Data Structure Occurs n Times	Data Structure Length	Decimal Positions	RPG Field Name
01	I	INFILE												
02	I								1	5				ITEMNO
03	I								6	20				DESC
04	I								21	25			0	QTY
05	I								25	30			2	COST
06	I													

RPG CALCULATION SPECIFICATIONS

	Keying Instruction	Graphic	Card Electro Number
		Key	

Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions
QTY	MULT	COST	TOTAL		2

The result field TOTAL is a new field because it is not defined on the Input Specifications form. Field length and decimal position entries are needed so that the compiler can set aside a storage area for this field.

Result Field Length

When you name a result field, make sure you specify one large enough to hold the results. Always consider the length of the fields involved in the operations. For example, if you are adding a two-position field to a three-position field, you must determine the largest result you could possibly have:

```
 999
  99
----
1098
```

Because there are four digits in this result, you would specify at least 4 as the result field length.

If this calculation occurred many times in your program, as in a running total, you would probably need a result field length larger than 4. It is up to you to determine the largest field length needed; failure to specify a large enough result field can mean a loss of data.

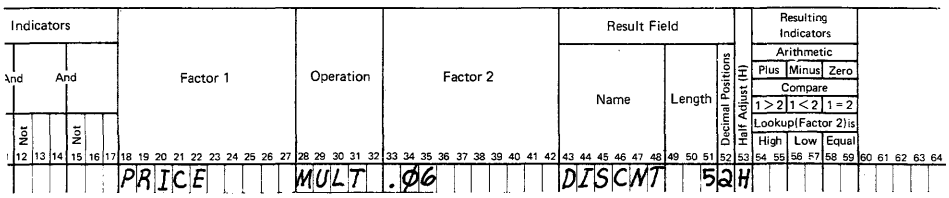
Decimal Positions

For a new result field, be certain to place an entry in position 52. If the new field is to be numeric but contains no digits to the right of the decimal point, enter a zero. Remember, this entry indicates type of field (numeric or character) as well as decimal positions. The result field of an arithmetic calculation must be specified as numeric by an entry in position 52.

Half-Adjusting Results (Rounding)

In RPG, rounding results is called *half-adjusting*. When the digit to the right of the last digit you want to keep is greater than 4, 1 is added to the last digit. The number 3.14159 rounded to four decimal positions becomes 3.1416. The same number rounded to two decimal positions is 3.14.

To half-adjust any calculation result, you place an H in position 53 of the Calculation Specifications form on the same line as the field to be half-adjusted:



In this example, DISCNT is half-adjusted. The entry in position 52 (Decimal Positions) indicates the number of decimal places to be retained after half-adjusting is completed. In this case, two decimal places are desired in the result. The multiplication and half-adjusting are done like this:

```
  74.98  <-- Assumed value of PRICE.
x  .06   <-- Constant representing 6% discount rate.
-------
4.4988   <-- Result that must be half-adjusted to two places.
   1     <-- 1 is added to 9 in the second decimal position because 8 in the
-------      third decimal position is greater than 4.
4.5088   <-- Slashed digits are dropped because only two decimal positions
             are required.
```

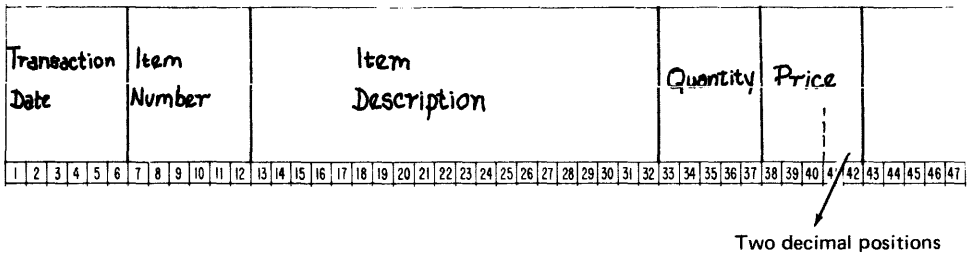
EXAMPLE 2 (TRNREG): DOING SIMPLE CALCULATIONS

Program Definition

Print a report listing all sales transactions for a week. This report is similar to the report created in Example 1. The only difference is the addition of the last column on the report, which is the sales amount per item. Sales amount (quantity sold times item price) is not found in the input record and must, therefore, be calculated.

Program Requirements

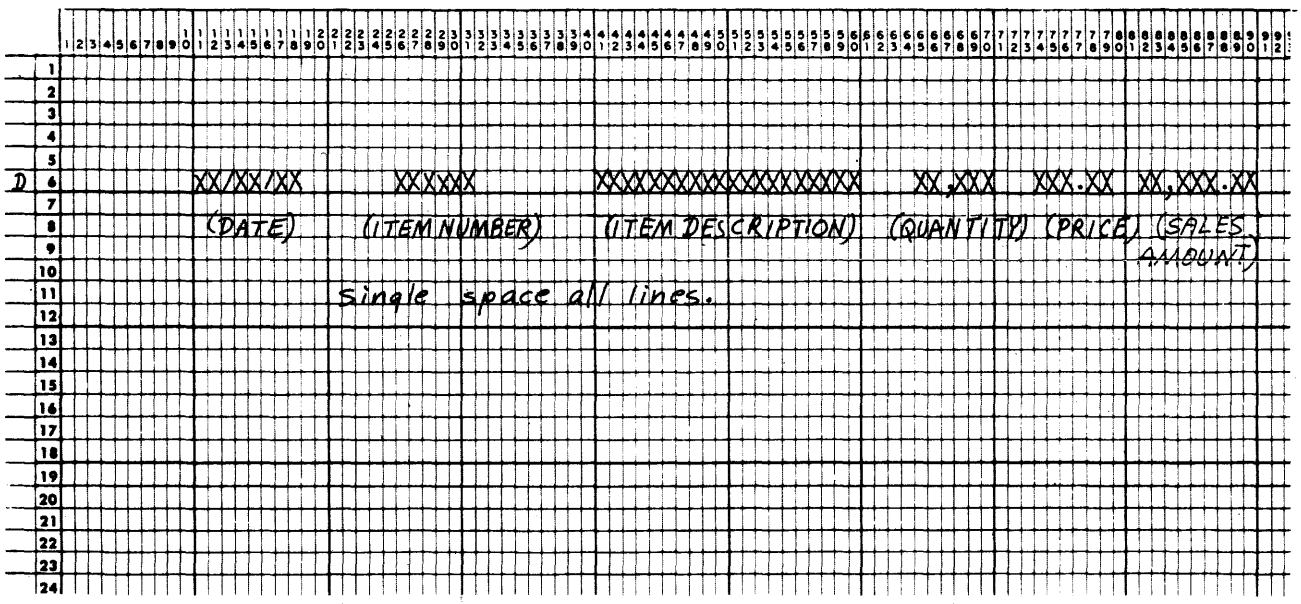
Input: Sales transaction file consisting of 96-character records. The format of the input records is shown on this record layout form:



Output: A printed transaction register:

7/23/--	413010	CH001 BOX 100A FLUSH	10	4.90	49.00
7/23/--	412146	CH14P BREAKER 15A	100	.89	89.00
7/23/--	411116	1500 TWIN SOCKET B	500	1.12	560.00
7/24/--	503029	MOTOR 1/2 HP 60 CYC	2	146.78	293.56
7/24/--	317802	TERMINAL CLIP	100	5.12	512.00
7/24/--	326917	TERMINAL BAR	100	4.12	412.00
7/24/--	411121	1506 SOCKT ADAPT BRN	400	.19	76.00
7/24/--	412997	CH173 BREAKER 30A	60	1.15	69.00
7/24/--	413098	CH176 BREAKER 60A	40	1.15	46.00
7/24/--	411174	C151 STL SWITCH BRN	200	1.16	232.00
7/24/--	413090	CH005 BR BOX 150A	10	4.98	49.80
7/24/--	718326	FL803 FUSE 15A	200	.32	64.00

This printer spacing chart shows how the report is formatted:



RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9092 UM/050*
Printed in U.S.A.

Program: Transaction Register
Programmer: John Doe
Date: 1/10/--
Keying Instruction: Graphic / Key
Card Electro Number
Page 01 of __
Program Identification: TRNREG

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

H

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	Reserved	Currency Symbol	Date Format	Date Edit	Inverted Print	Reserved	Number of Print Positions	Alternate Collating Sequence	Reserved	Inquiry	Reserved	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Reserved	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG to RPG II Conversion	Number of Formats	S/3 Conversion	Subprogram	CICS/DL/I	Transparent Literal
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	H																																	

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

F

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	Device	File Addition/Unordered	K	Option	Entry	A/U	R/U/I
			I/O/U/C/D	P/S/C/R/T/D/F	E	A/D	F/V/S/M/D/E			L/R		A/P/I/K	I/X/D/T/R/ or 2										
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	TRANS	I	P		F			96														
03	F	REPORT	O			F			96								PRINTER						
04	F																						

The same files are used for Example 1. Therefore, the File Description entries are the same.

RPG INPUT SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9094 UM/050*
Printed in U.S.A.

Program: Transaction Register
Programmer: John Doe
Date: 1/10/--
Keying Instruction: Graphic / Key
Card Electro Number
Page 02 of
Program Identification: TRNREG

I

Line	Form Type	Filename or Record Name / Data Structure Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / 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	Data Structure: Occurs n Times	Data Structure: Length	Decimal Positions	RPG 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																											
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	I	TRANS					01																									
02	I																				L		60	8		DATE						
03	I																				7		12	8		ITEMNO						
04	I																				13		32	30		DESC						
05	I																				33		37	03		QTY						
06	I																				38		42	2P		RICE						
07	I																															
08	I																															
09	I																															
10	I																															
11	I																															
12	I																															
13	I																															
14	I																															
15	I																															
16	I																															
17	I																															
18	I																															
19	I																															
20	I																															

Input specifications are the same as for Example 1 because the same input file is used.

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050* Printed in U.S.A.

Program: Transaction Register
Programmer: John Doe
Date: 1/10/--
Page: 03
Program Identification: TRNREG

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions
01	C	QTY	MULT	PRICE	EXTCST	7	2

This specification tells the system what calculation to perform. QTY must be multiplied by PRICE. QTY and PRICE are fields from the input record and are described on the Input Specifications form. Notice, however, that a new field, EXTCST, is created to hold the result. We needed a new result field because we could not use any of the fields described on the Input Specifications form. If we had used one of them, we would lose information needed for printing the detail line.

You must define any new field by describing field name, field length, and decimal positions. We chose the name EXTCST, an abbreviation for extended cost. Any valid name could be used. According to the printer spacing chart, the EXTCST field is 7 positions long with two decimal positions. We, therefore, used these figures when defining field length and decimal positions.

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050* Printed in U.S.A.

Program: Transaction Register
Programmer: John Doe
Date: 1/10/--
Page: 04
Program Identification: TRNREG

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space After	Output Indicators	Field Name or EXCPT Name	Edit Codes	End Position in Output Record
01	O	REPORT	D	1	01			
02	O					DATE	Y	18
03	O					ITEMNO		31
04	O					DESC		60
05	O					QTY	Z	70
06	O					PRICE	3	79
07	O					EXTCST	1	90

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
5-9 = User Defined

This Output Specifications form differs from the one in Example 1 by only one entry. The field EXTCST was added because it is to be included in the output line. EXTCST is to be edited with commas, so we use edit code 1.

Writing Specifications for Indicators

So far you have learned how to use the program cycle for producing simple reports. However, actual business reports would be more complex. They would include more information, have page and column headings, and probably include subtotals and final totals.

A report like that shown in Figure 11 would require printing four different types of lines: report heading, column headings, detail lines, and total lines. Some of these lines must be printed only at certain times: headings would be printed only at the top of the page and totals only after all detail lines are printed. To produce the report correctly, you must use *indicators* to specify when you want certain things done.

To you, indicators are two numbers or alphabetic characters you specify on the specification forms. To the object program, indicators are like switches, located in the system. They mean one thing when on, another when off. You can use several types of indicators; each type signals something different.

To use indicators correctly, you must know how program cycle operations affect indicators. In this section, indicators are discussed one at a time. You will learn when to use indicators, how to specify them, and which program cycle operations are associated with each type of indicator.

TRANSACTION REGISTER

TRANSACTION DATE	ITEM NO	DESCRIPTION	QUANTITY	UNIT PRICE	SALES AMOUNT
7/23/--	413010	CH001 BOX 100A FLUSH	10	4.90	49.00
	412146	CH148 BREAKER 15A	100	.89	89.00
	411116	1500 TWIN SOCKET B	500	1.12	560.00
					698.00
7/24/--	503029	MOTOR 1/2 HP 60 CYC	2	146.78	293.56
	327802	TERMINAL CLIP	100	5.12	512.00
	326013	TERMINAL BAR	100	4.12	412.00
	411121	1506 SOCKT ADAPT BRN	400	.19	76.00
	412997	CH173 BREAKER 30A	60	1.15	69.00
	413088	CH176 BREAKER 60A	40	1.15	46.00
	411174	C151 SIL SWITCH BRN	200	1.16	232.00
	413090	CH005 BR BOX 150A	10	4.98	49.80
	718326	FC803 FUSE 15A	200	.32	64.00
					1,754.36
7/27/--	321071	2-SPEED SAW	1	28.44	28.44
	325781	SATIN-CUT DADO SET	1	39.50	39.50
	412146	CH143 BREAKER 15A	50	.89	44.50
	573022	6-VOLT POWER BATTERY	2	14.45	28.90
					141.34
			WEEKLY TOTAL		2,593.70

Note: This report is similar to those shown before, but note the addition of headings and totals.

Figure 11. Printed Report

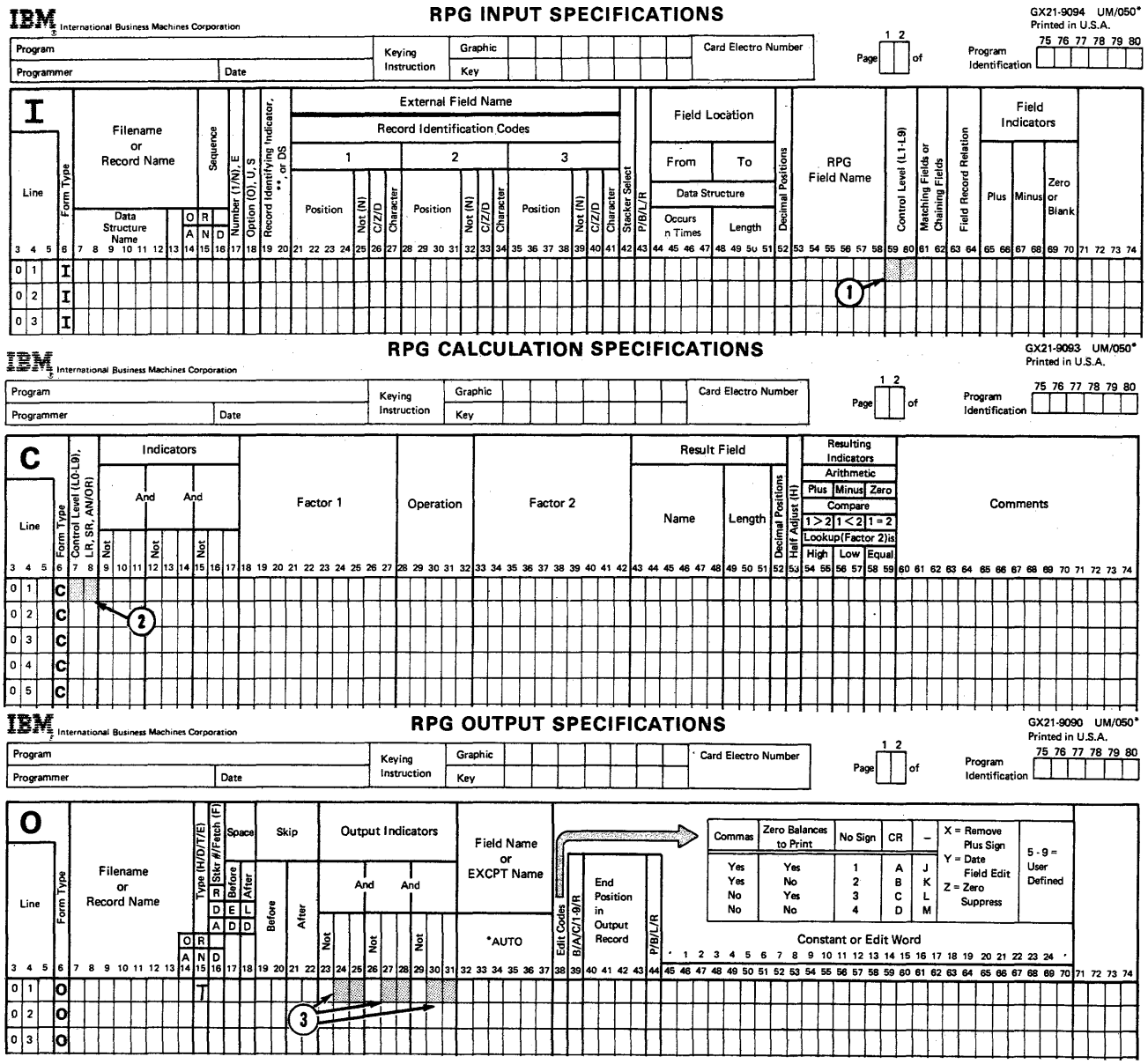
CONTROL LEVEL INDICATORS

Control level indicators are used when you want to calculate and print totals. Nine different indicators can be used (L1 through L9), allowing as many as nine different levels of totals in the same program. The control level indicators tell the program two things:

- When totals should be calculated
- Which calculations and output operations are total operations

A control level indicator in positions 59 and 60 next to an input field specified on the Input Specifications form determines when totals should be calculated and printed. This input field is called a *control field*. Whenever the contents of the control field changes, a *control break* occurs. A control break turns on the control level indicator assigned to the control field and all lower control level indicators. The system performs all calculations and output operations (total operations) that are conditioned by these control level indicators. If a control break causes L3 to be turned on, L1 and L2 are turned on. This allows control over several levels of totals and subtotals. Examples are daily, weekly, and monthly totals.

In Figure 11, the records are grouped by date, and a total of the sales amount field should be printed when the contents of the input record date field changes.



Control level indicators are specified at (1) to tell the compiler when total operations are to be done. They are used at (2) to tell which operations are total calculations. With a T in position 15, they are used at (3) to tell when total output lines are to be written.

Program Cycle Operations

Figure 12 shows the program cycle operations associated with control level indicators. The system can do calculations and output operations at two different times in one cycle: at detail time and at total time. Total operations associated with control level indicators are not done in every cycle; they are done during the cycle in which the control field changes.

After a record is read, the program determines whether the control field in the record just read is different from the control field in the previous record. If it is, a control break occurs and the control level indicator you specified is set on. When the indicator is on, it means that all records in the control group have been read and that total operations can be performed. Control level indicators remain on through the detail calculations and output processing of the record that caused the control break. They are then set off before the next record is read.

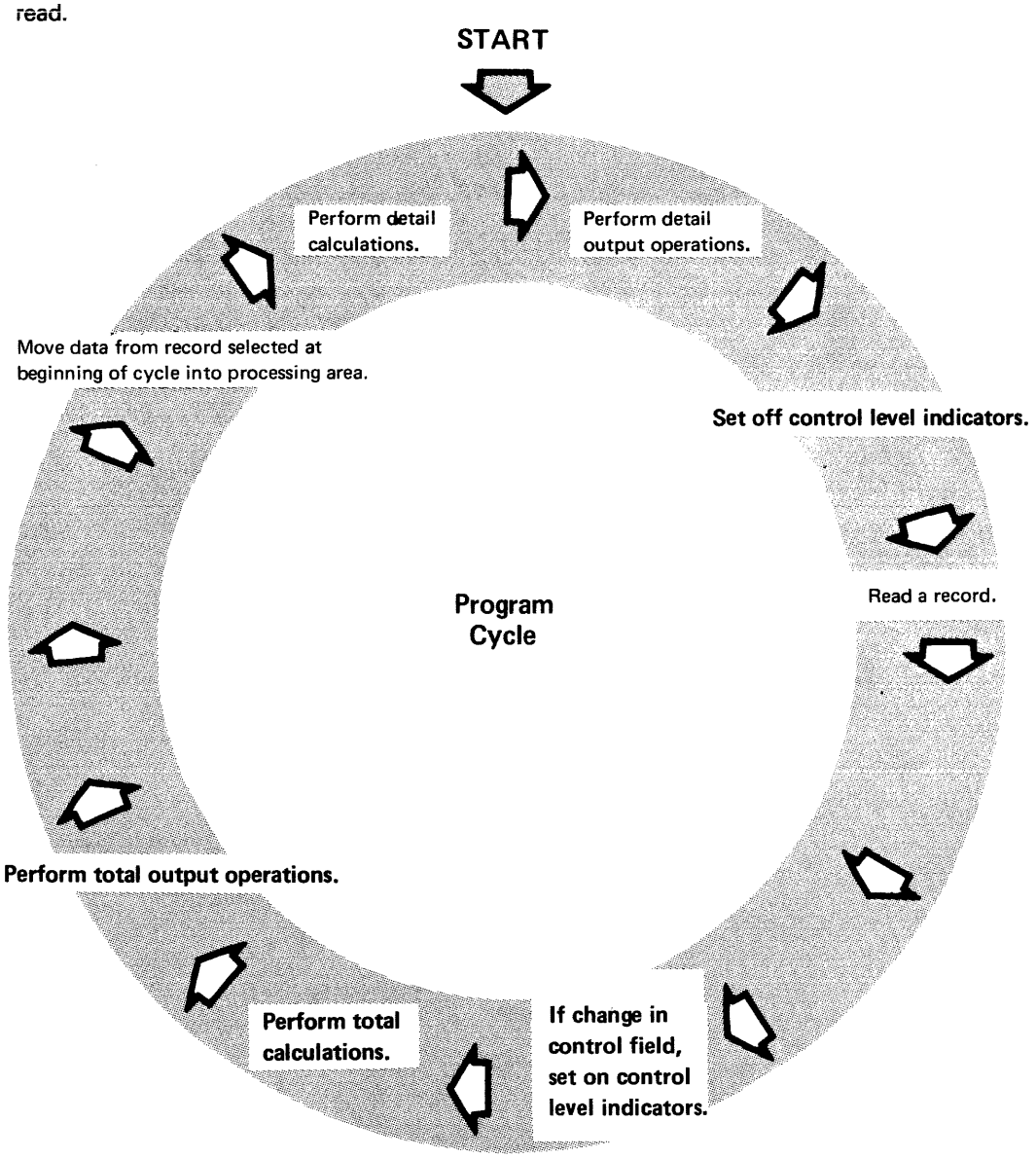
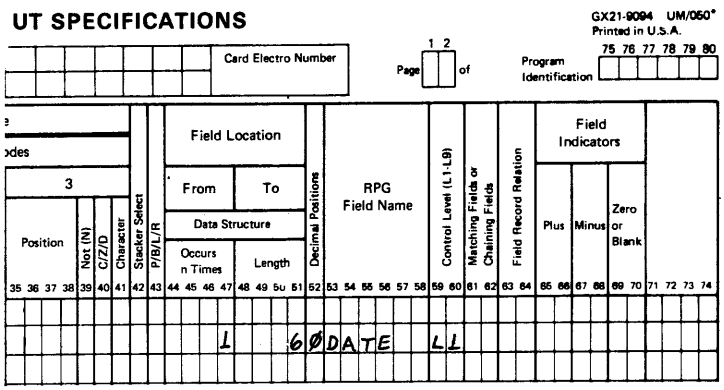


Figure 12. Program Cycle Operations for the Control Level Indicators

Detail operations for the record that caused the control break are done only after total operations for previous records. The control level indicator assigned to the field that caused the control break remains on so that the first record of the group can be identified with that indicator. The record that caused the control break is not processed before the total operations are done, because information from that record would be included with information from records in the previous group. The totals from the previous group would then be wrong.

RPG Specifications

To specify a field as a control field, you assign a control level indicator (L1 through L9) to an input field in positions 59 and 60 on the Input Specifications form:



L1, assigned on the same specification line as the date field, tells the system to use DATE as the control field.

To specify which operations are total operations, you assign the same control level indicator in positions 7 and 8 on the Calculation Specifications form and in positions 24 and 25, positions 27 and 28, or positions 30 and 31 on the Output Specifications form. On the Output Specifications form, you must also enter a T in position 15.

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	P/B/L/R	Field Location From	To	Data Structure: Occurs n Times	Length	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	INPUT				01																										
02	I																								L	60	DATE	L1				
03	I																								33	37	QTY					
04	I																								38	42	PRICE					
05	I																															

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9), LR, SR, AN/OR	Indicators: And 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					QTY	MULT	PRICE	EXTCST		72									
02	C					EXTCST	ADD	DAYTOT	DAYTOT		92									
03	C	L1				DAYTOT	ADD	FINCST	FINCST		102									
04	C																			
05	C																			
06	C																			
07	C																			
08	C																			
09	C																			
10	C																			
11	C																			

Use a control level indicator in positions 7 and 8 to show that a total operation is to be done only when a control break occurs. The operations on line 01 and line 02 are done during the detail processing of each input record. The ADD operation in line 03 is a total operation that will be done when L1 is on; that is, when the DATE field changes.

RPG OUTPUT SPECIFICATIONS

GX21-9090-4 UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

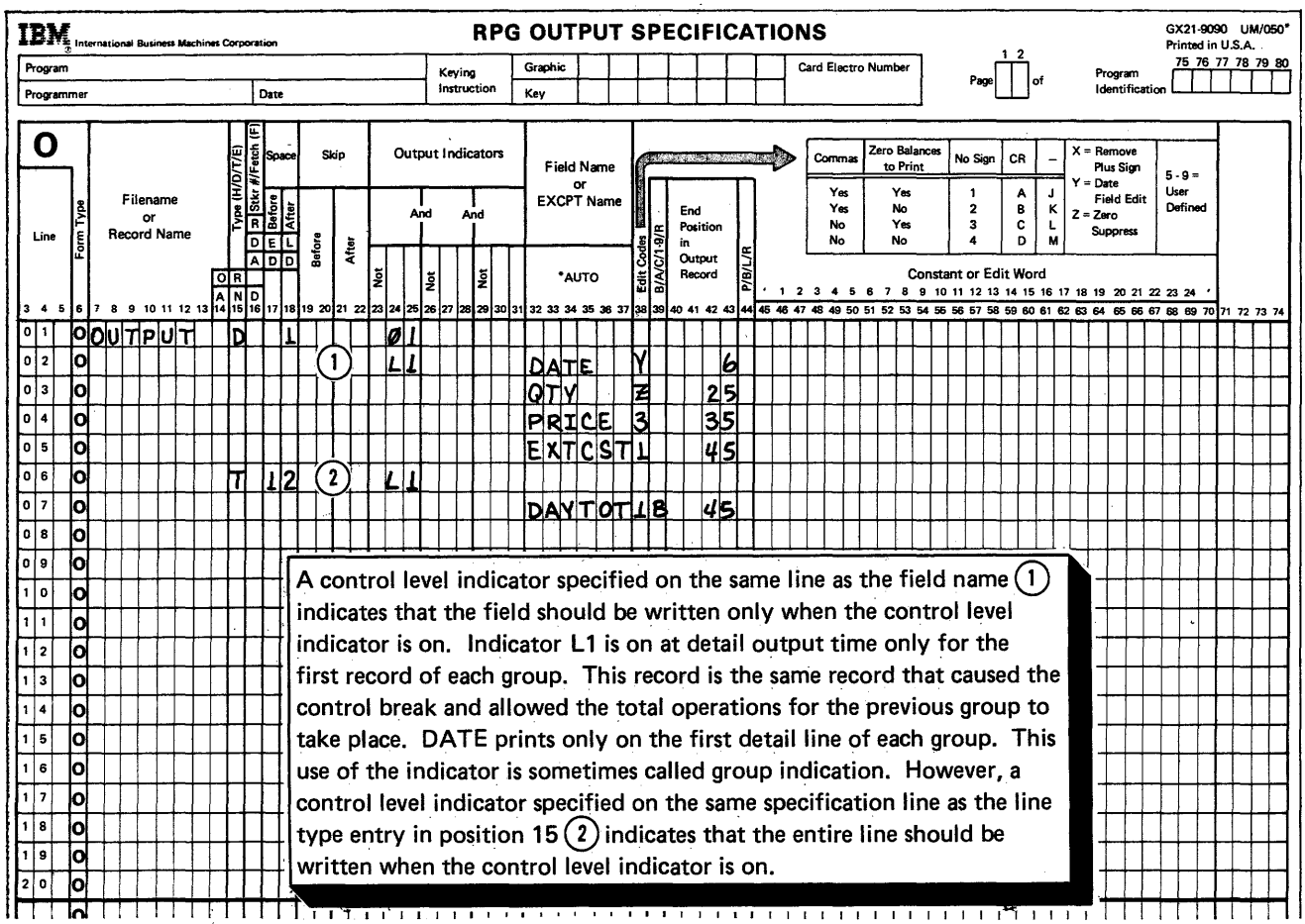
Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Fetch (F)	Space: Skr #/Before	After	Skip: Before	After	Output Indicators: And Not	And Not	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	OUTPUT	D		1				01							
02	O										DATE			6		
03	O										QTY		Z	25		
04	O										PRICE	3		35		
05	O										EXTCST	1		45		
06	O		T		12				L1							
07	O										DAYTOT	1B		45		
08	O															
09	O															
10	O															
11	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; 5 - 9 = User Defined

The T in position 15 indicates which output records are total records. Every total record should also have an indicator specified to tell the system when to do the operation. The output operation described on lines 06 and 07 is done only when L1 is on.

You can specify up to three different indicators on a line on the Output Specifications form. If you are using only one indicator, you can enter it in any one of the three positions. The control level indicators specified on this form can be used to condition an entire output record or only certain fields in the output record.



Using the Blank-After Specification

In RPG, you can set fields in storage to blanks (in the case of alphameric fields) or zeros (in the case of numeric fields) after they have been written out. You do this by entering a B in position 39 of the Output Specifications form.

This is a particularly useful feature when you are doing total operations. It allows you to use the same field over and over for accumulating and printing totals. For example, you could use a numeric field to accumulate totals for a particular group of records. After the totals are accumulated and printed for that group, you can use the same numeric field to accumulate the totals for the next group of records. To do this, place a B in position 39 for the total field. If you do not place a B in position 39, the totals for the second group of records would be added to the totals for the first group of records.

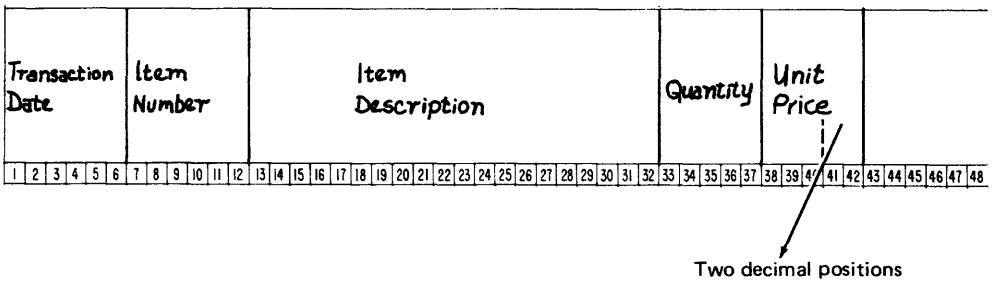
EXAMPLE 3 (TRNREG): USING CONTROL LEVEL INDICATORS TO CALCULATE AND PRINT TOTALS

Program Definition

Print a weekly sales transaction report that lists all daily transactions and gives the total sales for each day. This report is similar to the reports produced in Examples 1 and 2. All items sold each day are listed. Item number, item description, quantity sold, unit price, and sales amount (quantity times unit price) are included for each item. The date is printed only for the first transaction encountered that has a new date (the date is used as a group indication). The total sales amount for a day is printed after all transactions for that day have been recorded.

Program Requirements

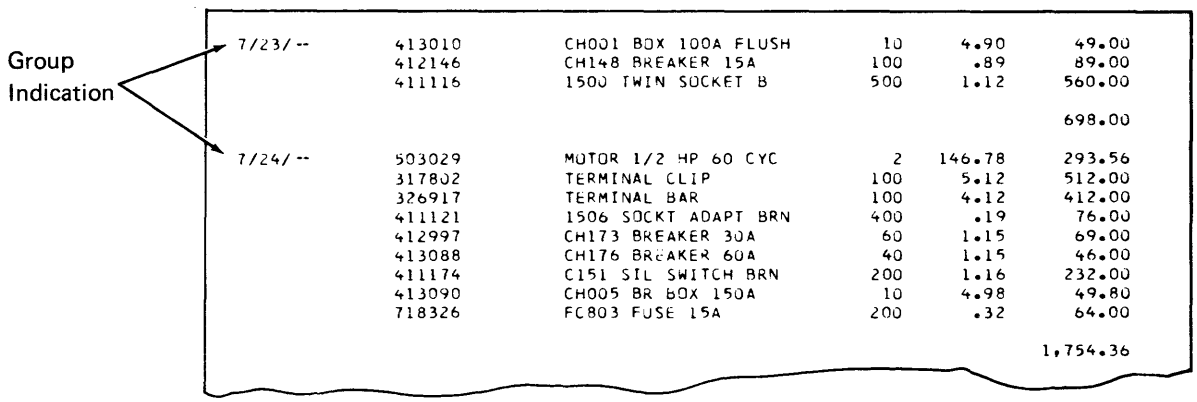
Input: Sales transaction file consisting of 96-character records. The records are arranged in ascending order by date. The format of the input records is shown on the record layout form:



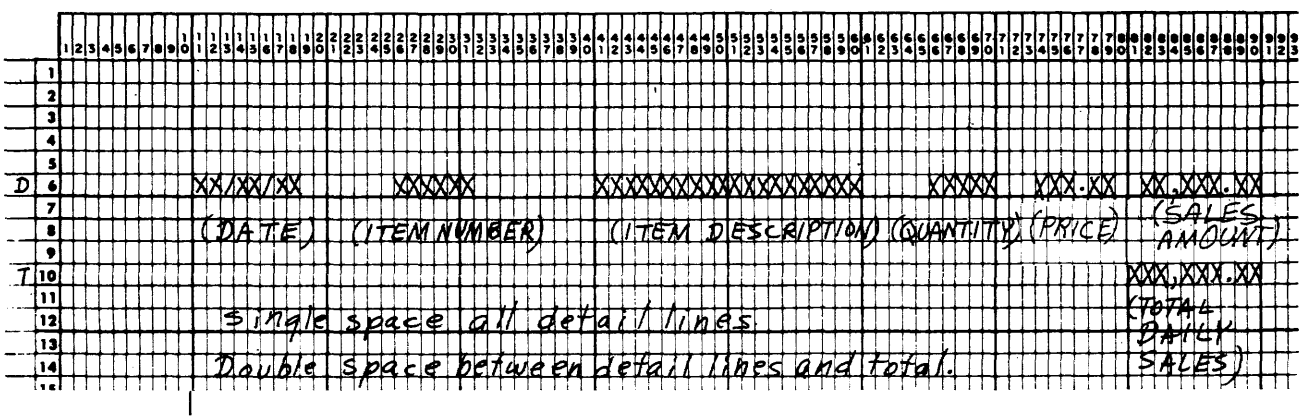
Processing:

- Multiply quantity times unit price to find sales amount.
- Find total of all item sales per day.

Output: A printed transaction register:



This printer spacing chart shows how the report is formatted:



RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9092 UM/050*
Printed in U.S.A.

Program: Transaction Register | Keying Instruction: Graphic / Key | Card Electro Number
Programmer: John Doe | Date: 1/10/--

Page 01 of __ | Program Identification: TRNREG

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	Reserved	Currency Symbol	Date Format	Date Edit	Inverted Print	Reserved	Number of Print Positions	Alternate Collating Sequence	Reserved	Inquiry	Reserved	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Reserved	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG to RPG II Conversion	Number of Formats	S/3 Conversion	Subprogram	CICS/DL/I	Transparent Literal
01	H																																

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

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 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, UC
			I/O/U/C/D	P/S/C/R/T/D/F	E	A/D	F/V/S/M/D/E	External Record Name		L/R	A/P/I/K	1/X/D/T/R/ or 2																		
02	F	TRANS	I	P			F		96																					
03	F	REPORT	O				F		96								PRINTER													
04	F																													
05	F																													

No new entries are made on this form.

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM/050*
Printed in U.S.A.

Program: Transaction Register | Keying Instruction: Graphic / Key | Card Electro Number
Programmer: John Doe | Date: 1/10/--

Page 02 of | Program Identification: TRNREG

Line	Form Type	Filename or Record Name / Data Structure Name	Sequence A	Sequence N	Sequence D	Number (1/N), E	Option (O), U, S	Record Identifying Indicator: *, or DS	External Field Name / 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	Data Structure: Occurs n Times	Data Structure: Length	Decimal Positions	RPG 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	TRANS						01																										
02	I																						1	6			0	DATE	L1					
03	I																						7	12				ITEMNO						
04	I																						13	32				DESC						
05	I																						33	37			0	QTY						
06	I																						38	42			2	PRICE						
07	I																																	
08	I																																	
09	I																																	
10	I																																	
11	I																																	
12	I																																	
13	I																																	
14	I																																	
15	I																																	
16	I																																	

Input fields are described as before. In this program, totals must be accumulated and printed. You know that totals must be printed whenever a record with a different date field is read. The date field determines when total operations should be done. The date field is the control field and must be specified as such. This is done by the L1 entry in positions 59 and 60.

RPG CALCULATION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9093 UM/050* Printed in U.S.A.

Program: Transaction Register | Programmer: John Doe | Date: 1/10/-- | Keying Instruction | Graphic | Key | Card Electro Number

Page 03 of | Program Identification: TRNREG

C

Line	Form Type	Control Level (L0-L9), LR, SR, AN/OR	Indicators: And / 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					QTY	MULT	PRICE	EXTCST	7	2			
02	C					EXTCST	ADD	DAYTOT	DAYTOT	8	2			
03	C													
04	C													
05	C													
06	C													
07	C													
08	C													
09	C													

For each item, QTY must be multiplied by PRICE to get EXTCST (sales amount). To get the total of all sales made during the day, EXTCST is added to DAYTOT (the field used to accumulate daily sales total).

RPG OUTPUT SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9090 UM/050* Printed in U.S.A.

Program: Transaction Register | Programmer: John Doe | Date: 1/10/-- | Keying Instruction | Graphic | Key | Card Electro Number

Page 04 of | Program Identification: TRNREG

O

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Stkr #/Fetch (F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And / Not	And / Not	And / Not	Field Name or EXCPT 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	–	Constant or Edit Word
01	O	REPORT	D		1				01													
02	O								L1			DATE	Y		18							
03	O											ITEMNO			31							
04	O											DESC			60							
05	O											QTY	Z		70							
06	O											PRICE	3		79							
07	O											EXTCST	1		90							
08	O																					
09	O		T	12					L1													
10	O											DAYTOT	1B		90							
11	O																					
12	O																					
13	O																					
14	O																					
15	O																					
16	O																					
17	O																					
18	O																					
19	O																					
20	O																					

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

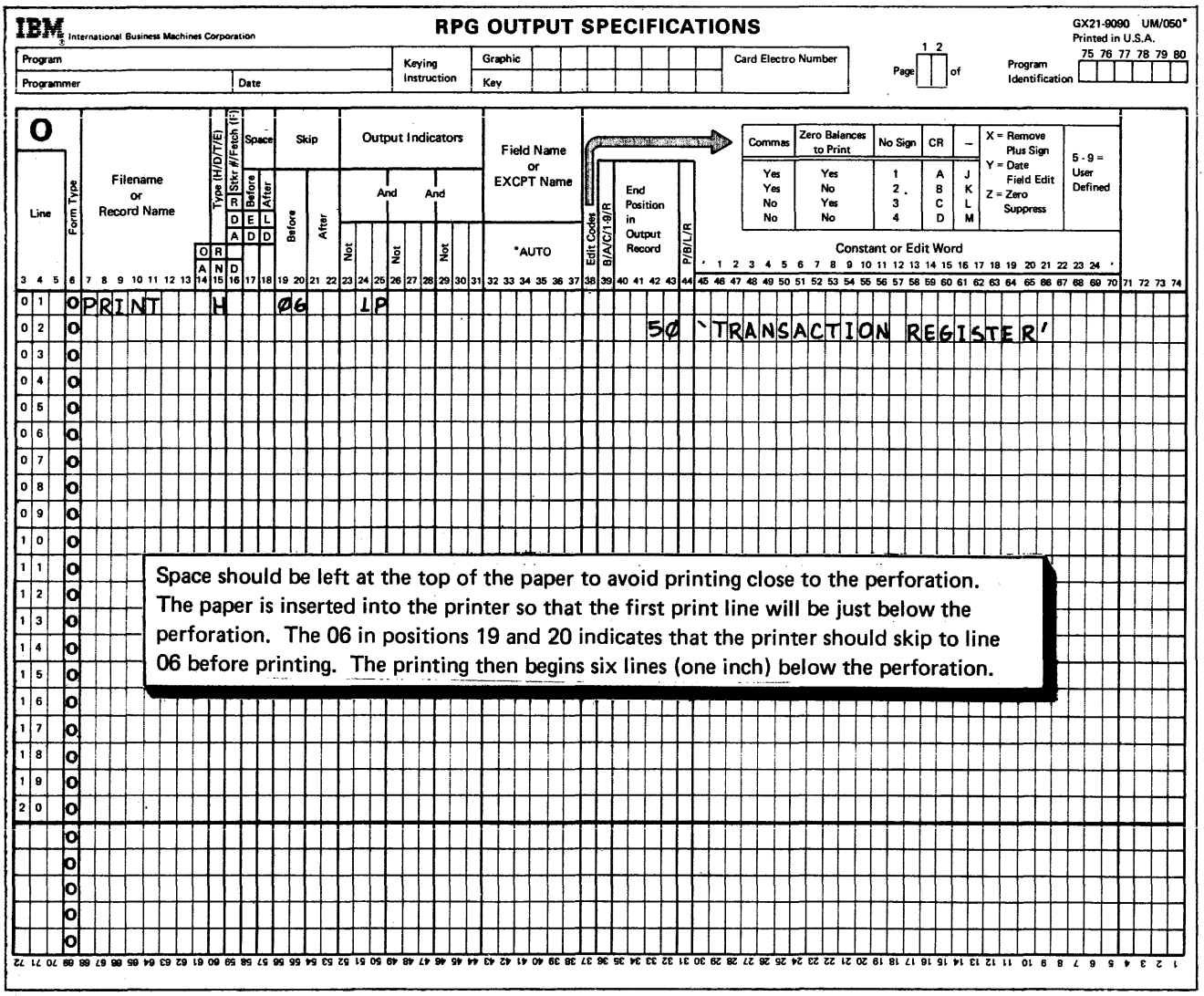
X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress; 5-9 = User Defined

Two different lines—detail and total—are needed for this report (note D and T in position 15). The detail line is described first. According to the report, the date field is to print only for the first record in a new control group. We do this by conditioning the date field with the L1 indicator. The date will now print only when L1 is on; that is, for the first record in each control group.

The total line, which contains only one field, is described next. The entire line is conditioned by L1 because it is a total line. The B in position 39 causes the DAYTOT field to be reset to zero after it is output, but before sales from the next group are added to it. If DAYTOT were never blanked out, the totals of all days would be accumulated.

FIRST PAGE INDICATOR

The first page (1P) indicator is used on the Output Specifications form to specify the headings you want printed only on the first page of a report. Headings, usually printed at the top of the page, include such things as report titles or column names:

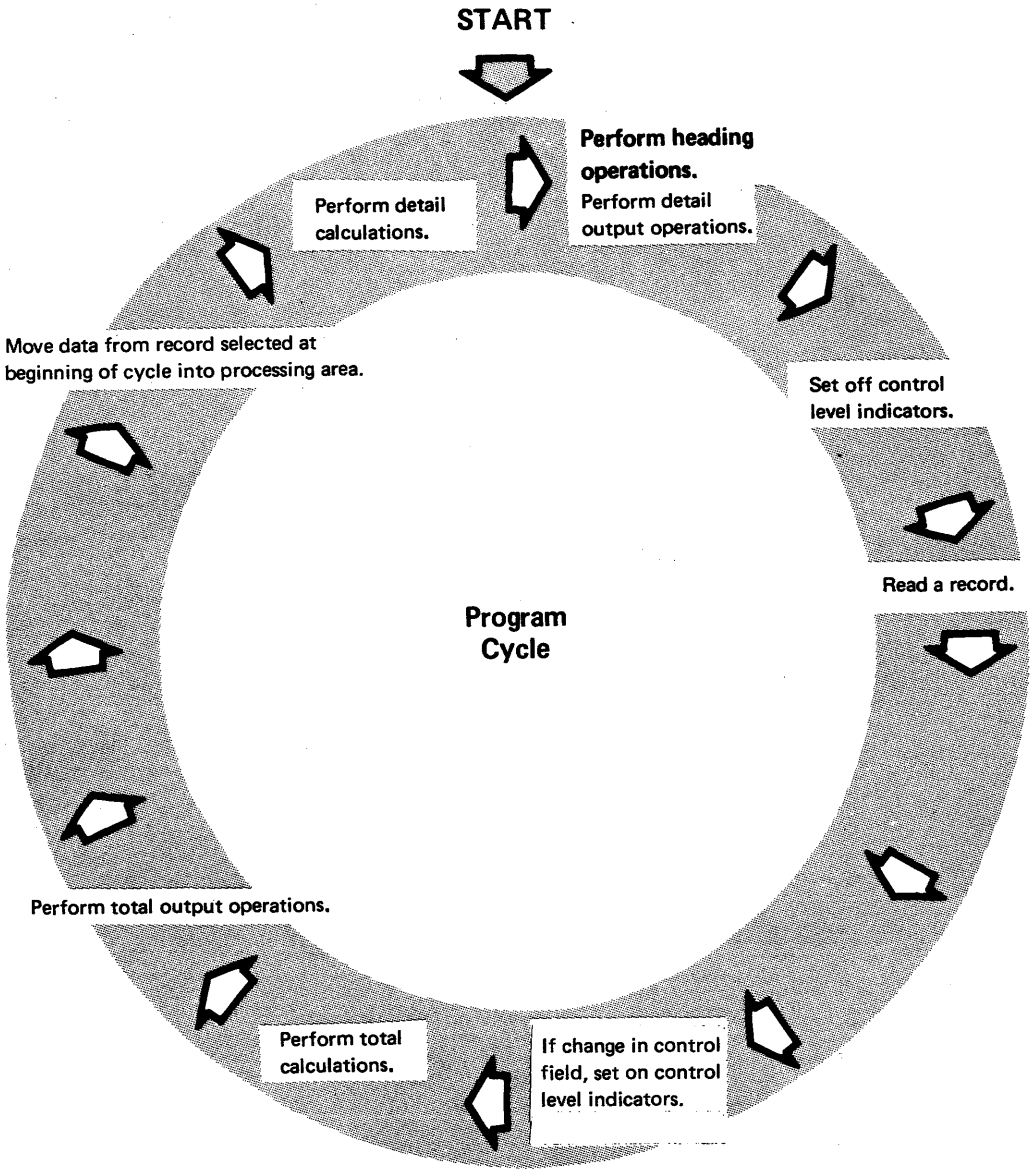


Space should be left at the top of the paper to avoid printing close to the perforation. The paper is inserted into the printer so that the first print line will be just below the perforation. The 06 in positions 19 and 20 indicates that the printer should skip to line 06 before printing. The printing then begins six lines (one inch) below the perforation.

Program Cycle Operations

One operation in the program cycle concerns the 1P indicator (see Figure 13). The 1P indicator is automatically set on at the beginning of every program, so the first operation in the program is to print any output record conditioned by 1P. After this is done, the first record is read and the program cycle operations are executed in order.

Headings conditioned by 1P are printed only once–at the beginning of the program on the first page of the report. Any heading records that are not conditioned by 1P are handled in the same way as detail records. This means that they are printed along with detail records in every cycle.

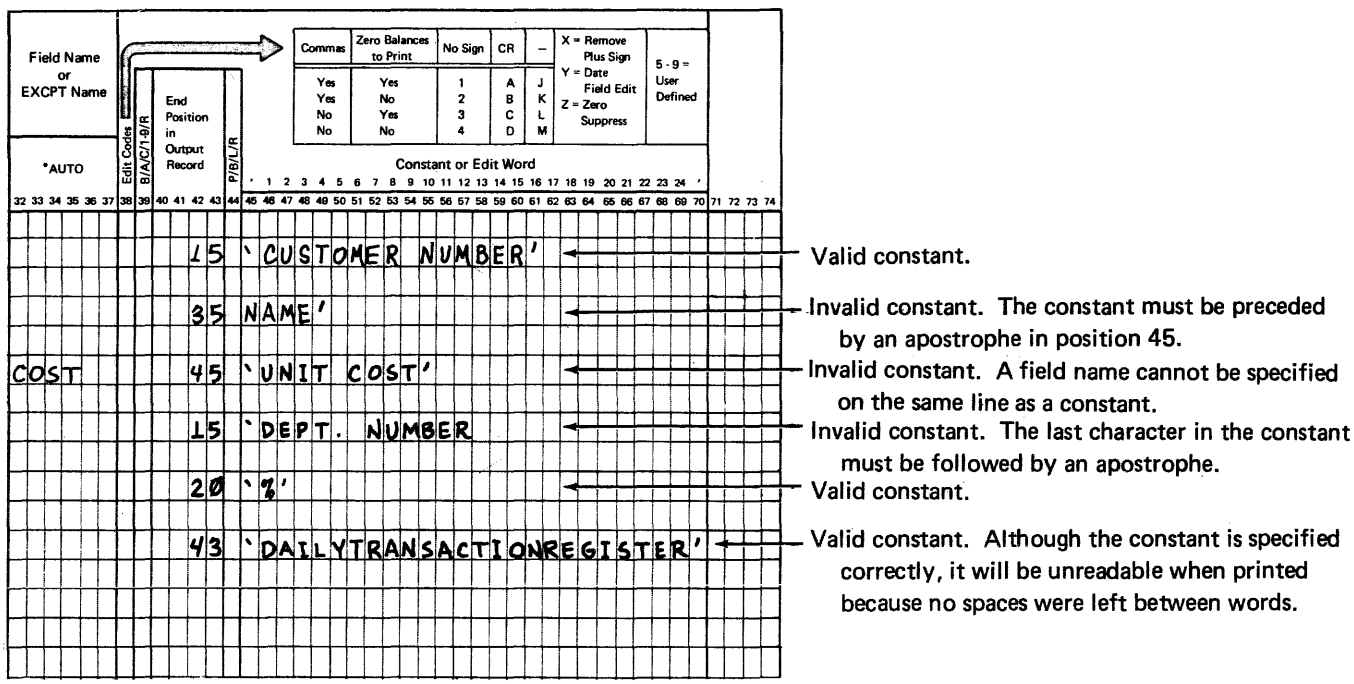


Note: The first operation in the first program cycle concerns output operations conditioned by the first page (1P) indicator.

Figure 13. Program Cycle Operations for 1P Indicators

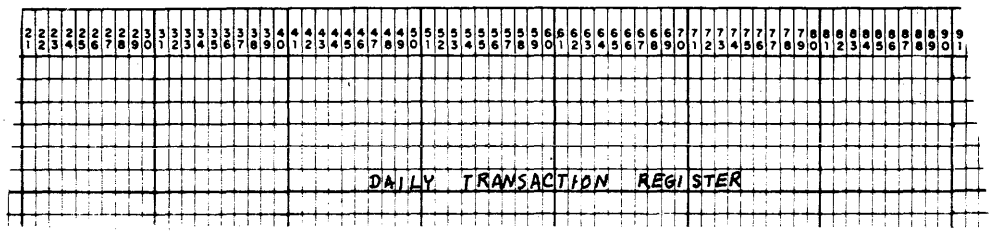
Heading information to be printed on the first page of a report is specified by using constants (actual information instead of field names). Constants for headings must be specified according to these rules:

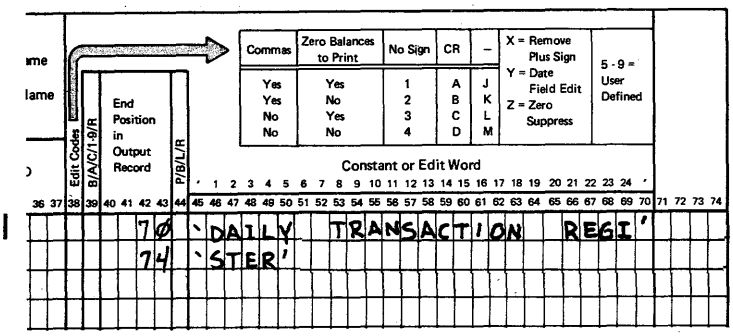
- Constants must be entered in positions 45 through 70 of the Output Specifications form.
- Constants can contain any of the characters in the data character set (the 256 EBCDIC characters).
- Constants must be enclosed in apostrophes. (The beginning apostrophe is always entered in position 45.)
- No field name can be used on the same line as a constant.
- An end-position entry must be entered for every constant.

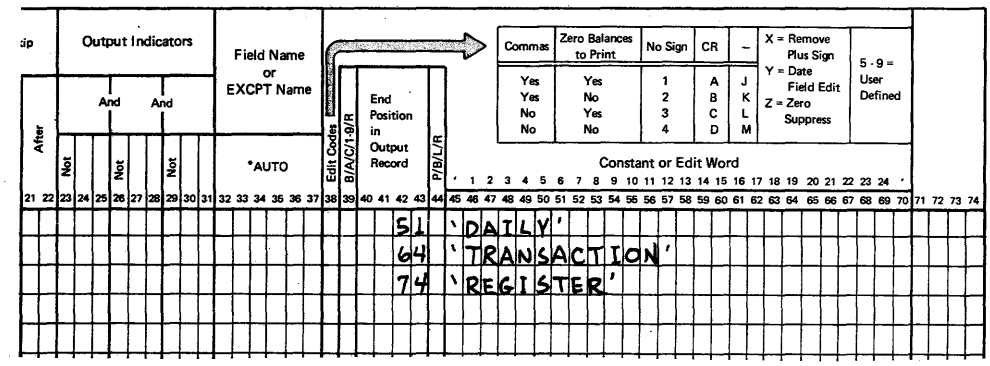


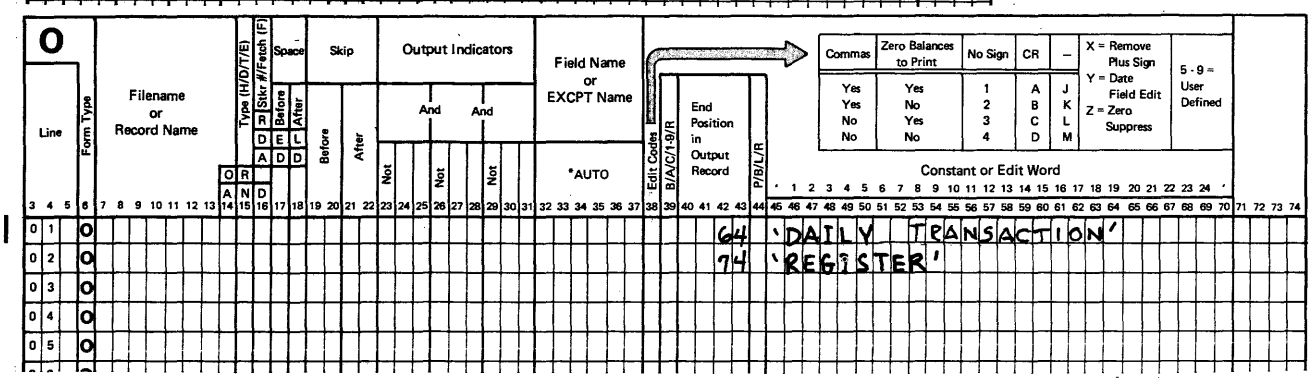
Headings too long to specify on one line of the Output Specifications form can be split and placed on separate lines. You must, however, give an end position for each part.

The heading shown in the printer spacing chart takes 28 positions. A constant containing a maximum of 24 characters can be specified on one specification line. Because the entire heading cannot be specified on one line, it must be broken into parts. The examples given show three different ways to specify this heading:









Heading lines should be specified first on the Output Specifications form. The best method is to specify your record types in this order: heading, detail, and total.

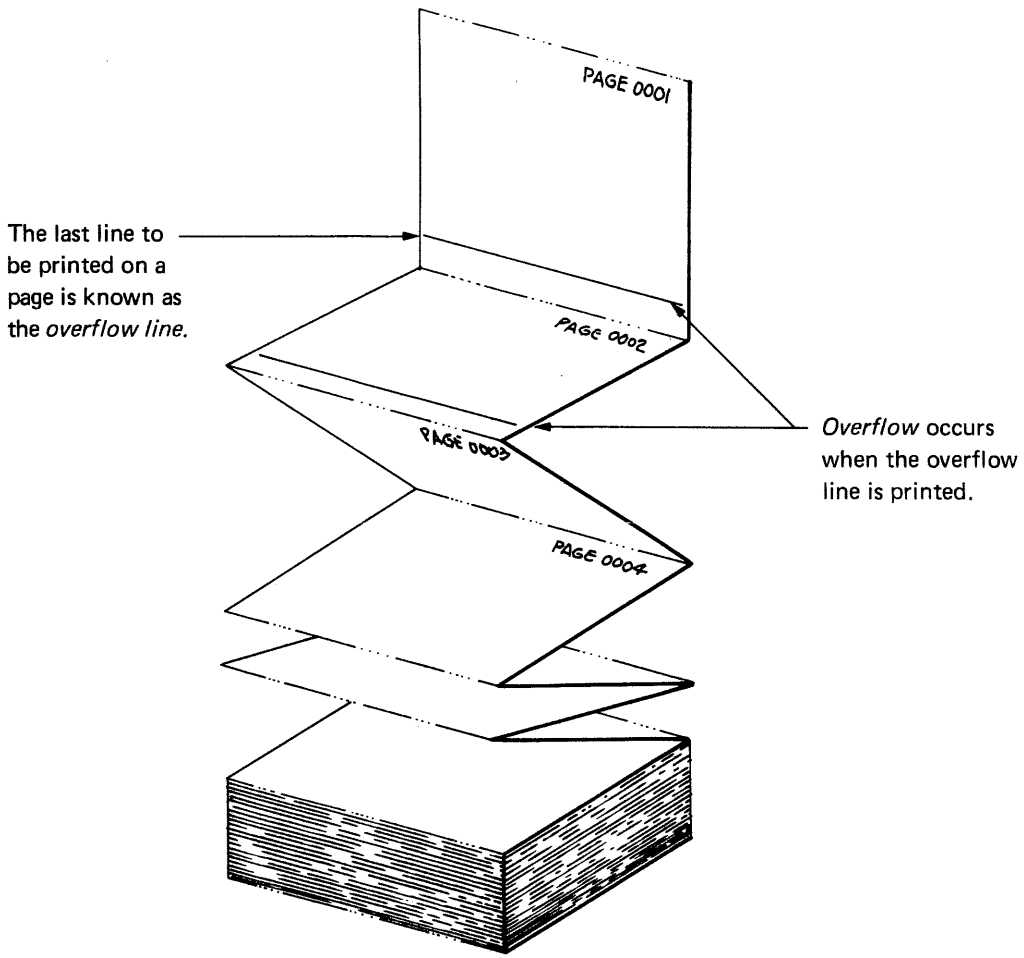
OVERFLOW INDICATORS

You use overflow indicators to:

- Print headings on every page but the first page of a report (the 1P indicator allows headings to be printed on the first page).
- Control where printing begins and ends on a page.
- Advance forms from one page to the next (provided a skip specification is also used).

To understand how overflow indicators work, you must know how the concept of overflow is defined in RPG:

- *Overflow* means the lines that remain to be printed after a page is full.
- *Overflow handling* means advancing forms to a new page after the last line has been printed on the current page.
- *Overflow line* means the last line to be printed on a page.
- *Overflow page* means the new page to be printed when overflow occurs.



Printers use continuous forms (a series of pages divided by perforations). Overflow handling refers to the means of advancing forms from one page to the next.

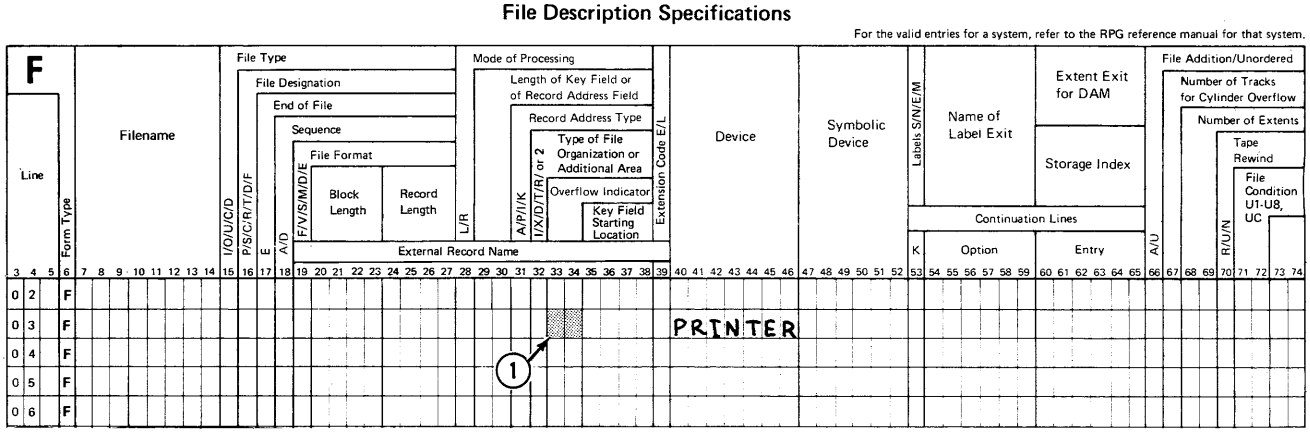
Overflow can be handled automatically by the system or through specifications you write.

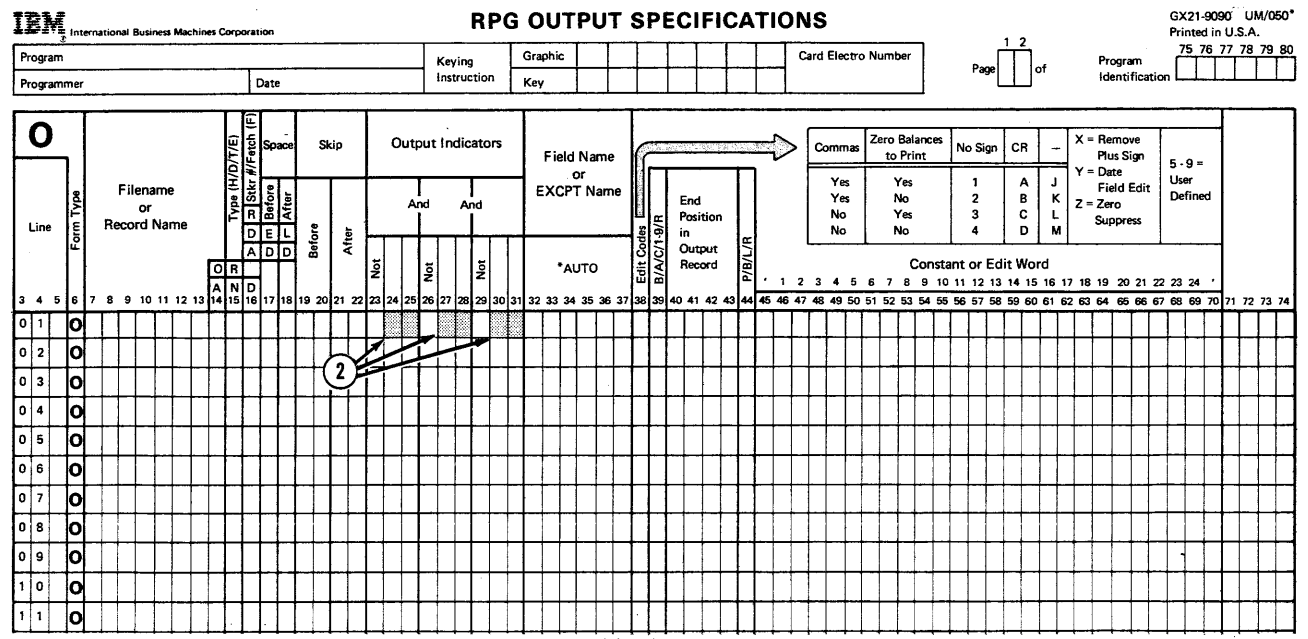
Note: This discussion assumes that you are using standard 66-line forms in the printer.

If the system handles overflow, the first heading line prints on the line specified by the skip-before entry (positions 19 and 20 of the Output Specifications form). If no skip-before entry is made, printing begins on line 06, which is the default. Printing ends on line 60 if the page length is set at 66 and if no overflow line is specified. Overflow occurs after line 60 prints; that is, the forms advance to a new page. If the system handles overflow, you cannot print special lines at the bottom of the page or any headings on overflow lines. Heading lines can be printed on the first page if 1P is used. No instructions are provided, however, to print headings on overflow pages. Heading lines are considered to be detail lines and print on each detail cycle if the conditions placed on printing are met.

When you do not want overflow handled automatically, you can specify on the coding forms how you want it handled, using overflow indicators:

File Description Specifications





Assign an overflow indicator to the printer at (1). Then use it on the Output Specifications form at (2) to show which operations must be done when overflow occurs.

Program Cycle Operations

Figure 14 shows operations in a program cycle in which overflow indicators are used.

The program sets on the overflow indicator you assigned whenever the overflow line is passed. By setting the overflow indicator on, the program remembers that overflow has occurred. As you can see in Figure 14, overflow indicators can be set on at one of two times: at detail time when a detail record prints on the overflow line, or at total time when a total record prints on the overflow line. Notice that the only time a check is made to see if the overflow indicator is on is right after total output. If the overflow indicator is on, overflow operations are done in this order:

1. Print any total lines conditioned by the overflow indicator.

2. Skip to new page, provided a skip specification was made on a line conditioned by the overflow indicator.

3. Print all heading and detail lines conditioned by the overflow indicator.

If multiple detail or total lines are to be printed in a single cycle, printing may occur past the designated overflow line. This is because all detail and total printing for a single cycle is completed before overflow operations occur.

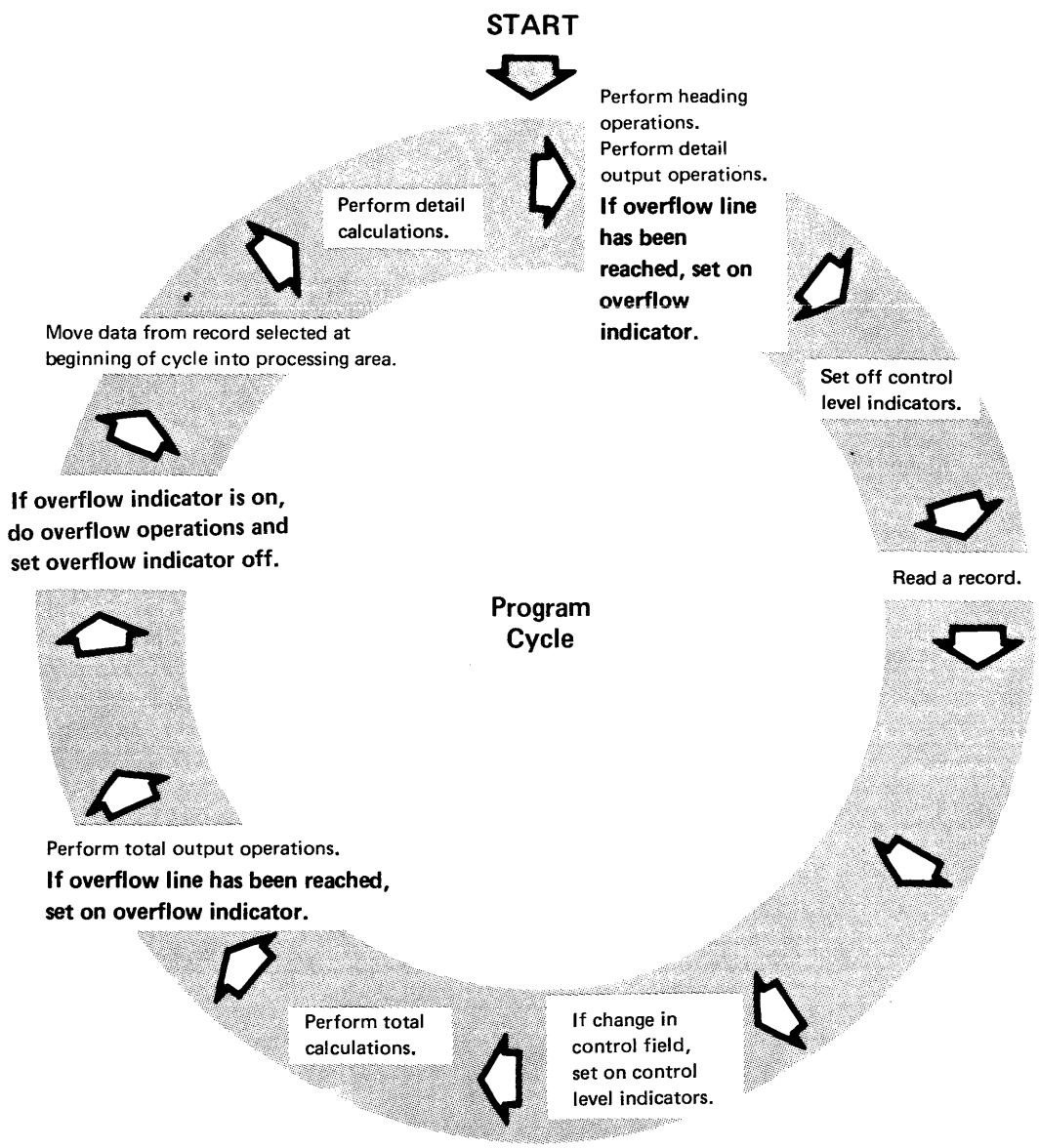
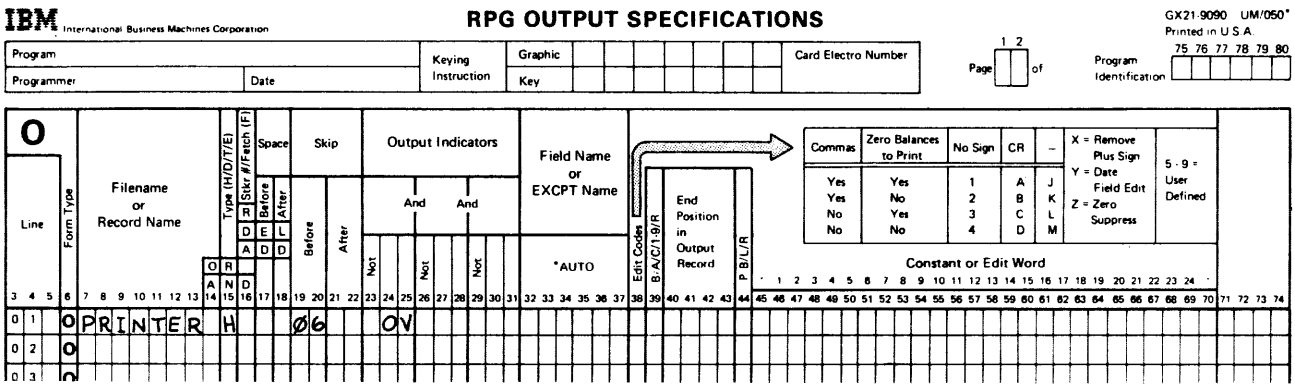


Figure 14. Program Cycle Operations for Overflow Indicators

There are eight overflow indicators: OA through OG and OV. You can enter any one of these indicators in positions 33 and 34 on the File Description Specifications form. If you have more than one printer file, however, you must specify a different overflow indicator for each file.

After you have specified an overflow indicator on the File Description Specifications form, you must specify the *same* indicator on the Output Specifications form. This specifies what you want done when overflow occurs.

Besides specifying the overflow indicator, you must also specify that forms should advance. You do this by placing a skip specification in positions 19 and 20 on the Output Specifications form.



The skip specification should be made on the first line you want printed on the page (usually a heading line). The skip entry is the line number of the beginning line (usually 6).

Always remember to enter a skip specification for advancing forms in a heading line conditioned by the overflow indicator. If you forget, forms will not advance when overflow occurs.

Using Spacing with Overflow

You already know that the overflow indicator is turned on when a record is printed on the overflow line. However, this indicator also turns on whenever the overflow line passes under the printing mechanism. This means that spacing to a line past the overflow line to a line on the same page causes the overflow indicator to turn on. Figure 15 shows an example of coding for spacing after printing.

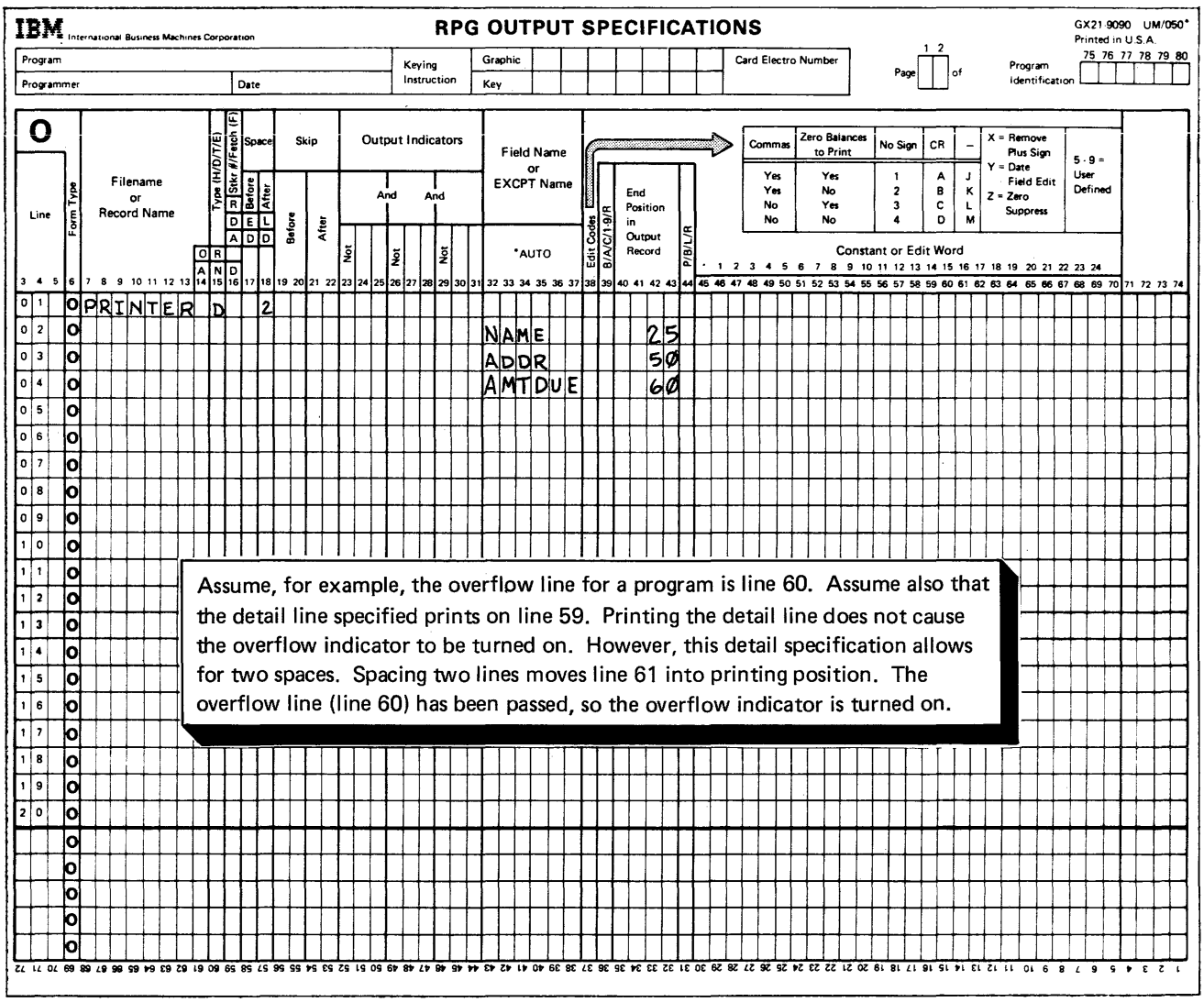
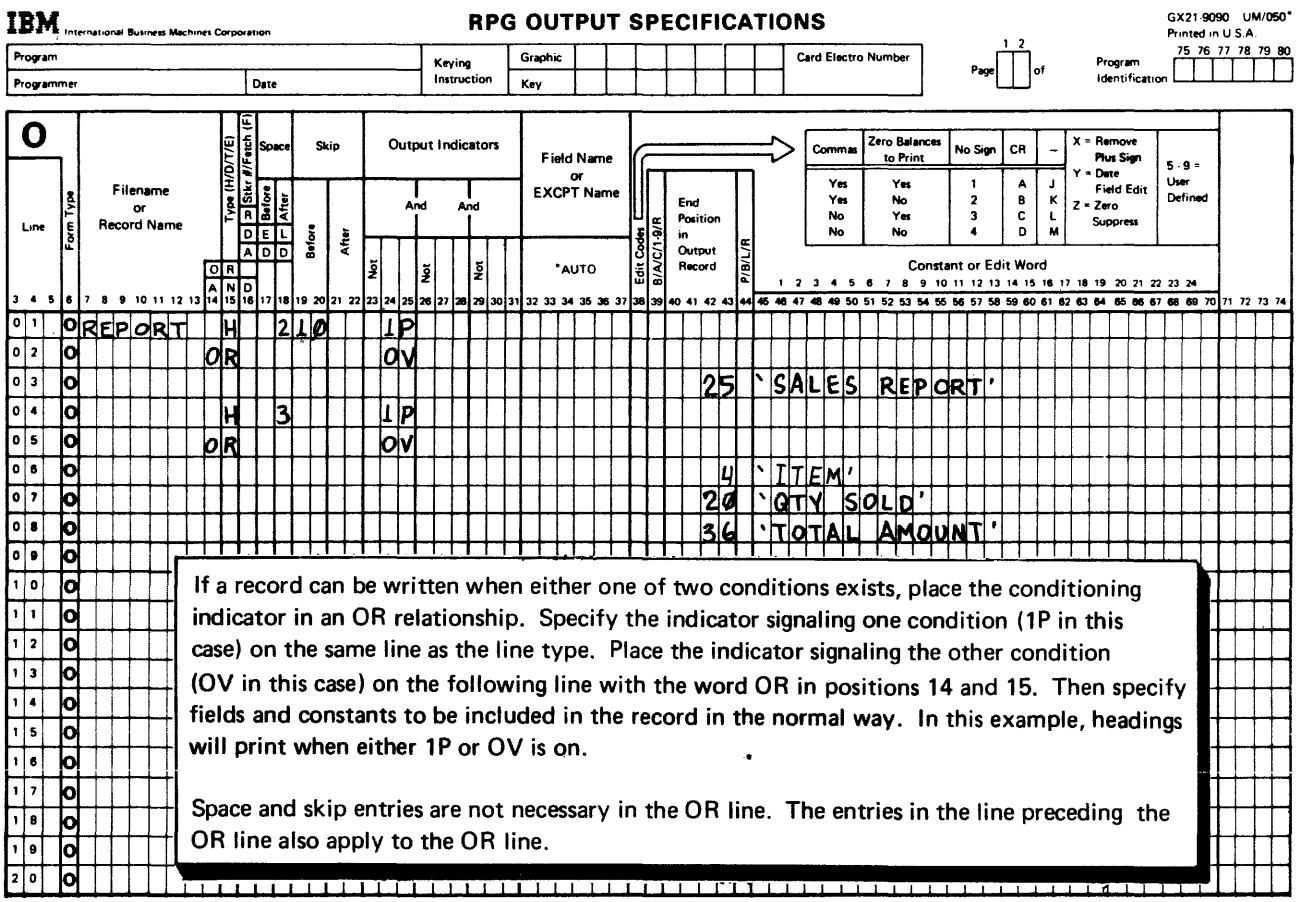


Figure 15. Spacing after Printing

Using Overflow and 1P Indicators Together

The overflow indicator is most often used with other indicators. However, we will discuss the use of overflow indicators only with the 1P indicator. Both 1P and OV cannot be on at the same time.

If you want headings on all pages of a report, you can use both the 1P indicator and an overflow indicator. 1P causes headings to print on the first page; the overflow indicator causes them to print on all succeeding pages. If a record should be printed when either one or another condition occurs (either 1P or OV is on), you can specify indicators in an *OR relationship*.



LAST RECORD (LR) INDICATOR

The last record (LR) indicator is associated with end-of-program procedures. The program uses LR to indicate that the last data record has been read and that end-of-program processing is to take place.

Use of the last record (LR) indicator is optional. When LR is not used in specifications, the compiler automatically supplies end-of-program instructions. If you use LR, you are indicating that certain operations, such as printing a total count of all records read, must be done after all input records are processed.

Program Cycle Operations

Figure 16 shows the operations in the program cycle associated with the last record indicator. RPG is set up so that it senses an end-of-file record containing some identifying information to indicate end of the data file. Each data file has an end-of-file record.

Whenever a record is read, the program checks to see if all records in a file have been processed. If all records in all the files have been processed, the program sets on all control level indicators L1 through L9. It also sets on the LR indicator to indicate that all records have been processed. All total operations (those conditioned by LR and L1 through L9) are performed. After total operations have been done, the program checks to see if LR is on. If it is, processing stops.

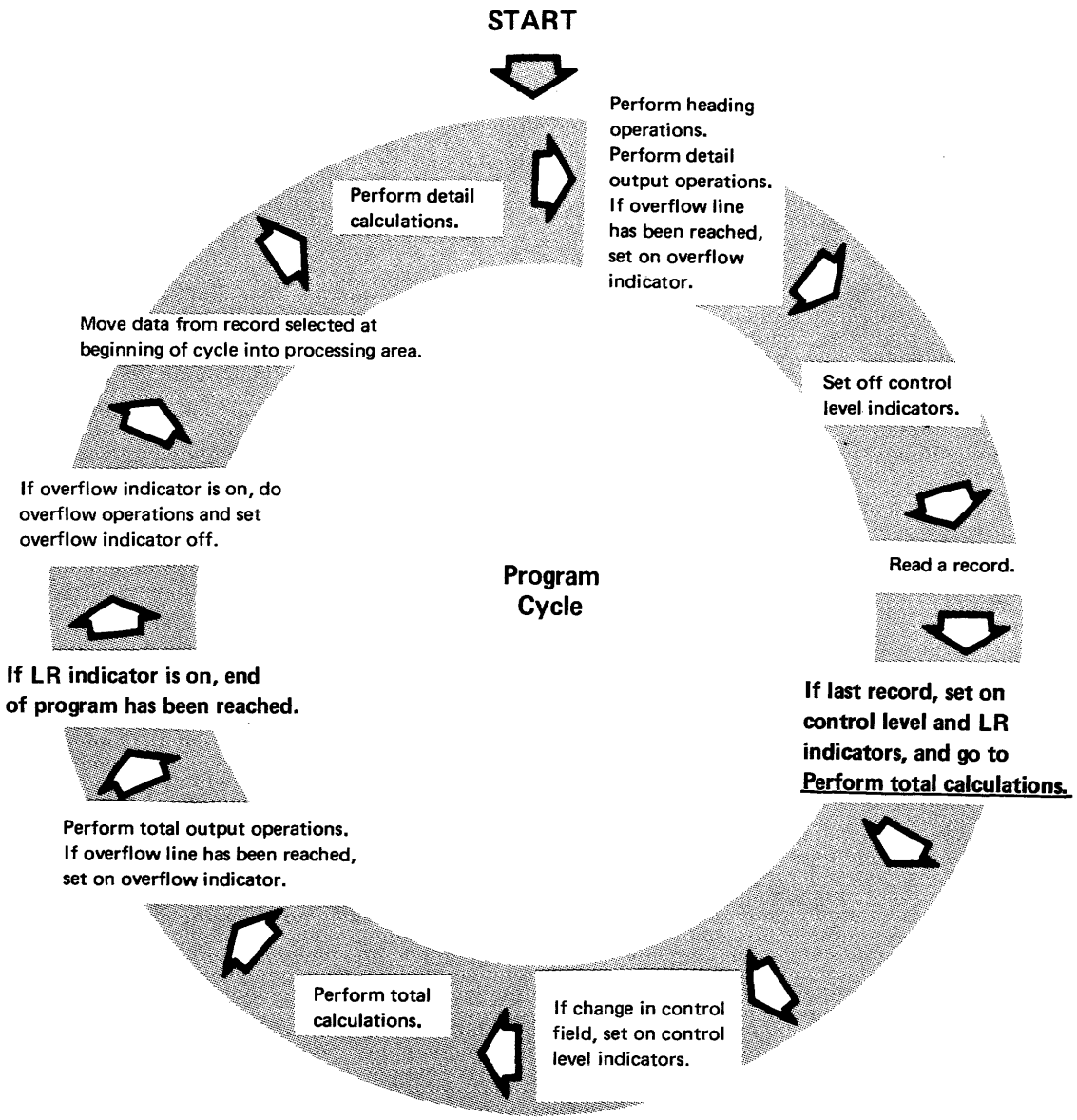
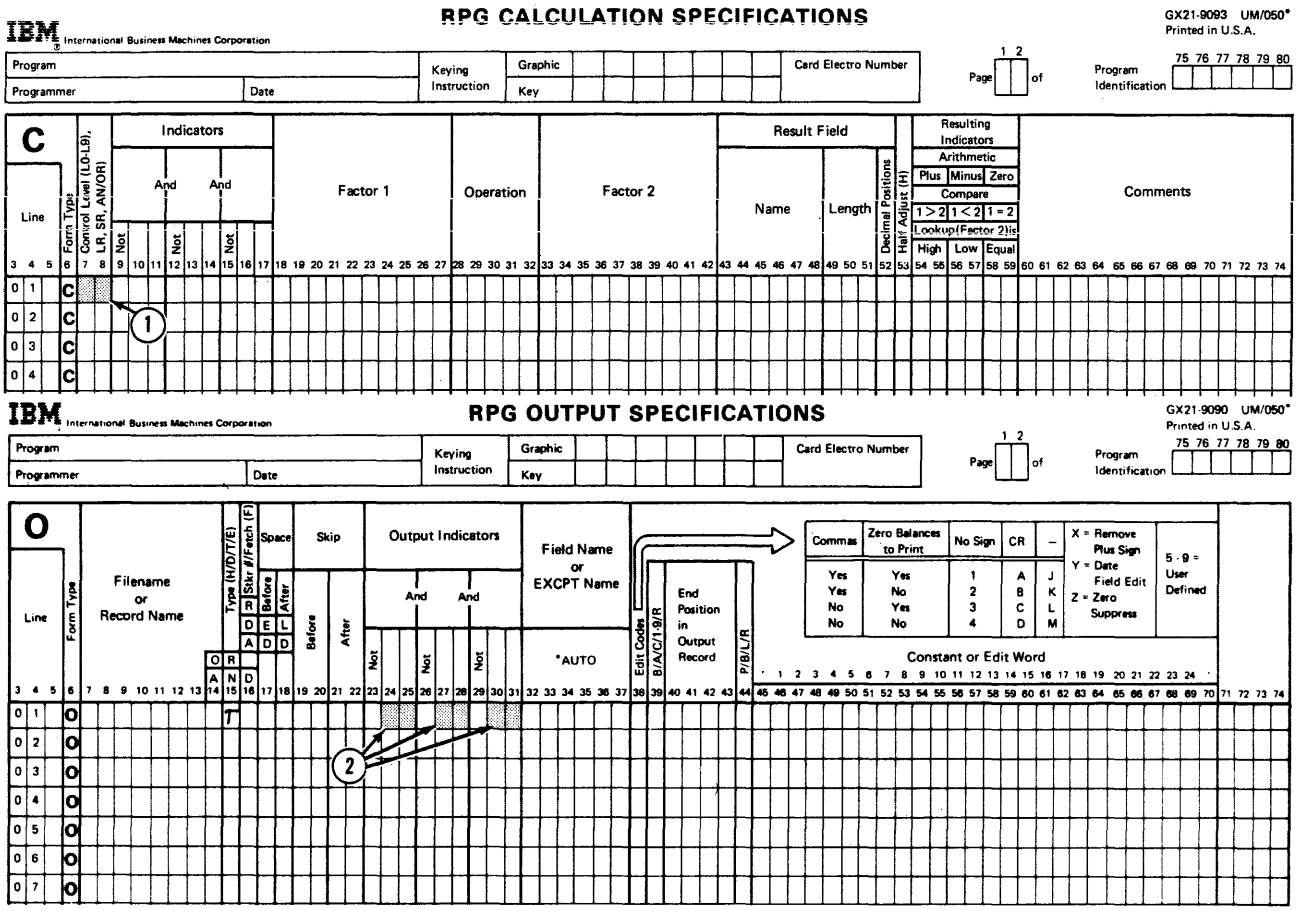


Figure 16. Program Cycle Operations for Last Record (LR) Indicator

The LR indicator is specified by an LR on the Calculation Specifications form or Output Specifications form. This entry specifies which operations are to be done after the last record is processed:



The LR indicator is specified at (1) to tell the system which calculations are to be done after the last record is processed. The LR indicator is specified at (2) to tell the system which output operations are to be done after the last record is processed.

EXAMPLE 4 (TRNREG): USING FIRST PAGE, OVERFLOW, AND LAST RECORD INDICATORS TO PRINT HEADINGS AND TOTALS

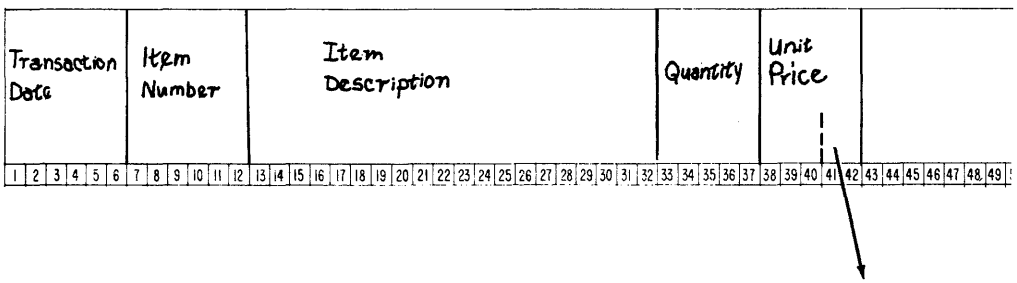
Program Definition

Print a weekly sales transaction report that lists daily transactions, total sales for the day, and total sales for the week. This report is similar to the one created in Example 3. The only difference is the addition of headings and final total.

The report title and column headings are printed on every page of the report. All items sold each day are listed. Item number, item description, quantity sold, unit price, and sales amount are included for every item. The date is printed for the first transaction in each group. After all transactions for a day are listed, the daily sales amount is printed. A final total of all daily sales is printed at the end of the report.

Program Requirements

Input: Sales transaction file consisting of 96-character records. Records are arranged in ascending order by date. The format of the input records is shown on this record layout form:



Processing:

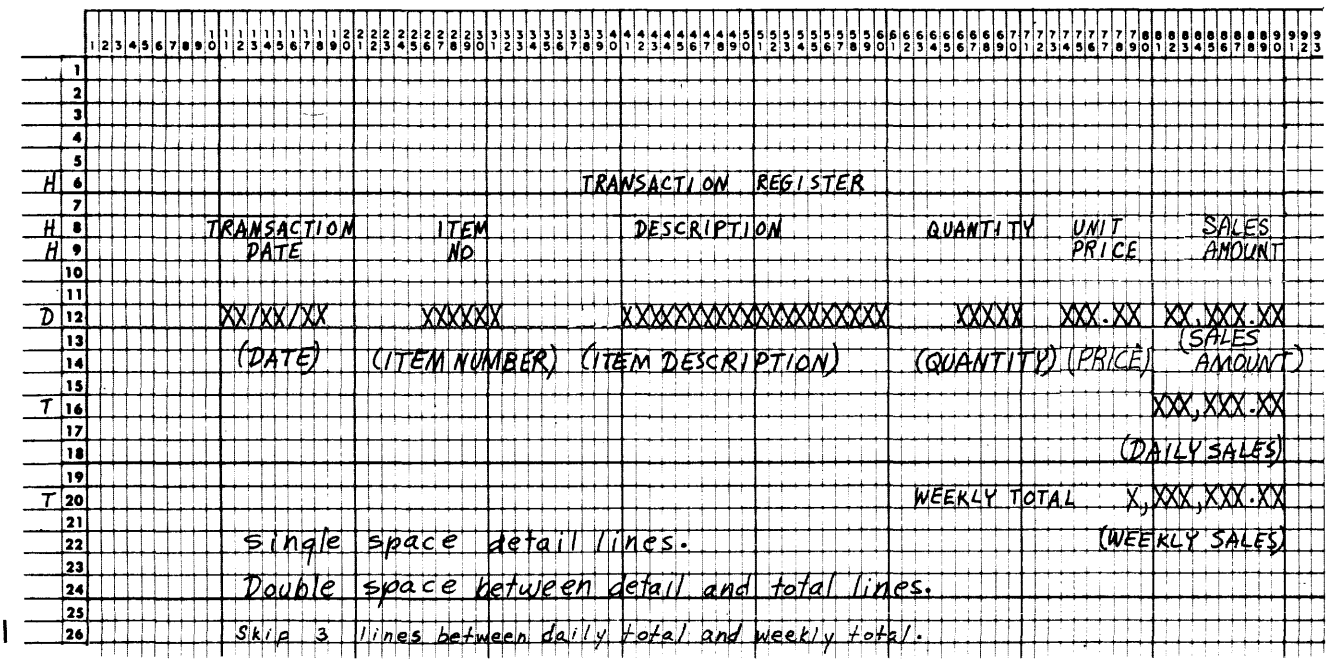
- Multiply quantity times unit price to find sales amount.
- Accumulate sales amount to find total item sales per day.
- Accumulate total daily sales to find total weekly sales.

Output: A printed transaction register:

TRANSACTION REGISTER

TRANSACTION DATE	ITEM NO	DESCRIPTION	QUANTITY	UNIT PRICE	SALES AMOUNT
7/23/--	413010	CH001 BOX 100A FLUSH	10	4.90	49.00
	412146	CH148 BREAKER 15A	100	.89	89.00
	411116	1500 TWIN SOCKET B	500	1.12	560.00
					698.00
7/24/--	503029	MOTOR 1/2 HP 60 CYC	2	146.78	293.56
	327802	TERMINAL CLIP	100	5.12	512.00
	326013	TERMINAL BAR	100		
7/27/--	321071	2-SPEED SAW	1	28.44	28.44
	325781	SATIN-CUT DADO SET	1	39.50	39.50
	412146	CH143 BREAKER 15A	50	.89	44.50
	573022	6-VOLT POWER BATTERY	2	14.45	28.90
					141.34
			WEEKLY TOTAL		2,593.70

This printer spacing chart shows how the report is formatted:



RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation — GX21-9092 UM:050* Printed in U.S.A.

Program: Transaction Register | Programmer: John Doe | Date: 1/10/-- | Keying Instruction | Graphic | Key | Card Electro Number

Page 01 of __ | Program Identification: TRNREG

File Description Specifications

The only new entry here is the OV indicator. Headings are to be printed at the top of every page. Overflow must be controlled in order to specify exactly where headings are to print. The overflow indicator is, therefore, assigned to the printer file to tell the compiler that it should not handle overflow automatically.

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D/F	E	A/D	F/V/S/M/D/E	Block Length	Record Length	L/R	A/P/I/K	I/X/D/T/R/ or 2	Overflow Indicator	Key Field Starting Location	External Record Name	K	Option	Entry	A/	R/
02	F	TRANS	I	P			F		96											
03	F	REPORT	O				F		96				OV		PRINTER					
04	F																			

RPG INPUT SPECIFICATIONS

IBM International Business Machines Corporation — GX21-9094 UM/060* Printed in U.S.A.

Program: Transaction Register | Programmer: John Doe | Date: 1/10/-- | Keying Instruction | Graphic | Key | Card Electro Number

Page 02 of | Program Identification: TRNREG

No new entries are made on the Input Specifications form.

Line	Form Type	Filename or Record Name / Data Structure Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / 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	To	Data Structure: Occurs n Times	Length	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	TRANS				01																									
02	I																				1	6			00	DATE	L1				
03	I																				7	12				ITEMNO					
04	I																				13	32				DESC					
05	I																				33	37			00	QTY					
06	I																				38	42			2	PRICE					

RPG CALCULATION SPECIFICATIONS

IBM International Business Machines Corporation — GX21-9093 UM/050* Printed in U.S.A.

Program: Transaction Register | Programmer: John Doe | Date: 1/10/-- | Keying Instruction | Graphic | Key | Card Electro Number

Page 03 of | Program Identification: TRNREG

Lines 01 and 02 are the same as for Example 3. The addition coded on line 03, conditioned by L1, will be done only when a control break occurs. Once daily sales have been accumulated for a group (L1 is on), daily total can be added to the field WEEKTO used for accumulating weekly total. (Weekly total could also have been obtained by adding EXTCST to WEEKTO for every record read.)

Line	Form Type	Control Level (L0-L9), LR, SR, AN/OR	Indicators: 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				QTY	MULT	PRICE	EXTCST	7	2											
02	C				EXTCST	ADD	DAYTOT	DAYTOT	8	2											
03	C	L1			DAYTOT	ADD	WEEKTO	WEEKTO	9	2											
04	C																				
05	C																				
06	C																				
07	C																				
08	C																				
09	C																				
10	C																				
11	C																				
12	C																				
13	C																				
14	C																				

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050* Printed in U.S.A.

Program: Transaction Register | Programmer: John Doe | Date: 1/10/--
Keying Instruction | Graphic | Key | Card Electro Number
Page 04 of | Program Identification: TRNREG

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Stkr # / Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And And Not	Field Name or EXCPT Name	Edit Codes	B/A/C/Z/1-9/R	End Position in Output Record	P/B/L/R	Commas	Zero Balances to Print	No Sign	CR	–	Constant or Edit Word
01	O	REPORT	H		2	06			1P											
02	O		OR						OV											
03	O												58							'TRANSACTION REGISTER'
04	O		H		1				1P											
05	O		OR						OV											
06	O												30							'TRANSACTION ITEM'
07	O												52							'DESCRIPTION'
08	O												78							'QUANTITY UNIT'
09	O												89							'SALES'
10	O		H		3				1P											
11	O		OR						OV											
12	O												29							'DATE NO'
13	O												90							'PRICE AMOUNT'
14	O		D		1				01											
15	O								L1	DATE	Y		18							
16	O																			

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050* Printed in U.S.A.

Program: Transaction Register | Programmer: John Doe | Date: 1/10/--
Keying Instruction | Graphic | Key | Card Electro Number
Page 05 of | Program Identification: TRNREG

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Stkr # / Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And And Not	Field Name or EXCPT Name	Edit Codes	B/A/C/Z/1-9/R	End Position in Output Record	P/B/L/R	Commas	Zero Balances to Print	No Sign	CR	–	Constant or Edit Word
01	O									ITEMNO			31							
02	O									DESC			60							
03	O									QTY	Z		70							
04	O									PRICE	3		79							
05	O									EXTCST	1		90							
06	O		T		1	2			L1											
07	O									DAYTOT	1	8	90							
08	O		T		2				LR											
09	O												74							'WEEKLY TOTAL'
10	O									WEEKTO	1		90							
11	O																			
12	O																			
13	O																			
14	O																			
15	O																			
16	O																			
17	O																			
18	O																			
19	O																			
20	O																			

Six different lines are to be printed, so six are described. The three heading lines should be printed either when 1P is on (first page) or when OV is on (overflow has occurred and forms have advanced). Thus, each heading is conditioned by 1P and OV used in an OR relationship. Notice that the first heading line has the skip specification. This skip must be made before forms will advance.

The weekly total is printed only once on the report, after all records have been processed. It is conditioned by LR. A constant is specified for this total record so that the words WEEKLY TOTAL print as indicated on the printer spacing chart.

RECORD IDENTIFYING INDICATORS

In the programs discussed so far, we have assumed that all records in the input file were alike. They did not necessarily contain the same information, but they had the same fields and the same kind of information in each field. They were of the same *type*.

However, in real programs, data files do not contain records of only one kind. Files often contain many kinds of records with different fields and different information. When using more than one kind of record in a program, you must have a way of telling the system what operations (calculations and output) you want done for each record read. Record identifying indicators are used for this purpose.

Program Cycle Operations

Figure 17 shows program cycle operations associated with record identifying indicators. A record identifying indicator is set on right after a record is read and is set off before the next record is read.

Normally, record identification indicators condition detail calculations and detail output operations because detail operations are done for the record just read (the one associated with the record identifying indicator). On the other hand, total operations are not performed for any particular record; they are done after detail records are processed.

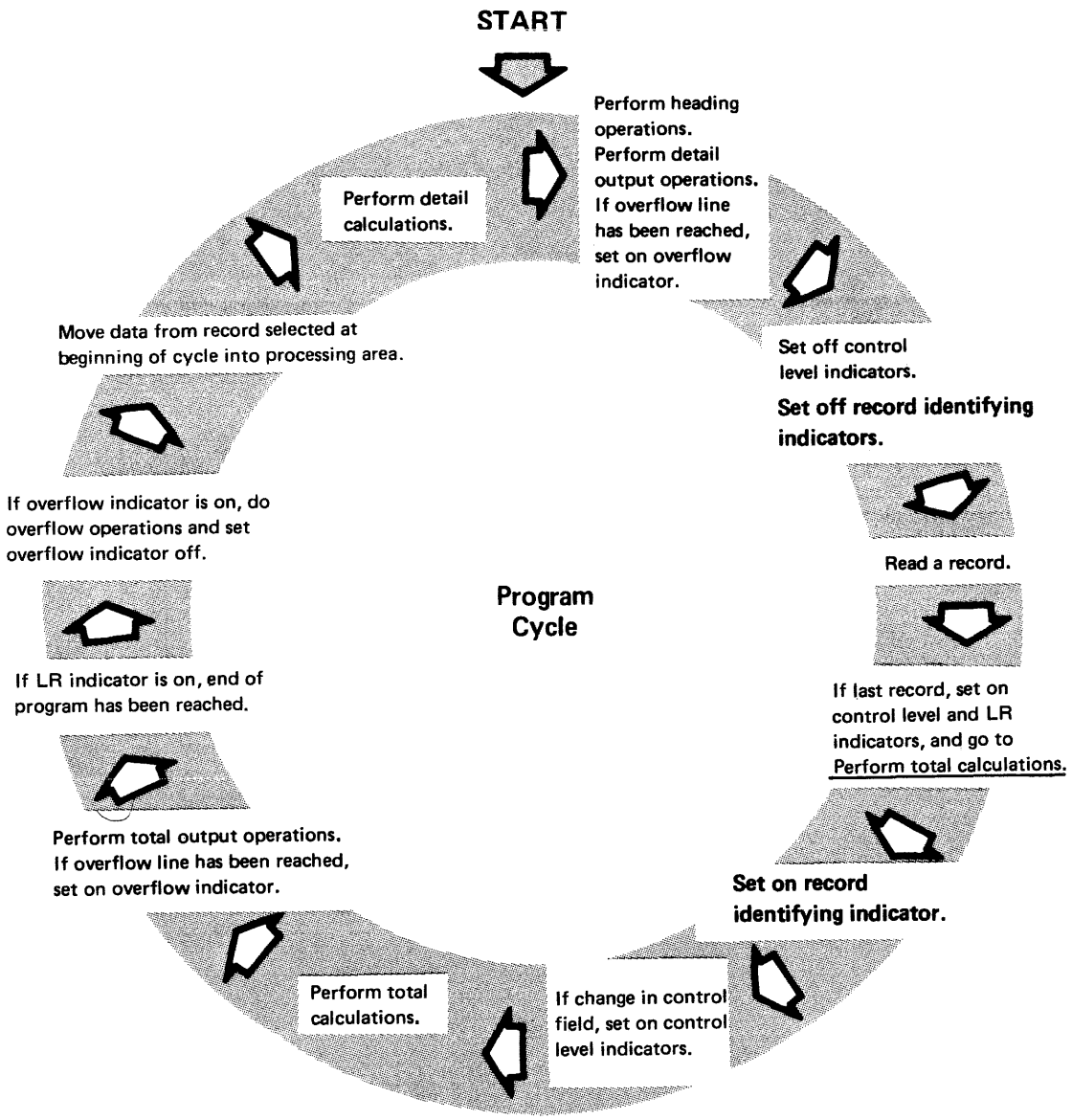
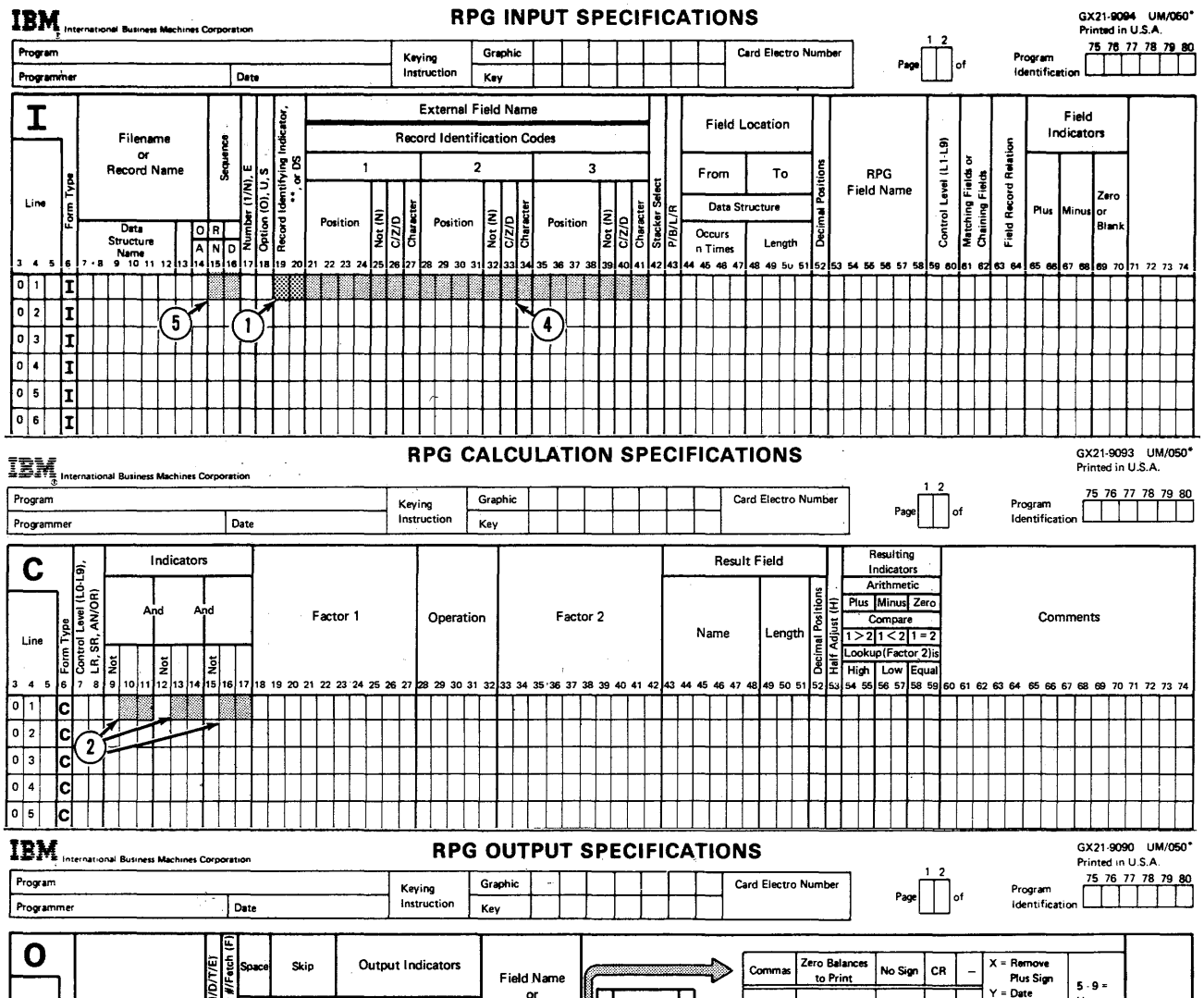


Figure 17. Program Cycle Operations for Record Identifying Indicators

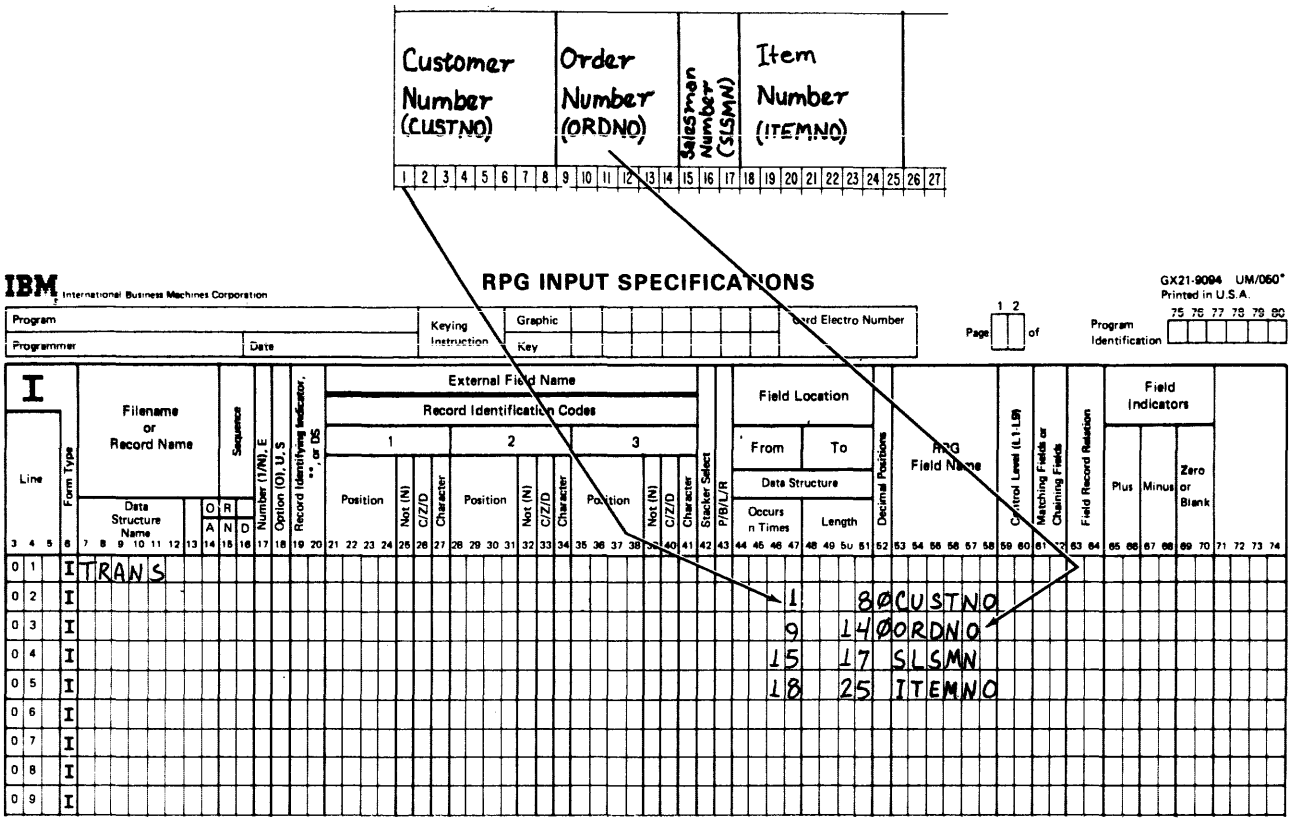
RPG Specifications

These are the RPG specifications you use when a file contains more than one kind of record:



Assign a record identifying indicator (a two-digit number from 01 through 99) to each record at (1). Then use that same indicator to condition calculations at (2) and detail output operations at (3). The calculations and detail output operations must be done for only that kind of record. Identify each kind of record by describing a code that identifies it at (4). If records must be in a certain order, indicate that sequence at (5).

As we said earlier, you must describe each kind of record in your input file. This description includes the names of all fields used from the record, the location of the fields, and the type of data in the fields:



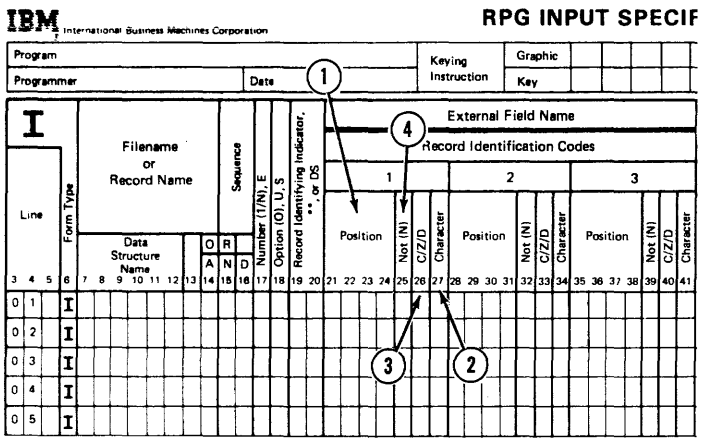
When you describe fields from each record in the input file, you can give a unique field name to each field. If, however, two or more records contain identical fields, you can assign the same name to the field in each kind of record. Only one storage area would be assigned the same field from several records, but it does not matter because the information in the field is the same.

You must do more, however, than merely list the fields in all records, because the RPG compiler would not know which fields were in each kind of record. Furthermore, it would not know which record it was reading. To give the compiler a means of identifying records, you use record identification codes and record identifying indicators.

Specifying Record Identification Codes

When you create records, you should include an identification code in each one. For example, to identify an item transaction record, you might place the code TR somewhere in that record. You can use any combination of letters and numbers for the identification code, and you can place the code in any record positions.

When you describe the record on the Input Specifications form, you use positions 21 through 41 to describe the record's identification code and to indicate where the code is located in the record:



For each character in the code you must specify:

1. Where in the record the character is found.
2. What the character is.
3. The letter C to indicate character.
4. The letter N before the letter C if the code requires that a character must *not* be present.

You can specify a code of up to three characters on one line. If your code contains more than three characters, use the next line and the word AND in positions 14 through 16. Figure 18 shows some examples of how to specify record identifications codes.

To produce in the printed report the record identification codes listed below, write the corresponding input specifications shown on the form at the right:

T in position 80 ——→

ST in positions 1 and 2 ——→

1A in positions 95 and 96 ——→

No A in position 95; 1 in position 96 ——→

ABCD in positions 1 through 4 ——→

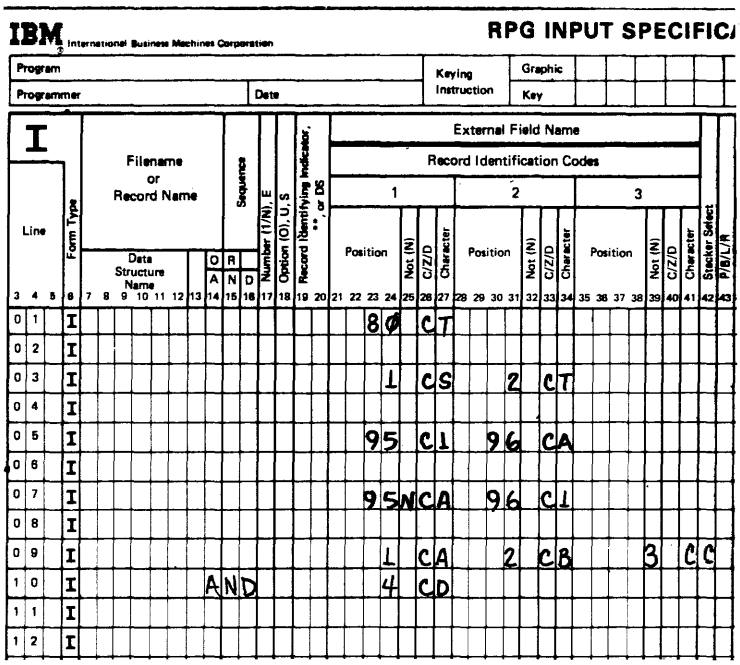
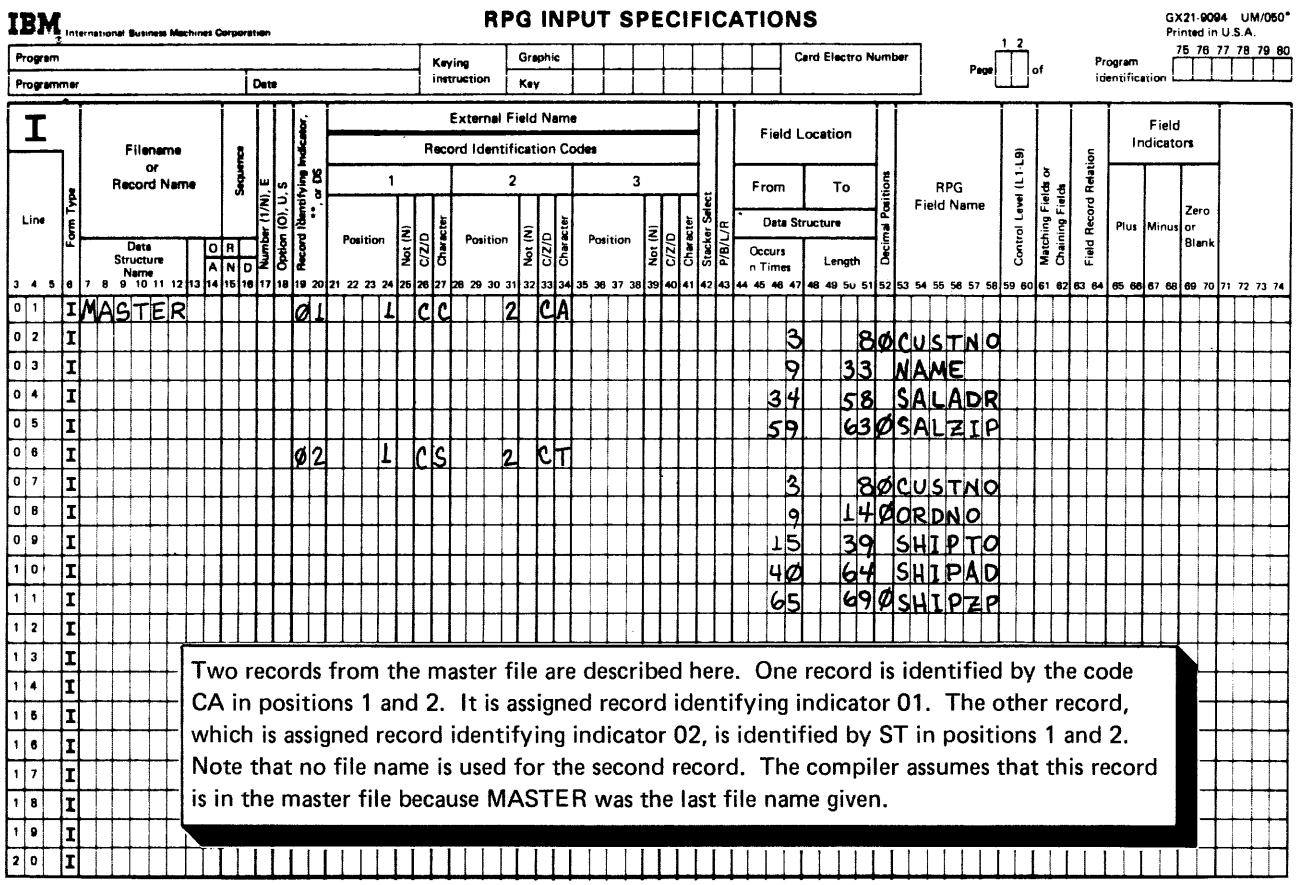


Figure 18. Valid Specifications for Record Identification Codes

Specifying Record Identifying Indicators

You specify a record identifying indicator in positions 19 and 20 for each kind of record used in the program. Record identifying indicators are numbered 01 through 99. Use a different number for each record type.

A record identifying indicator is specified on the same line as the identification code. All fields for the record are then listed, starting one line below the identification code and identifying indicator. The file name need be specified only once, on the specification line describing the first record in the file.



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying instruction | Graphic | Key | Card Electro Number | Page 1 2 of | Program identification 75 76 77 78 79 80

Line	Form Type	Filename or Record Name	Data Structure Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	P/B/L/R	Field Location: From	To	Data Structure: Occurs n Times	Length	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	MASTER					01	L		C	C	2		CA																			
02	I																							3	8		CUSTNO						
03	I																							9	33		NAME						
04	I																							34	58		SALADR						
05	I																							59	63	Ø	SALZIP						
06	I						02	L		C	S	2		CT																			
07	I																							3	8	Ø	CUSTNO						
08	I																							9	14	Ø	ORDNO						
09	I																							15	39		SHIPTO						
10	I																							40	64		SHIPAD						
11	I																							65	69	Ø	SHIPZP						
12	I																																
13	I																																
14	I																																
15	I																																
16	I																																
17	I																																
18	I																																
19	I																																
20	I																																

Two records from the master file are described here. One record is identified by the code CA in positions 1 and 2. It is assigned record identifying indicator 01. The other record, which is assigned record identifying indicator 02, is identified by ST in positions 1 and 2. Note that no file name is used for the second record. The compiler assumes that this record is in the master file because MASTER was the last file name given.

After reading a record, the program checks the identification codes to determine which record it has read. When it finds a match between the code on the record and the code stored from the Input Specifications form, it turns on the record identifying indicator associated with that record. This is the program's way of remembering which record it read.

Record identifying indicators can be used to condition calculations, output records, or output fields. In this way, the program performs the proper operations for each record.

When you use record identifying indicators to condition calculation and output operations, you are assured that these operations are done only for appropriate records. If you do not use these indicators, operations are done on all records.

Record identifying indicators should be assigned even when there is only one kind of record in a file. If you do not use them, the compiler prints a message telling you that record identifying indicators have not been assigned.

The record identifying indicator should also be used to condition detail output records. This indicator prevents detail lines from being written on the first cycle. If the detail line is not conditioned by a record identifying indicator, any constants you specified on the detail lines are printed on the first cycle before the first record is read.

Specifying Record Sequence

Sometimes records must be in a particular order within a group, but at other times the order makes no difference. When records need not be in a particular order, enter two alphabetic characters in positions 15 and 16. Different alphabetic characters can be used for each record:

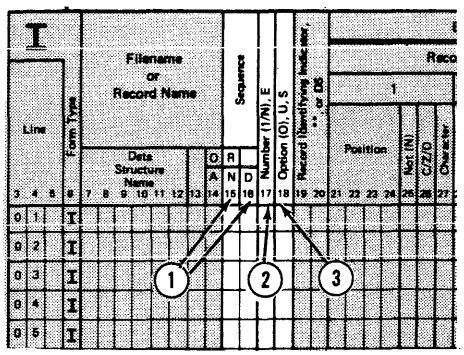
IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

Program		Keying Instruction	Graphic	Card Electro Number
Programmer	Date		Key	

Line	Form Type	Filename or Record Name	Data Structure Name	Sequence (O, R, A, N, D)	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / 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	Data Structure: Occurs n Times	Data Structure: Length	Decimal Positions	RPG Field Na
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 5u 51	52	53 54 55 56		
01	I	INPUT		AA			10	1		C	A																
02	I																										
03	I																										
04	I			BB			20	1		C	B																
05	I																										

If the records must be in a specific sequence, you must make numeric entries in positions 15 and 16. Positions 17 and 18 provide additional information about the records.



(1) Positions 15 and 16 (Sequence) contain numeric entries (01 through 99). The sequence entries must be in numeric order.

(2) Position 17 (Number) tells how many records of this kind to expect in the group:

1 = One and only one
N = One or more

(3) Position 18 (Option) indicates whether the record is required in the group:

Blank = Record is required
O = Record is optional

Positions 15 and 16 should have an entry for every record specified. If there is only one kind of record in a file, the entry should be character. If you fail to use a sequence entry, the compiler prints a message saying that no entry was made in positions 15 and 16.

EXAMPLE 5 (STOKST): USING RECORD IDENTIFYING INDICATORS TO PROCESS DIFFERENT KINDS OF RECORDS

Program Definition

Print a stock status report. This report is printed whenever the inventory is updated. It gives detailed information on all active merchandise. The first line for each item in the report shows standard descriptive data for the item: item number, item description, quantity on hand, and quantity on order. This information is taken directly from the input record.

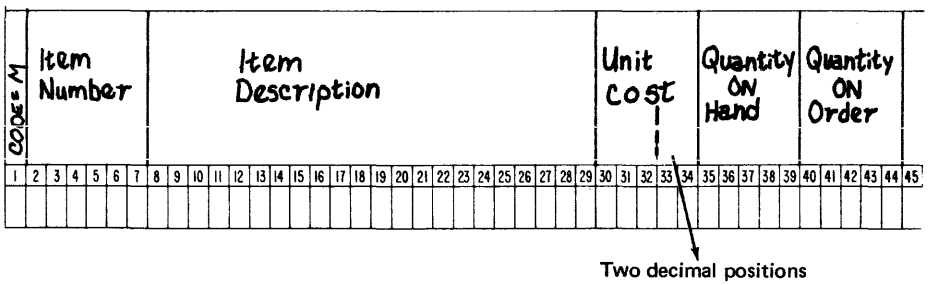
Subsequent lines give the detail on current transactions involving the item: sales to customers and receipts from suppliers. This information is also taken directly from input records.

Quantities remaining on hand and on order are calculated for each item and printed after all transactions for the item are listed.

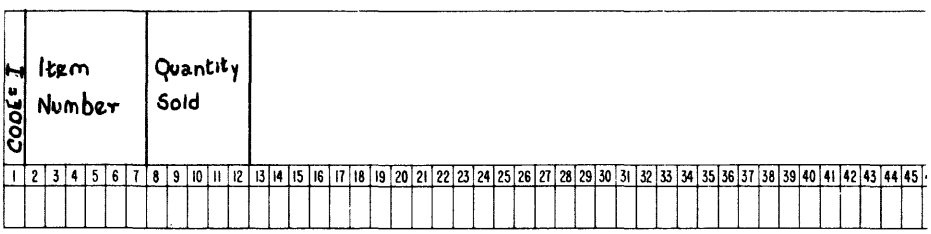
Program Requirements

Input: An inventory file consisting of three different types of records. Formats of the three record types are:

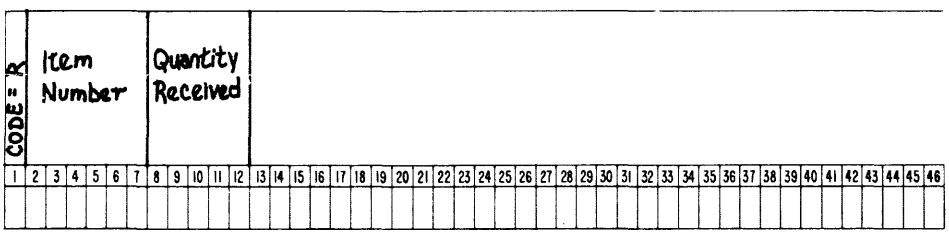
Name ITEM MASTER RECORD



Name ISSUE RECORD



Name RECEIPT RECORD



The file is organized in ascending order by item number. For each item, one master record is required. Issue and receipt records are optional. When present, however, there may be any number of each. Records for each item must be in this order:

1. Item master
2. Issue(s)
3. Receipt(s)

Processing:

- Find total number of each item sold. To do this, perform the calculation ISSUE + TOTAL ISSUE = TOTAL ISSUE for each issue record.
- Find total number of each item received. Perform the calculation RECEIPT + TOTAL RECEIPT = TOTAL RECEIPT for each receipt record.
- When all transaction records for one item have been read, find new quantity on hand (ON HAND + TOTAL RECEIPT - TOTAL ISSUE = NEW ON HAND) and new quantity on order (ON ORDER - TOTAL RECEIPT = NEW ON ORDER).

Output: A printed stock status report:

STOCK STATUS REPORT

ITEM NO	DESCRIPTION	QUANTITY ON HAND	QUANTITY ON ORDER	TRANSACTION QUANTITY
411116	B500 TWIN SOCKET BLUE	458	500	
	ISSUE			50
	RECEIPT			500
		908**	0**	
411122	B506 SOCKET ADAPT BRN	325	0	
	ISSUE			20
	ISSUE			38
	ISSUE			10
		257**	0**	
411173	C151C SIL SWITCH IVORY	50	150	
	RECEIPT			150
		200**	0**	
411254	A210 PULL CORD GOLD	62	75	
	ISSUE			16
	ISSUE			30
		16**	75**	

This printer spacing chart shows the format of the report:

H	6	STOCK STATUS REPORT
H	9	ITEM NO DESCRIPTION QUANTITY ON HAND QUANTITY ON ORDER TRANSACTION QUANTITY
D	12	XXXXXX XXXXXXXXXXXXXXXXXXXX XX,XXX XX,XXX
D	13	ISSUE XX,XXX
D	14	RECEIPT XX,XXX
T	16	XXX,XXX** XXX,XXX**
	17	(NEW (NEW
	18	ON HAND) ON ORDER)
	23	Single space detail lines.
	25	Double space between detail and total lines.
	27	Print headings on every page of the report.
	29	Use asterisks to emphasize New On Hand and New On Order.

Program Specifications

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9092 UM/050*
Printed in U.S.A.

Program: Stock Status
Programmer: John Doe
Date: 1/10/--
Keying Instruction
Graphic
Key
Card Electro Number
Page 01 of __
Program Identification: STOKST

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

H

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	Reserved	Currency Symbol	Date Format	Date Edit	Inverted Print	Reserved	Number of Print Positions	Sequence
01	H													

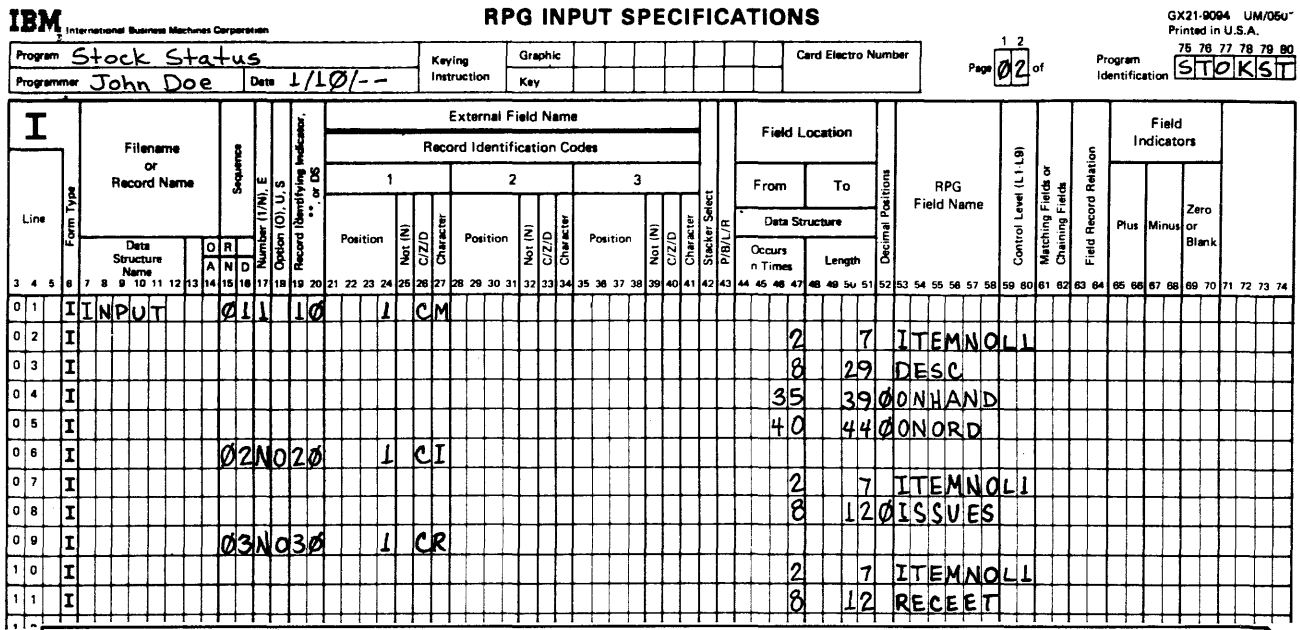
Two files are used for the program: one input and one output. In the program description given, the output file is a printer. The entry for the input device depends on your system.

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

F

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D/F	E	A/D	F/V/S/M/D/E	Block Length	Record Length	L/R	A/P/H/K	I/X/D/T/R/ 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	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8, UC	R/U/N
02	F	INPUT	I	P		F			64																					
03	F	OUTPUT	O			F			96					OV			PRINTER													
04	F																													



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM/050
Printed in U.S.A.

Program: Stock Status
Programmer: John Doe
Date: 1/10/--
Keying Instruction: Graphic / Key
Card Electro Number
Page 02 of
Program Identification: STOKST

Line	Form Type	Filename or Record Name	Data Structure Name	O (A)	R (N)	D	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name	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	Data Structure: Occurs n Times	Data Structure: Length	Decimal Positions	RPG 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	INPUT			01	1			10			1		CM																							
02	I																									2			7		ITEMNOLL						
03	I																									8			29		DESC						
04	I																									35			39	0	ONHAND						
05	I																									40			44	0	ONORD						
06	I				02	N	02	0				1		CI																							
07	I																									2			7		ITEMNOLI						
08	I																									8			12	0	ISSUES						
09	I				03	N	03	0				1		CR																							
10	I																									2			7		ITEMNOLL						
11	I																									8			12		RECEET						

All three kinds of records in the input file must be described. Because they are to be arranged in a certain sequence within a group, they are described in the order in which they will be read.

The first record in the group is the item master record (identified by M in position 1). The 01 in positions 15 and 16 indicates that this record is first. The record is required (position 18 is blank), and there must be only one per group (1 in position 17). It is assigned record identifying indicator 10. The fields are then described. Note that the record code appears as a field on the record layout form. However, on the Input Specifications form it is described not as a field but as the record identification code.

The second record in the group (02 in positions 15 and 16) is the issue record, identified by I in position 1. There may be several of these records per group (N in position 17). The record is optional (O in position 18). The record identifying indicator 20 is assigned. Fields in the record are then described.

The third record in the group is described as were the previous two.

ITEMNO field in the item master record is assigned as the control field. When it changes, all records for a particular item number have been processed and the L1 indicator is set on. Sequence checking is not related to the control field specification in any way. Sequence checking is based only on the sequence specification.

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050*
Printed in U.S.A.

IBM International Business Machines Corporation

Program Stock Status | Keying Instruction | Graphic | Key | Card Electro Number

Programmer John Doe | Date 1/10/--

Page 03 of __ | Program Identification STOKST

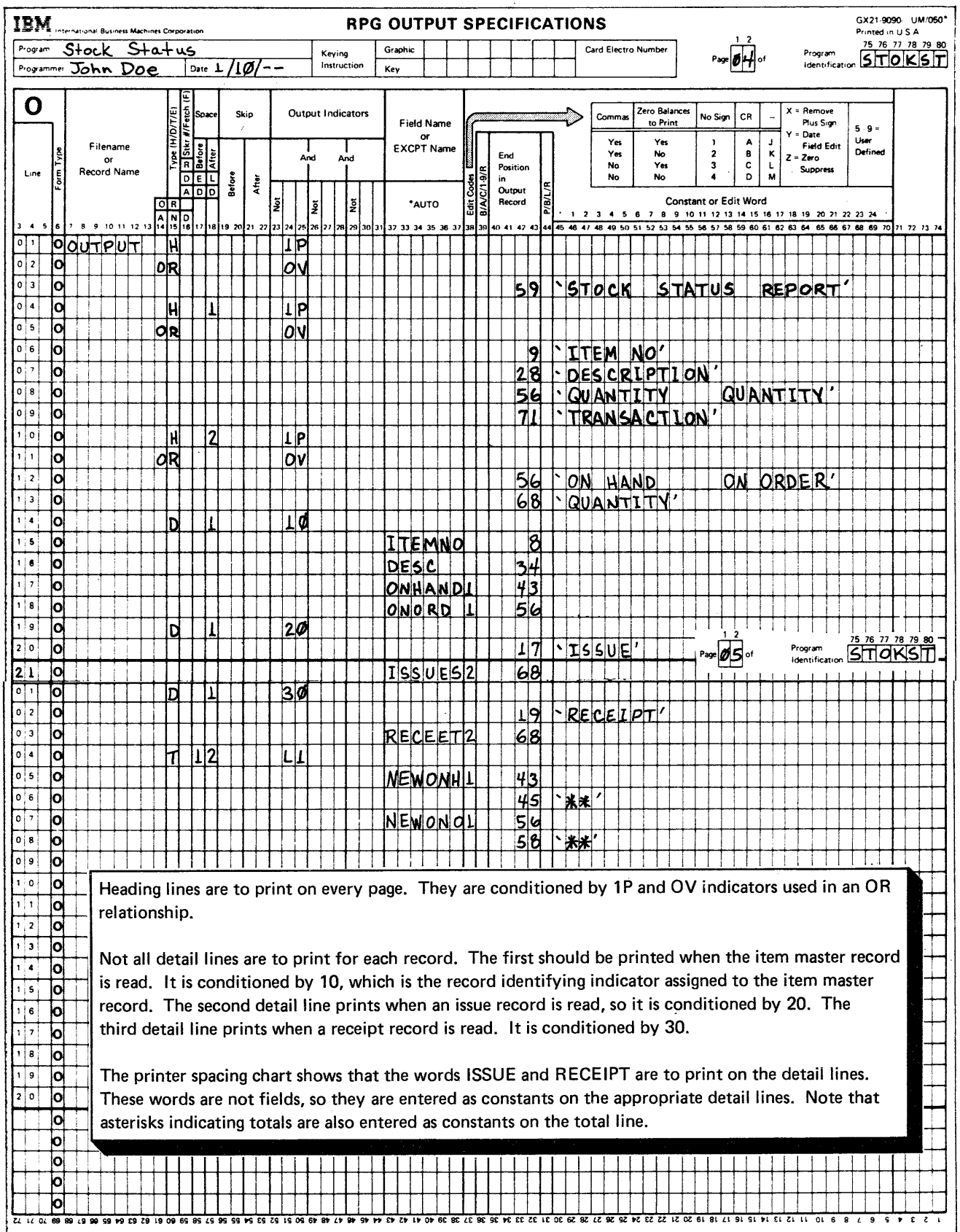
C

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
01	C		20	ISSUES	ADD	TOTISU	TOTISU	50				FIND TOTAL SOLD
02	C		30	RECEET	ADD	TOTREC	TOTREC	50				FIND TOTAL RECD
03	C	L1		ONHAND	SUB	TOTISU	NEWONH	60				FIND NEW TOTAL
04	C	L1		NEWONH	ADD	TOTREC	NEWONH					ONHAND
05	C	L1		ONORD	SUB	TOTREC	NEWONO	60				FIND NEW ONORDR
06	C	L1		TOTISU	SUB	TOTISU	TOTISU					CLEAR TOTISU
07	C	L1		TOTREC	SUB	TOTREC	TOTREC					CLEAR TOTREC
08	C											
09	C											

To update the quantity on hand and the quantity on order, total number issued (TOTISU) and total number received (TOTREC) for each item are needed. Quantity sold is found only in the issue record. Thus, the calculation to find TOTISU is done only when the issue record is read. Record identifying indicator 20 was assigned to the issue record. When 20 is on (an issue record has been read), we can calculate TOTISU. Thus, the operation (line 01) is conditioned by indicator 20. The operation to find TOTREC can be done only when a receipt record is read. The operation (line 02) is conditioned by 30, the record identifying indicator assigned to the receipt record.

Calculations to update the quantity on hand and the quantity on order are total operations and can be done only after all transaction records for the item have been processed. They are conditioned by L1, which is set on when a new item number is read.

The operations on line 06 and 07 must be used to clear the total issues (TOTISU) and total receipts (TOTREC) fields after the quantity on hand and the quantity on order have been calculated.



Heading lines are to print on every page. They are conditioned by 1P and OV indicators used in an OR relationship.

Not all detail lines are to print for each record. The first should be printed when the item master record is read. It is conditioned by 10, which is the record identifying indicator assigned to the item master record. The second detail line prints when an issue record is read, so it is conditioned by 20. The third detail line prints when a receipt record is read. It is conditioned by 30.

The printer spacing chart shows that the words ISSUE and RECEIPT are to print on the detail lines. These words are not fields, so they are entered as constants on the appropriate detail lines. Note that asterisks indicating totals are also entered as constants on the total line.

RESULTING INDICATORS

Sometimes your decision to do a certain operation is based on the result of a previous operation. Resulting indicators allow you to specify which operations you want done and the conditions under which the operations are to be done. Resulting indicators can be used to determine such things as:

- Whether a result is larger, smaller, or equal to a predetermined number.
- Whether a certain result is plus, minus, or zero.

Program Cycle Operations

Figure 19 shows the operations in the program cycle associated with resulting indicators. Resulting indicators are set when the associated calculation operation is performed. This means that resulting indicators can be set either at detail calculation time or at total calculation time.

Resulting indicators are not set off automatically. They change their setting only when a calculation is performed or when they are set off intentionally. For example, if a resulting indicator is set on by a detail calculation, it retains this setting until the next time it is used as a resulting indicator.

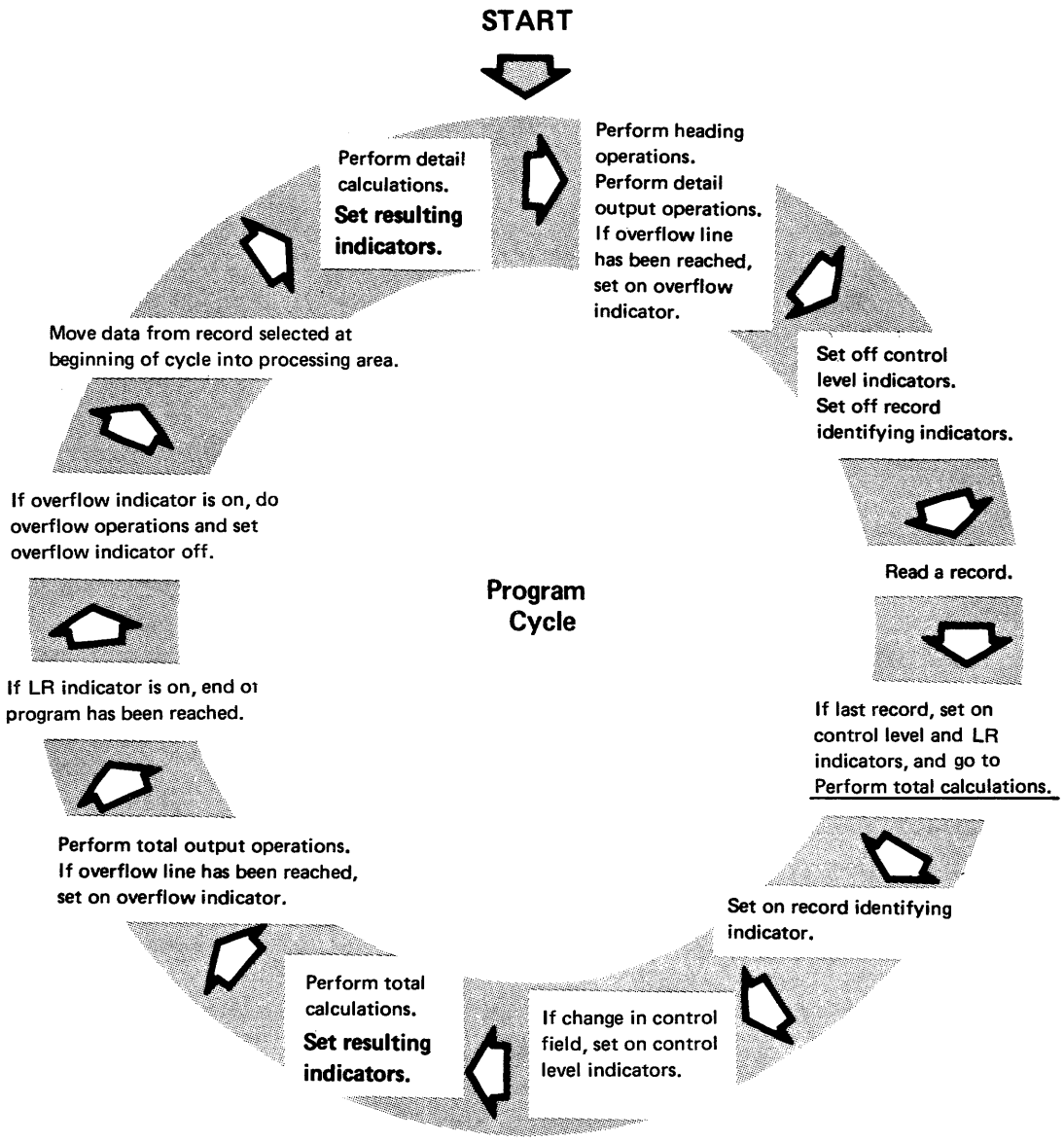
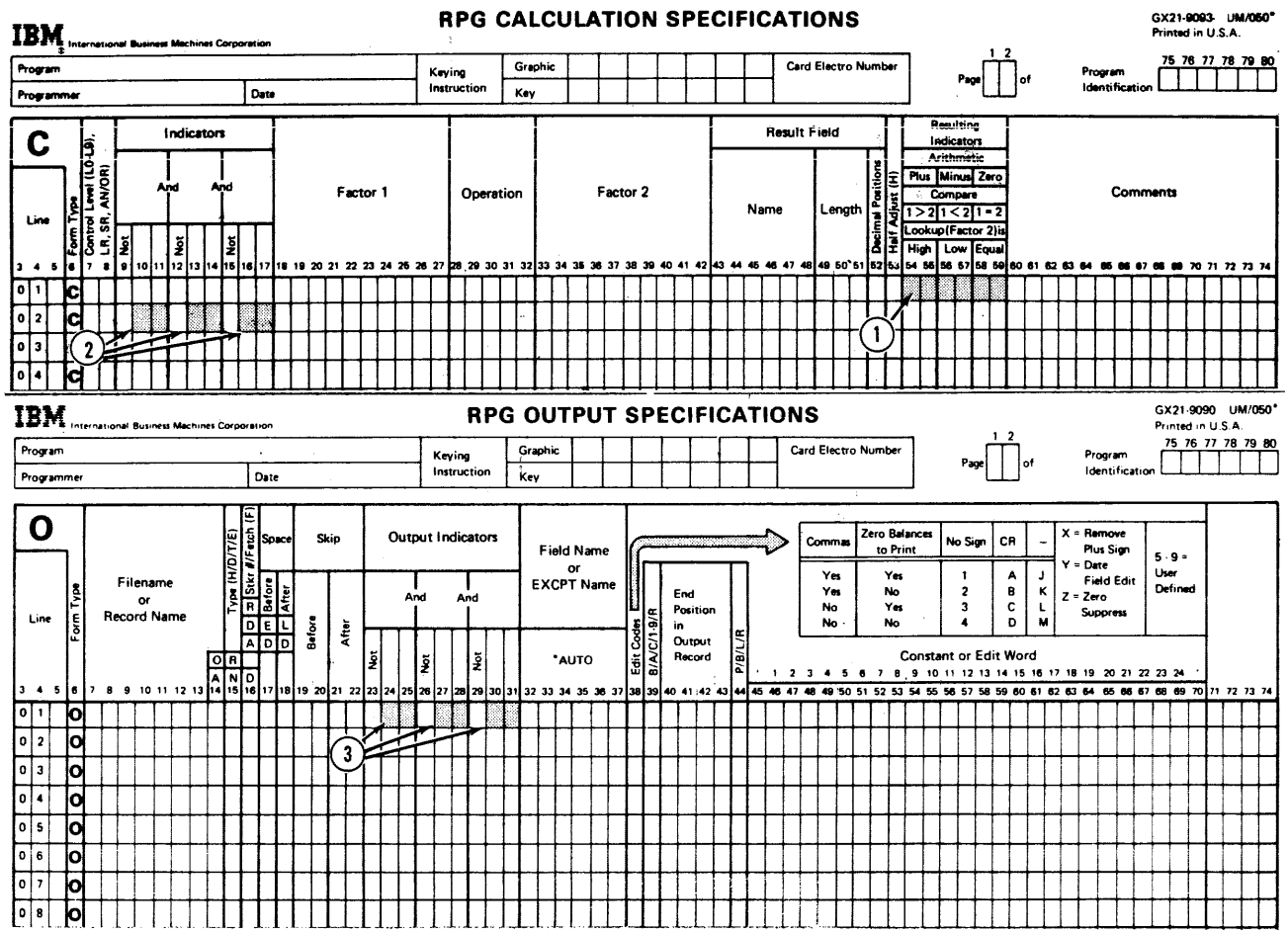


Figure 19. Program Cycle Operations for Resulting Indicators

The use of resulting indicators depends on the operation specified. If you want to determine whether the result field is larger, smaller, or equal to a certain number, you must use a compare (COMP) operation. If you want to determine if the result field is plus, minus, or zero, use an arithmetic operation (ADD, SUB, MULT, DIV). You can specify resulting indicators 01 through 99 on these specifications forms:



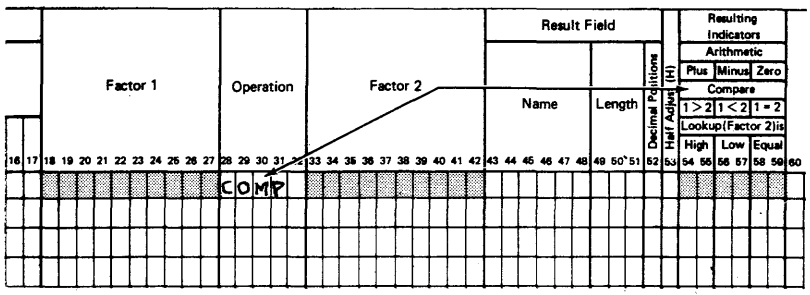
Resulting indicators are assigned at (1), then used to condition calculation operations at (2) and output operations at (3).

Resulting indicators and record identifying indicators are used in the same way to condition output and calculation specifications. You should not use the same entry as both a resulting indicator and a record identifying indicator.

Using the Compare Operation

In many programs you need to know whether a field is greater than, smaller than, or equal to another field. RPG has an operation code, COMP, that allows you to compare fields. The compare operation requires entries in these columns on the Calculation Specifications form:

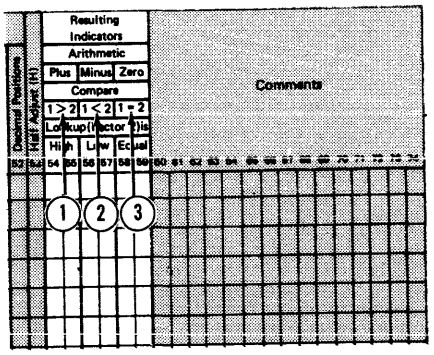
- Factor 1 (either a field name or a constant)
- Factor 2 (either a field name or a constant)
- Resulting indicators



When compared, Factor 1 and Factor 2 can be in one of three relationships:

- Factor 1 can be greater than Factor 2.
- Factor 1 can be less than Factor 2.
- Factor 1 can be equal to Factor 2.

You indicate that a test should be made to check for one, two, or all of these relationships by entering indicators in the appropriate columns:



① A resulting indicator entered in positions 54 and 55 tells the system to determine if Factor 1 is greater than Factor 2.

② A resulting indicator entered in positions 56 and 57 tells the system to determine if Factor 1 is less than Factor 2.

③ A resulting indicator entered in positions 58 and 59 tells the system to determine if Factor 1 is the same as Factor 2.

The test you specify is made each time the COMP operation is executed. However, the resulting indicator is set on only when the proper relationship exists. If you enter indicator 50 in positions 54 and 55 to test whether Factor 1 is greater than Factor 2, indicator 50 is set on only when Factor 1 is greater than Factor 2. If the relationship does not exist, the indicator is set off.

When testing for more than one condition, you can use the same or different indicators in these positions. If you intend to do different operations for each of the three conditions, enter a different resulting indicator to test for each condition on the Calculation Specifications form:

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050* Printed in U.S.A.

Program | Keying Instruction | Graphic | Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

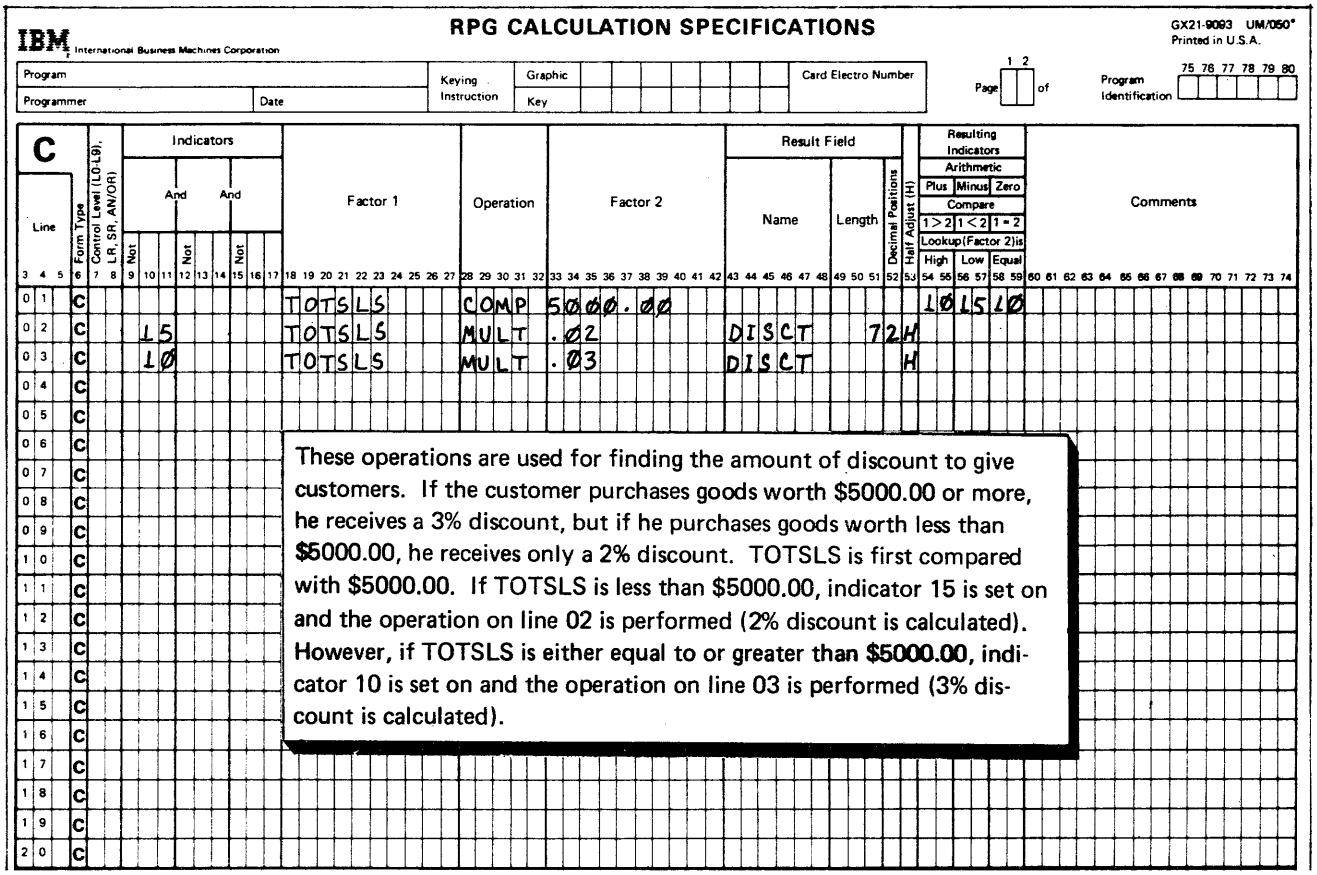
Programmer | Date

C

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				MAX	COMP	TOTAL											11	22	33	
02	C		11		TOTAL	MULT	.05	COMM		72	H										
03	C		22		TOTAL	MULT	.06	COMM			H										
04	C		33		TOTAL	MULT	.07	COMM			H										
05	C																				
06	C																				
07	C																				
08	C																				
09	C																				
10	C																				
11	C																				
12	C																				
13	C																				
14	C																				
15	C																				
16	C																				

In this operation, the contents of the field MAX is compared with the contents of the field TOTAL. If MAX is greater than TOTAL, indicator 11 is set on and the operation on line 02 is done. If MAX is less than TOTAL, indicator 22 is set on and the operation on line 03 is done. If MAX is equal to TOTAL, indicator 33 is set on and the operation on line 04 is done. Only the resulting indicator for the condition that is true is set on. The other resulting indicators are set off.

If you want to do the same operations when either one of two conditions exists (Factor 1 is greater than Factor 2; Factor 1 equals Factor 2), you could use the same indicator to test for both conditions on the Calculation Specifications form:



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050* Printed in U.S.A.

Program | Keying Instruction: Graphic / Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Programmer | Date

C

Line	Form Type	Control Level (L0-L9), LR, SR, AN/OR	Indicators: And / 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			TOTSLS	COMP	5000.00					10 15 10	
02	C		15	TOTSLS	MULT	.02	DISCT		72	H		
03	C		10	TOTSLS	MULT	.03	DISCT			H		
04	C											
05	C											
06	C											
07	C											
08	C											
09	C											
10	C											
11	C											
12	C											
13	C											
14	C											
15	C											
16	C											
17	C											
18	C											
19	C											
20	C											

These operations are used for finding the amount of discount to give customers. If the customer purchases goods worth $5000.00 or more, he receives a 3% discount, but if he purchases goods worth less than $5000.00, he receives only a 2% discount. TOTSLS is first compared with $5000.00. If TOTSLS is less than $5000.00, indicator 15 is set on and the operation on line 02 is performed (2% discount is calculated). However, if TOTSLS is either equal to or greater than $5000.00, indicator 10 is set on and the operation on line 03 is performed (3% discount is calculated).

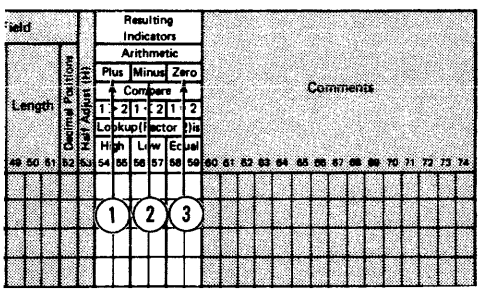
We explained previously that constants can be used in arithmetic calculation operations, but they must be numeric constants. In a COMP operation, however, constants can be either character or numeric. Rules for using character constants as Factor 1 or Factor 2 are a little different from those using numeric constants:

Rules for Numeric Constants	Rules for Character Constants
• A numeric constant can be any combination of digits 0 through 9. Decimal points and signs can also be included.	• A character constant can be any combination of characters (letters, digits, and special characters). Blanks are also valid.
• The maximum length of a numeric constant is 10 characters, including sign and decimal point.	• The maximum length of a character constant is 8 characters.
• Numeric constants must not be enclosed in apostrophes (').	• Character constants must be enclosed in apostrophes (').

When you use COMP operation code, always remember to compare two numeric fields or constants or two character fields or constants. You cannot compare a numeric field or constant with a character field or constant.

Using an Arithmetic Operation

You can test the results of an arithmetic operation (ADD, SUB, MULT, DIV) for plus, minus, or zero by entering resulting indicators in the appropriate positions on the Calculation Specifications form:



(1) A resulting indicator entered in positions 54 and 55 tells the system to determine if the result field is positive (plus).

(2) A resulting indicator entered in positions 56 and 57 tells the system to determine if the result field is negative (minus).

(3) A resulting indicator entered in positions 58 and 59 tells the system to determine if the result field is zero.

The tests you indicate are performed each time the operation is executed. However, the assigned indicator is set on only if the field satisfies the condition tested. If you enter indicator 99 in positions 54 and 55 to test the result field for plus, indicator 99 is set on only if the result field is plus. If the condition is not met, the indicator is set off.

Again, as with the COMP operation, you can test for one, two, or all three conditions at the same time. When testing for more than one condition, you can use the same or different indicators in these positions. If you intend to do different operations for each of the three conditions, enter a different resulting indicator to test for each condition:

RPG CALCULATION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9093 U/M/050* Printed in U.S.A.

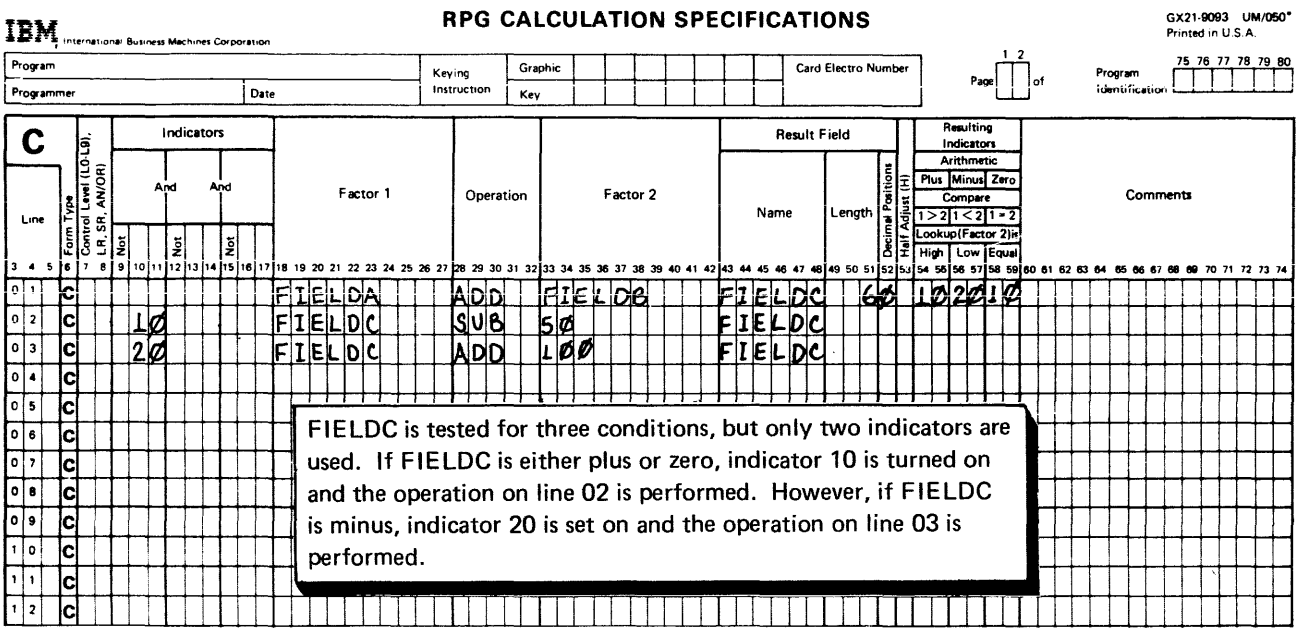
Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

C

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			FIELDA	ADD	FIELDB	FIELDC	60			L02549	
02	C		L0	FIELDC	SUB	50	FIELDC					
03	C		25	FIELDC	SUB	L00	FIELDC					
04	C		49	FIELDC	ADD	10	FIELDC					
05	C											
06	C											
07	C											
08	C											
09	C											
10	C											
11	C											
12	C											
13	C											
14	C											
15	C											
16	C											
17	C											

FIELDC is tested for all three conditions. If the field is positive, indicator 10 is set on and the operation on line 02 is performed. If the field is negative, indicator 25 is set on and the operation on line 03 is done. If the field is zero, indicator 49 is set on and the operation on line 04 is done.

If you want to do the same operations when the result field meets either one of two conditions (plus or zero, minus or zero), you could use the same indicator to test for both:



FIELDC is tested for three conditions, but only two indicators are used. If FIELDC is either plus or zero, indicator 10 is turned on and the operation on line 02 is performed. However, if FIELDC is minus, indicator 20 is set on and the operation on line 03 is performed.

EXAMPLE 6 (STOKST): USING RESULTING INDICATORS TO TEST CONTENTS OF RESULT FIELDS

Program Definition

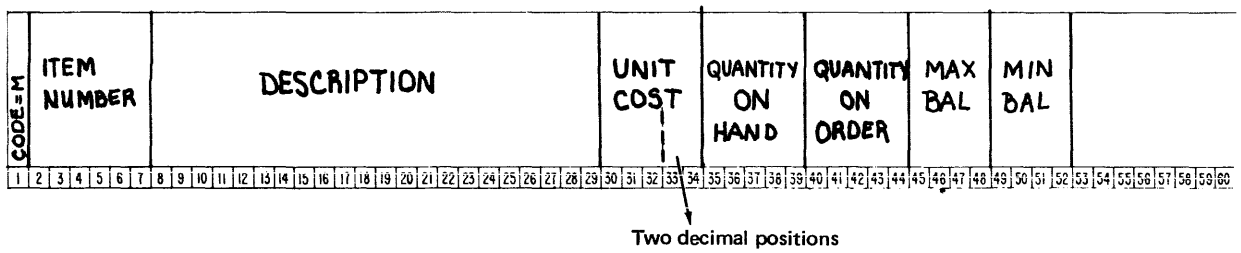
Print a stock status report similar to the one in Example 5. The only difference is the addition of maximum and minimum balances. Item master records usually include the maximum and minimum on-hand quantity for all items. These figures are kept so that checks can be made, whenever the inventory is updated, to determine if quantity on hand is within the limits set.

The first line for each item in the report shows standard descriptive data for the item: item number, item description, quantity on hand, quantity on order, maximum and minimum balances. Subsequent lines give the detail on current transactions involving the item. Quantities remaining on hand and on order are calculated for each item and are printed after all transactions for the item are listed. Whenever shipments reduce stock on hand below the predetermined minimum balance or whenever receipts push the quantity on hand above the predetermined maximum, an exception condition is noted on the report.

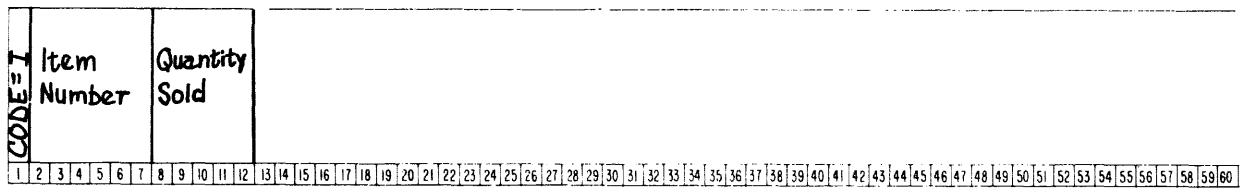
Program Requirements

Input: An inventory file consisting of three different record types. Formats of the three record types are:

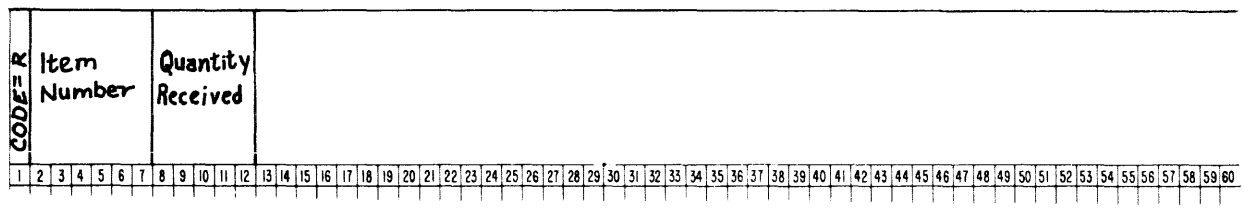
Name ITEM MASTER RECORD



Name ISSUE RECORD



Name RECEIPT RECORD



The file is organized in ascending order by item number. For each item, one master record is required. Issue and receipt records are optional. When present, however, there may be any number of each. Records for each item are in this order:

1. Item master
2. Issue(s)
3. Receipt(s)

Processing:

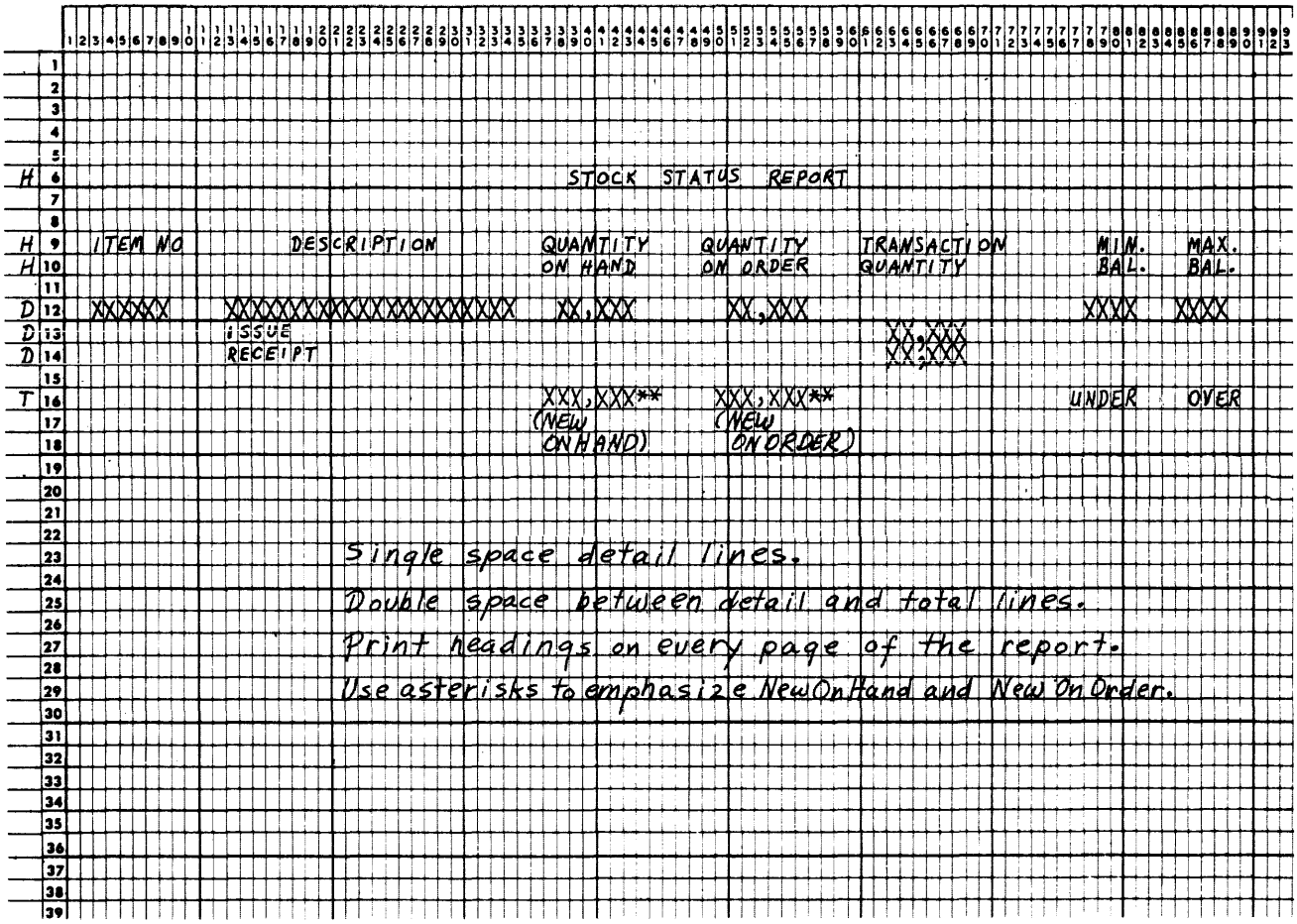
- Find total number of each item sold. To do this, perform the calculation ISSUE + TOTAL ISSUE = TOTAL ISSUE for each issue record.
- Find total number of each item received. To do this, perform the calculation RECEIPT + TOTAL RECEIPT = TOTAL RECEIPT for each receipt record.
- When all transaction records for one item have been read, find new quantity on hand (ON HAND + TOTAL RECEIPT - TOTAL ISSUE = NEW ON HAND) and new quantity on order (ON ORDER - TOTAL RECEIPT = NEW ON ORDER).
- Compare the new quantity on hand to maximum and minimum balances to determine if an exception condition should be noted on the report.

Output: A printed stock report:

STOCK STATUS REPORT

ITEM NO	DESCRIPTION	QUANTITY ON HAND	QUANTITY ON ORDER	TRANSACTION QUANTITY	MIN. BAL.	MAX. BAL.
411116	B500 TWIN SOCKET BLUE	458	500		800	1600
	ISSUE			50		
	RECEIPT			500		
		908**	0**			
411122	B506 SOCKET ADAPT BRN	325	0		300	800
	ISSUE			20		
	ISSUE			38		
	ISSUE			10		
		257**	0**		UNDER	
411173	C151C SIL SWITCH IVORY	50	150		100	200
	RECEIPT			150		
		200**	0**			
411254	A210 PULL CORD GOLD	62	75		80	165
	ISSUE			16		
	ISSUE			30		
		16**	75**		UNDER	

This printer spacing chart shows the format of the report:



RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9092 UM/050*
Printed in U.S.A.

Program: Stock Status
Programmer: John Doe
Date: 1/10/--
Keying Instruction
Graphic
Key
Card Electro Number
Page 01 of __
Program Identification: STOKST

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

H

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	Reserved	Currency Symbol	Date Format	Date Edit	Inverted Print	Reserved	Number of Print Positions	Alternate Collating Sequence	Reserved	Inquiry	Reserved	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Reserved	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG to RPG II Conversion	Number of Formats	S/3 Conversion	Subprogram	CICS/DL/I	Transparent Literal
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
01	H																																

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

F

Line	Form Type	Filename	I/O/U	File Type C/R/T/D/F	E	File Designation A/D	End of File F/V/S/M/D/E	Sequence	File Format	Block Length	Record Length	External Record Name	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/X/D/T/R/ 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	Tape Rewind R/U/N	File Condition U1-U8, UC
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	INPUT	I	P		F				64																							
03	F	OUTPUT	O			F				96								OV			PRINTER											
04	F																																
05	F																																

The entry for the input device depends on your system.

RPG INPUT SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9094 UM/050*
Printed in U.S.A.

Program: Stock Status
Programmer: John Doe
Date: 1/10/--
Keying Instruction
Graphic
Key
Card Electro Number
Page 02 of
Program Identification: STOKST

I

Line	Form Type	Filename or Record Name	Data Structure Name	Sequence O	Sequence R	Sequence A	Sequence N	Sequence D	Number (1/N)	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name	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	Data Structure Occurs n Times	Data Structure Length	Decimal Positions	RPG 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	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	I	INPUT		01					L		L0		L		CM																						
02	I																										2	7		ITEMNOLL						
03	I																										8	29		DESC						
04	I																										35	39	0	ONHAND						
05	I																										40	44	0	ONORD						
06	I																										45	48	0	MAX						
07	I																										49	52	0	MIN						
08	I			02	N	02	0						L		CI																						
09	I																										2	7		ITEMNOLL						
10	I																										8	12	0	ISSUES						
11	I			03	N	03	0						L		CR																						
12	I																										2	7		ITEMNOLL						
13	I																										8	12	0	RECEET						
14	I																																				
15	I																																				

No new functions are described on the Input Specifications form.

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050*
Printed in U.S.A.

Program: Stock Status
Programmer: John Doe
Date: 1/10/--
Keying Instruction: Graphic / Key
Card Electro Number
Page 03 of
Program Identification: STOKST

C

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		20		ISSUES	ADD	TOTISU	TOTISU	50												FIND TOTAL SOLD
02	C		30		RECEET	ADD	TOTREC	TOTREC	50												FIND TOTAL REC.
03	C	LL			ONHAND	SUB	TOTISU	NEWONH	60												FIND NEW TOTAL
04	C	LL			NEWONH	ADD	TOTREC	NEWONH													ONHAND
05	C	LL			ONORD	SUB	TOTREC	NEWONO	60												FIND NEW ONORDR
06	C	L1			NEWONH	COMP	MAX										99				IS ONHAND > MAX
07	C	LL			NEWONH	COMP	MIN											88			IS ONHAND < MIN
08	C	LL			TOTISU	SUB	TOTISU	TOTISU													CLEAR TOTISU
09	C	LL			TOTREC	SUB	TOTREC	TOTREC													CLEAR TOTREC
10	C																				
11	C																				
12	C																				
13	C																				
14	C																				
15	C																				
16	C																				
17	C																				
18	C																				
19	C																				
20	C																				
	C																				
	C																				
	C																				
	C																				
	C																				

Calculations on lines 01 through 05, 08, and 09 are needed to update the quantity on hand and the quantity on order. See Example 5 for an explanation of these entries. After the new quantity on hand (NEWONH) has been calculated, it is compared with MAX to see if it exceeds the maximum limits set (line 06). Indicator 99 in positions 54 and 55 specifies a test to determine whether Factor 1 (NEWONH) is greater than Factor 2 (MAX). If NEWONH is greater, indicator 99 is set on. On line 07, NEWONH is compared with MIN to see if the quantity on hand is less than the minimim set. If it is, indicator 88 is set on.

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050* Printed in U.S.A.

Program: Stock Status | Programmer: John Doe | Date: 1/10/-- | Keying Instruction | Graphic | Key | Card Electro Number | Page 04 of | Program Identification: STOKST

O

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators And And Not	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	OUTPUT	H		3	06		1P						
02	O		OR					OV						
03	O											59		'STOCK STATUS REPORT'
04	O		H		1			1P						
05	O		OR					OV						
06	O											9		'ITEM NO'
07	O											28		'DESCRIPTION'
08	O											56		'QUANTITY QUANTITY'
09	O											71		'TRANSACTION'
10	O											89		'MIN. MAX.'
11	O		H		2			1P						
12	O		OR					OV						
13	O											56		'ON HAND ON ORDER'
14	O											68		'QUANTITY'
15	O											89		'BAL. BAL.'
16	O		D		1			10						
17	O								ITEMNO			8		
18	O								DESC			34		
19	O								ONHAND	1		43		
20	O								ONORD	1		56		
21	O								MIN		Z	81		
22	O								MAX		Z	88		
01	O		D		1			20						
02	O											17		'ISSUE'
03	O								ISSUES	2		68		
04	O		D		1			30						
05	O											19		'RECEIPT'
06	O								RECEET	2		68		
07	O		T	1	2			L1						
08	O								NEWONH	1		43		
09	O											45		'**'
10	O								NEWONO	1		56		
11	O											58		'**'
12	O							88				81		'UNDER'
13	O							99				89		'OVER'
14	O													
15	O													
16	O													
17	O													
18	O													
19	O													
20	O													
	O													

Commas: Yes/Yes, Yes/No, No/Yes, No/No | Zero Balances to Print: 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; 5-9 = User Defined

GX21-9090 UM/050* Printed in U.S.A.

Page 05 of __ | Program Identification: STOKST

Resulting indicators, which indicate whether the new quantity on hand exceeds the maximum or is less than the minimum, are used on the Output Specifications form to tell when exceptional conditions should be noted. Notice that, on the total line, the words UNDER and OVER print only when conditions set by the conditioning indicators are met. UNDER prints when indicator 88 is on. Remember that 88 is set on when NEWONH is less than the minimum. OVER prints only when 99 is on; that is, when NEWONH is greater than the maximum.

FIELD INDICATORS

Field indicators, like resulting indicators, are used to test the contents of a field and to condition operations based on the results of the test.

Program Cycle Operations

Figure 20 shows the operations in the program cycle associated with field indicators. Note that input fields are tested and field indicators are set to reflect the result of the test at the time data is moved into the processing area. Field indicators are not set off at the end of the program cycle. If a field indicator is set on when data is moved into the processing area in the first cycle, it is not reset until the appropriate field is moved into the processing area in a subsequent cycle.

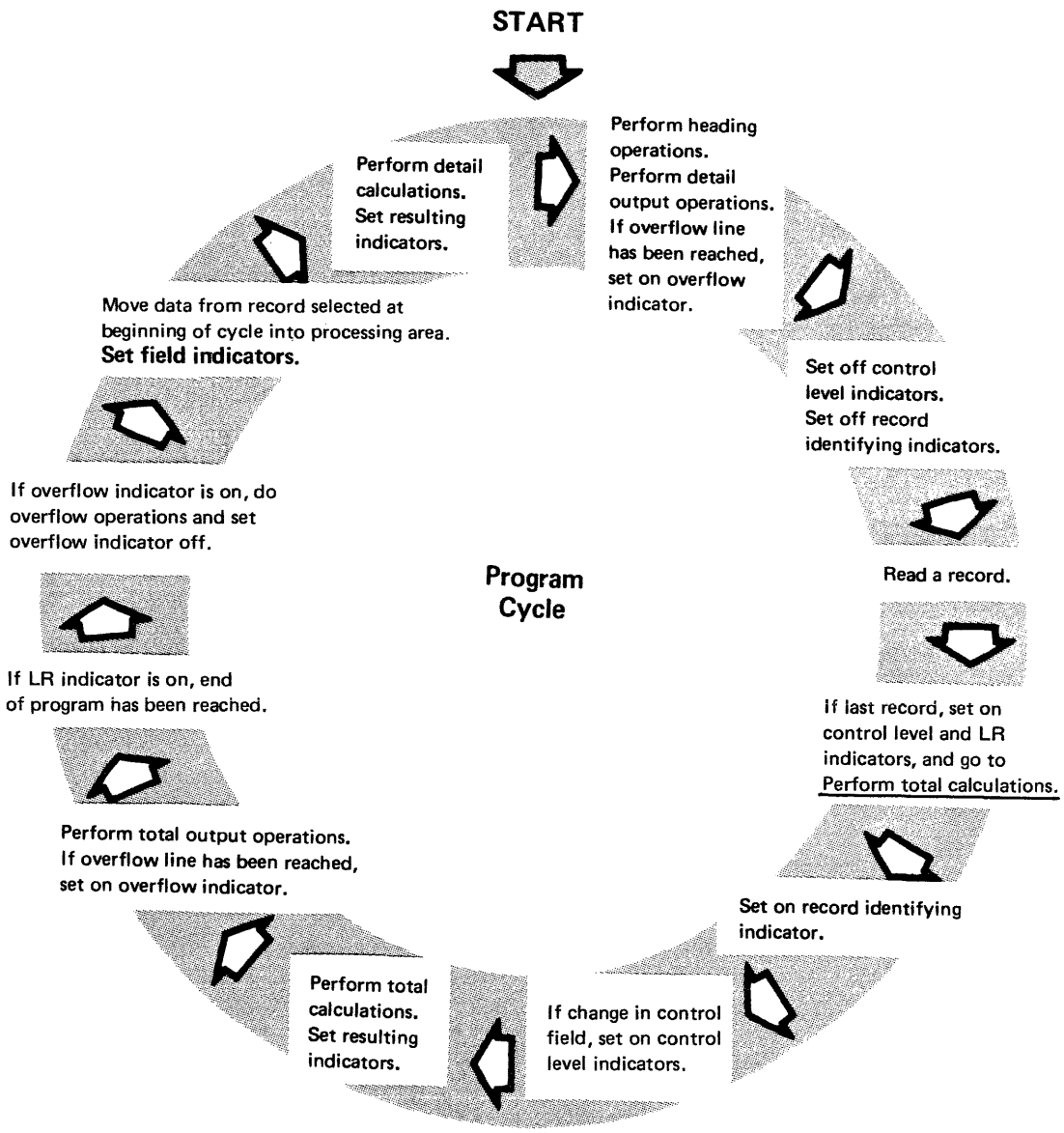
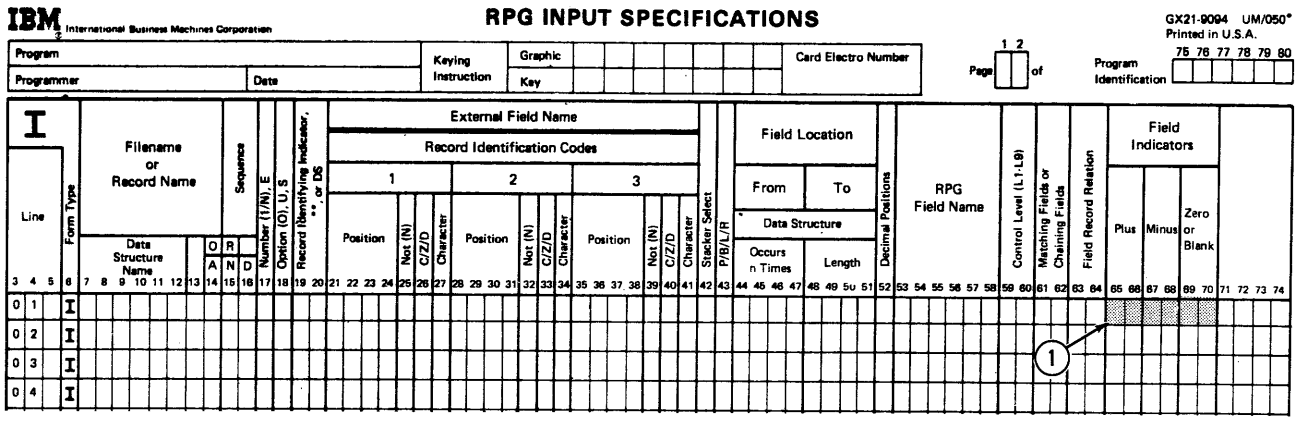
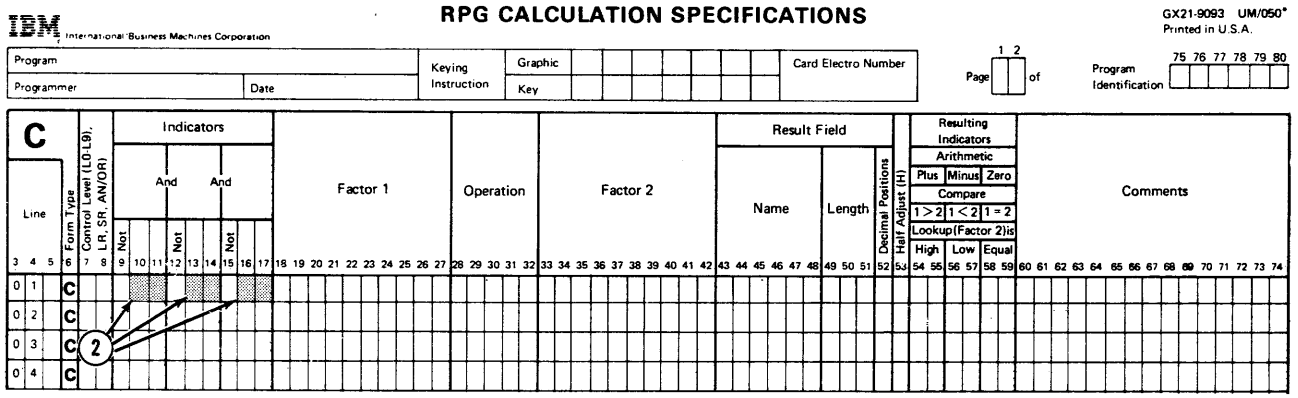


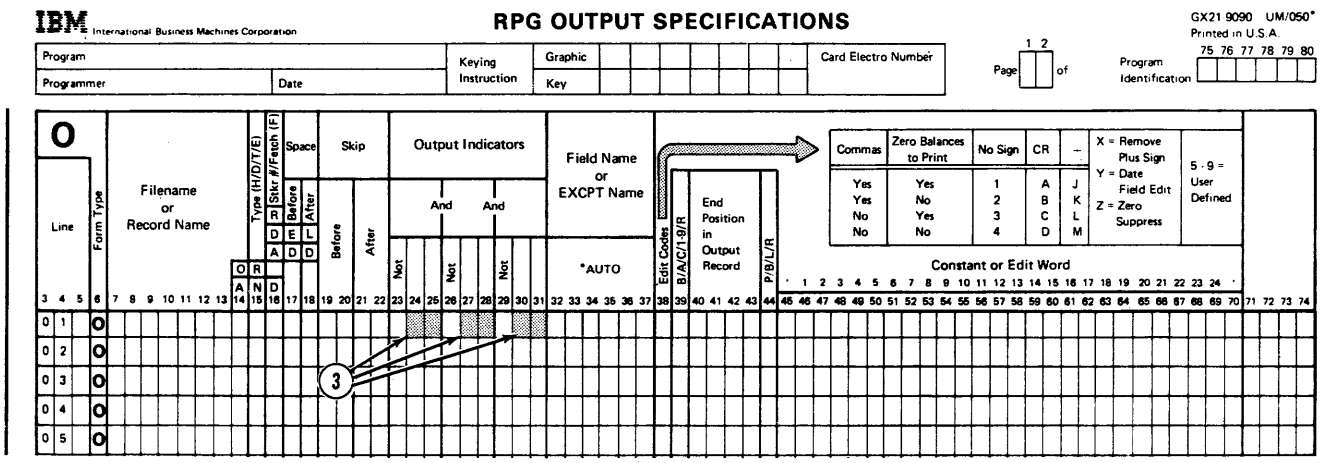
Figure 20. Program Cycle Operations for Field Indicators

RPG Specifications

Make these RPG specifications when you use field indicators:

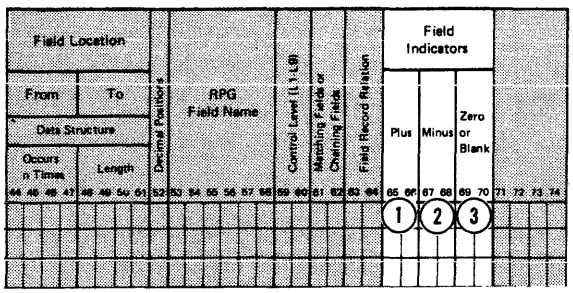






Field indicators are assigned at (1). They may be used to condition calculation operations at (2) and output operations at (3).

You can enter any one of the indicators 01 through 99 in positions 65 through 70 of the Input Specifications form to test an input field. You may assign indicators to test for three possible conditions:



① A field indicator assigned in positions 65 and 66 tells the system to determine if a numeric input field is positive (plus).

② A field indicator assigned in positions 67 and 68 tells the system to determine if a numeric input field is negative (minus).

③ A field indicator assigned in positions 69 and 70 tells the system to determine if a character input field is blank or if a numeric field has a value of zero.

EXAMPLE 7 (AGETB): USING FIELD INDICATORS TO TEST CONTENTS OF INPUT FIELDS

Program Definition

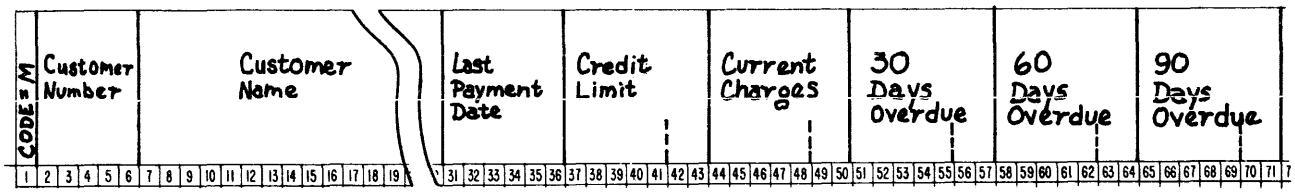
Create an aged trial balance report that lists:

- Name and customer number of all charge customers who have payments due.
- Amounts due.
- Overdue balances.

The customer master file, which contains records for all customers regardless of their balance, is used as the input file. The report is to show only those customers with payments due. Thus, information from customer records that contain a zero or credit balance should not be printed.

Program Requirements

Input: A customer master file consisting of one record type:



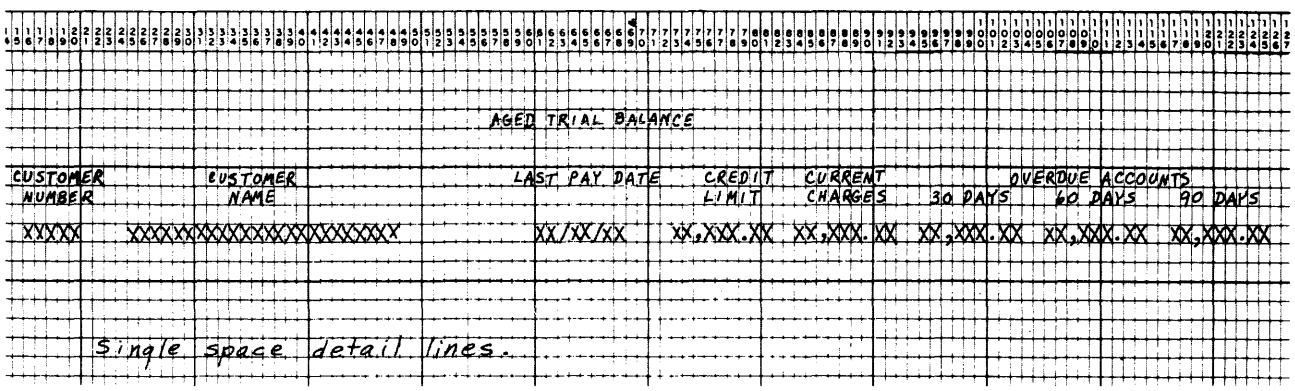
Processing: Check input balance due field for zero or credit balance (use field indicators).

Output: A printed aged trial balance report:

AGED TRIAL BALANCE

CUSTOMER NUMBER	CUSTOMER NAME	LAST PAY DATE	CREDIT LIMIT	CURRENT CHARGES	OVERDUE ACCOUNTS 30 DAYS	60 DAYS	90 DAYS
10867	ALLEN & CO.	2/16/--	15,000.00	6,919.77	375.58		
16535	ANDERSON AUTO SUPPLY	1/28/--	2,500.00	1,665.49			
17849	ANDREWS AND SONS INC	2/05/--	750.00			146.64	
18978	ARGONAUT ENGINEERING	12/27/--	2,000.00	2,111.30,	611.54	312.13	90.44
24743	BERKLEY PAPER CO	2/21/--	6,300.00	1,185.50	2,652.45	1,400.05	51.00
25271	BEST DISTRIBUTION CO	10/06/--	1,000.00	3.25			762.19

This printer spacing chart shows how the report is formatted:



RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9092 UM/050*
Printed in U.S.A.

Program: Aged Trial Balance
Programmer: John Doe
Date: 1/10/--
Keying Instruction: Graphic / Key
Card Electro Number
Page 01 of __
Program Identification: AGETB

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	Reserved	Currency Symbol	Date Format	Date Edit	Inverted Print	Reserved	Number of Print Positions	Alternate Collating Sequence	Reserved	Inquiry	Reserved	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Reserved	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG to RPG II Conversion	Number of Formats	S/3 Conversion	Subprogram	CICS/DL/I	Transparent Literal
01	H																																

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D/F	E	A/D	F/V/S/M/D/E	Block Length	Record Length	L/R	A/P/I/K	I/X/D/T/R/ 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	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8, UC	Continuation Lines K	Option	Entry	A/U	R/U/N
02	F	MASTER	I	P			F		128																						
03	F	REPORT	O				F		132					OV			PRINTER														
04	F																														

The entry for the input device depends on your system.

RPG INPUT SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9094 UM/050*
Printed in U.S.A.

Program: Aged Trial Balance
Programmer: John Doe
Date: 1/10/--
Keying Instruction: Graphic / Key
Card Electro Number
Page 02 of
Program Identification: AGETB

Line	Form Type	Filename or Record Name / Data Structure Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, ** or DS	External Field Name / 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	Data Structure Occurs n Times	Data Structure Length	Decimal Positions	RPG 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	MASTER	NS			01	1		CM																							
02	I																				2	6				CUSTNO						
																					7	30				NAME						
																					31	360				LASTPA						
																					37	432				LIMIT						
																					44	502				CHARGE				99		
																					51	572				OVER30				98		
																					58	642				OVER60				97		
																					65	712				OVER90				96		
15	I																															

To prevent listing customers who have a zero or credit balance, we must know if the current charge field (CHARGE) or the overdue fields contain a plus amount. Indicators 96 through 99, specified in positions 65 and 66, test the fields for these conditions. These indicators are set on when the fields are plus.

Note that the record identification code has been described (positions 21 through 27) and a record identifying indicator assigned (positions 19 and 20). This should always be done even though there is only one record type per file.

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21 9090 UM 050*
Printed in U.S.A.

Program: Aged Trial Balance | Keying Instruction | Graphic | Key | Card Electro Number
Programmer: John Doe | Date: 1/10/--

Page 03 of | Program Identification: AGETB

O

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Stkr #/Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And/Not	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	REPORT	H		3	06			LP					
02	O		OR						OV					
03	O											74		'AGED TRIAL BALANCE'
04	O		H		1				LP					
05	O		OR						OV					
06	O											22		'CUSTOMER'
07	O											39		'CUSTOMER'
08	O											71		'LAST PAY DATE'
09	O											91		'CREDIT CURRENT'
10	O											118		'OVERDUE ACCOUNTS'
11	O		H		2				LP					
12	O		OR						OV					
13	O											21		'NUMBER'
14	O											37		'NAME'
15	O											91		'LIMIT CHARGES'
16	O											102		'30 DAYS'
17	O											124		'60 DAYS 90 DAYS'

Commas: Yes/Yes, Yes/No, No/Yes, No/No | Zero Balances to Print | 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 | 5-9 = User Defined

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21 9090 UM 050*
Printed in U.S.A.

Program: Aged Trial Balance | Keying Instruction | Graphic | Key | Card Electro Number
Programmer: John Doe | Date: 1/10/--

Page 04 of | Program Identification: AGETB

O

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Stkr #/Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And/Not	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O		D			1			99 01					
02	O		OR						98 01					
03	O		OR						97 01					
04	O		OR						96 01					
05	O									CUSTNO		20		
06	O									NAME		48		
07	O									LASTPAY		68		
08	O									LIMIT 2		81		
09	O									CHARGE 2		92		
10	O									OVER30 2		103		
11	O									OVER60 2		114		
12	O									OVER90 2		125		
13	O													
14	O													
15	O													
16	O													

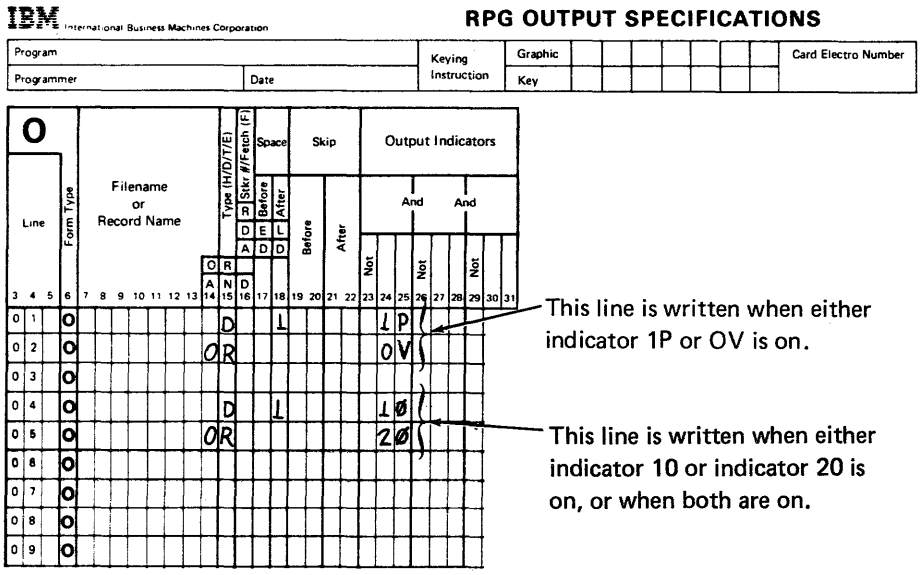
Commas: Yes/Yes, Yes/No, No/Yes, No/No | Zero Balances to Print | No Sign: 1, 2, 3, 4 | CR: A, B, C, D | –: J, K, L, M
X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress | 5-9 = User Defined

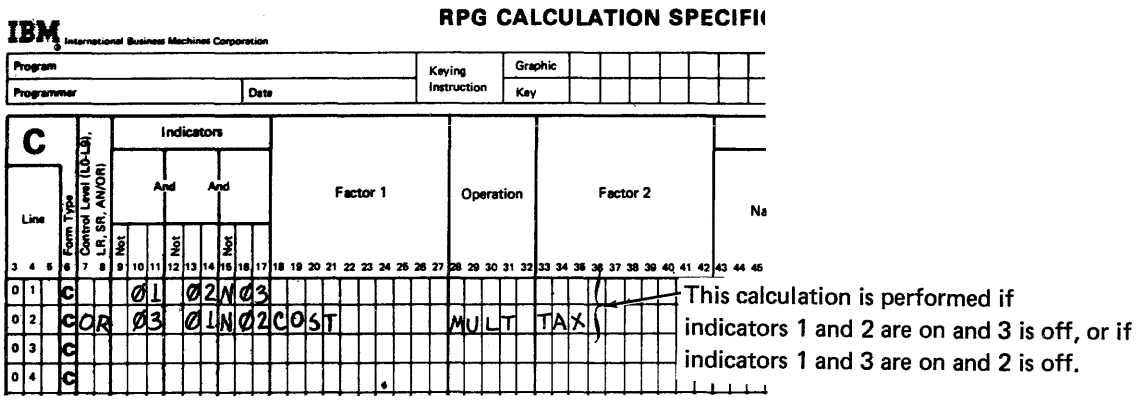
Field indicators 96 through 99, used to test the CHARGE and overdue fields for a plus condition, are used to condition detail records. The record is to be printed when one of the fields is plus; that is, when indicator 99, 98, 97, or 96 is on. Note that record identifying indicator 01 is also used to condition the detail record. This is done to prevent printing information from a record type other than 01 (one that may have accidentally gotten into the file).

Conditioning Operations by More Than One Indicator

In this chapter, you have learned about many different kinds of conditioning indicators: control level, first page, overflow, last record, record identifying, resulting, and field indicators. In many programs you use two or more conditioning indicators. Indicators used together can be in either an OR or AND relationship.

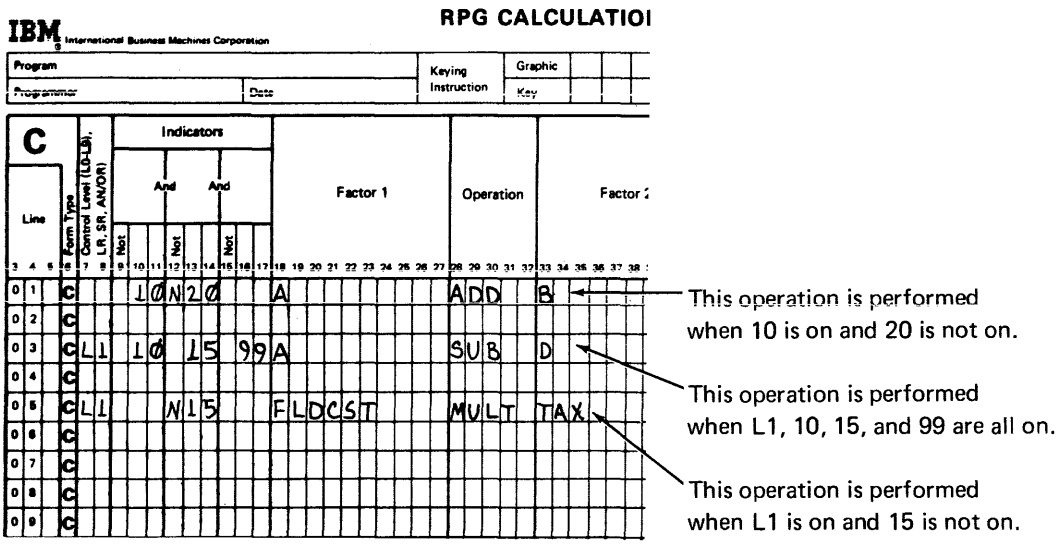
You have already read about indicators in an OR relationship. You learned that, if an operation can be done when either one of two conditions or both conditions exist, you can specify the conditioning indicators like this:

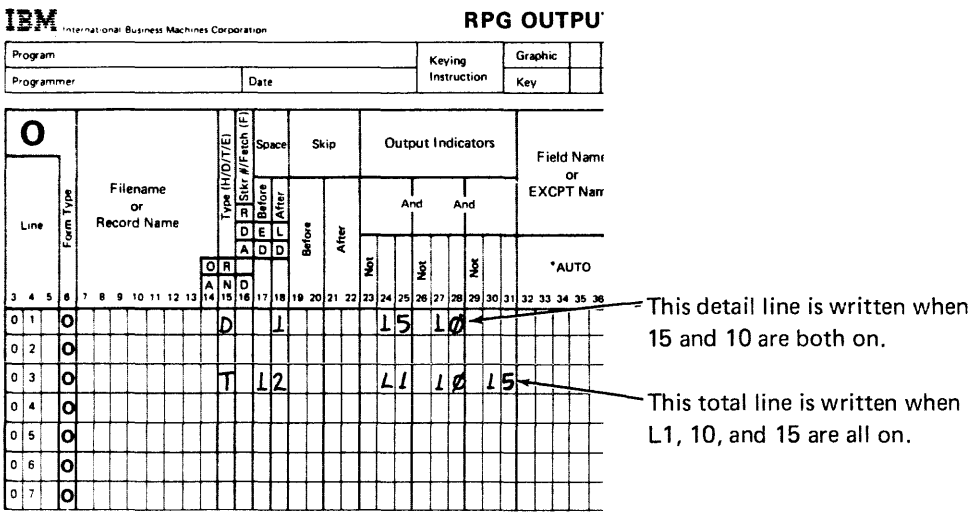




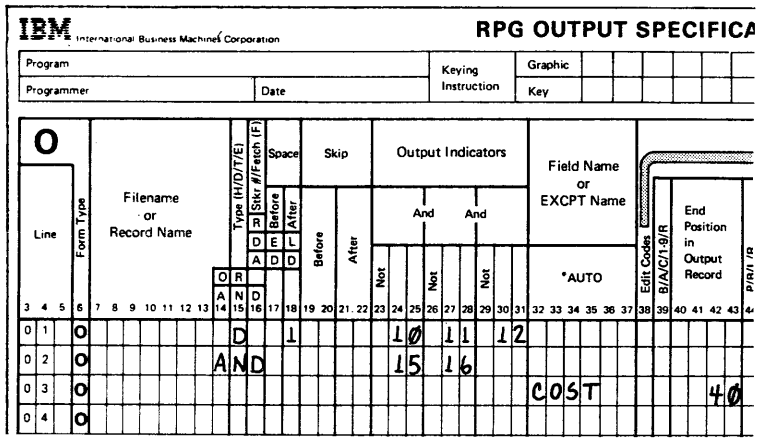
Conditioning indicators can be used in the OR relationship on both the Calculation Specifications form and the Output Specifications form.

If you specify two or more conditioning indicators on one line, they are in an AND relationship. The AND relationship means that all conditions must be satisfied before the operation is performed:

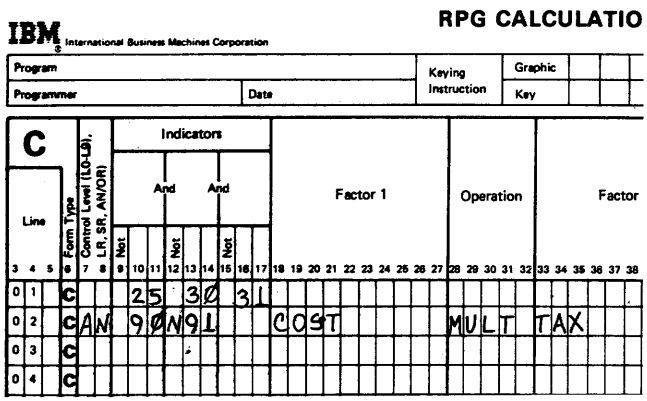




If your calculation or output operation must be conditioned by more than three indicators, additional indicators can be specified on the next line if AND is entered in positions 14, 15, and 16 of the Output Specifications form or AN is entered in positions 7 and 8 of the Calculation Specifications form:



Five indicators used in an AND relationship condition this detail record. Three indicators are specified on one line; the remaining are specified on the following line with the word AND in positions 14 through 16. Indicators 10, 11, 12, 15, and 16 must all be on before the detail line will be printed.



This calculation operation is done only if indicators 25, 30, 31, and 90 are on and 91 is not on. Note that the operation is specified on the AN line.

Your responsibilities as a programmer include:

- Determining the program requirements
- Determining what RPG specifications and program cycle operations are needed for the program
- Writing the specifications
- Documenting the program
- Preparing your source program for compilation
- Compiling the source program
- Testing the program

DETERMINE THE PROGRAM REQUIREMENTS

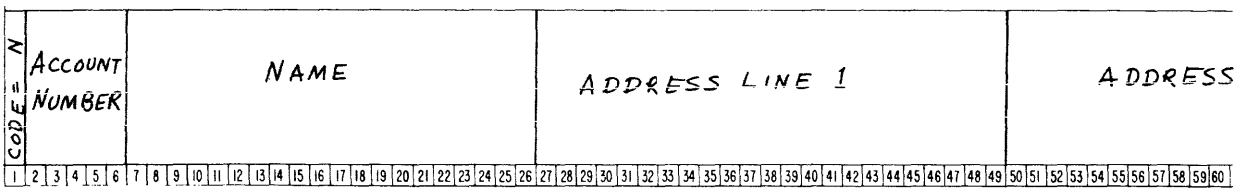
The requirements for a program are generally described in terms of the input provided and the output required. The following paragraphs and illustrations describe the program requirements.

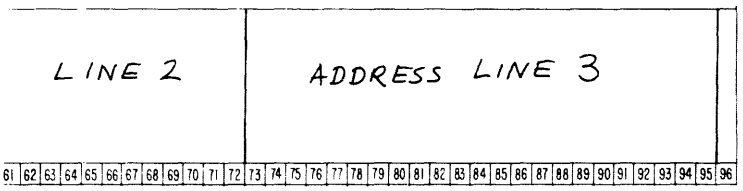
An invoice is to be prepared like that shown below:

```
                                    INVOICE

ACCOUNT NUMBER 09621

NAME              SMITH MANUFACTURING

ADDRESS           13620 9TH ST NE
                  BERNALILLO
                  NEW MEXICO 56120


SHIPPING INSTRUCTIONS   BY AIR

ITEM NUMBER          DESCRIPTION          QUANTITY     UNIT PRICE          AMOUNT
  439167             SHEARS                    100          27.56          2,756.00
  629408             GASKET CORK              3000           1.15          3,450.00
  102139             SPRIDGET WHITE             50         750.00         37,500.00

                                                      INVOICE TOTAL       212,157.92
```

The input file contains two types of records: name/address records for all customers who made purchases on credit during the month and transaction records for each item purchased by the customers during the month. The name/address and transaction records look like this:

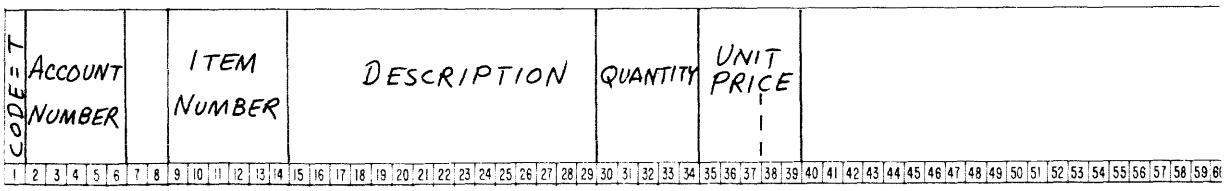
Name NAME/ADDRESS RECORD



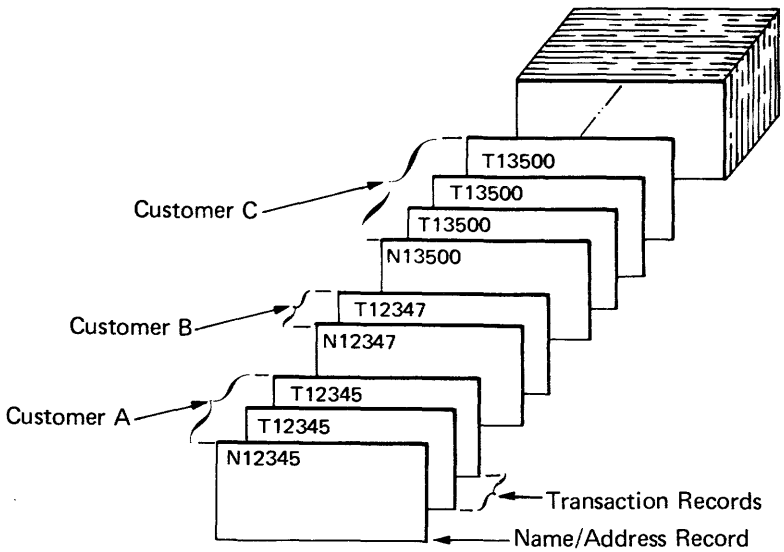


Shipping code (1, 2, or 3)

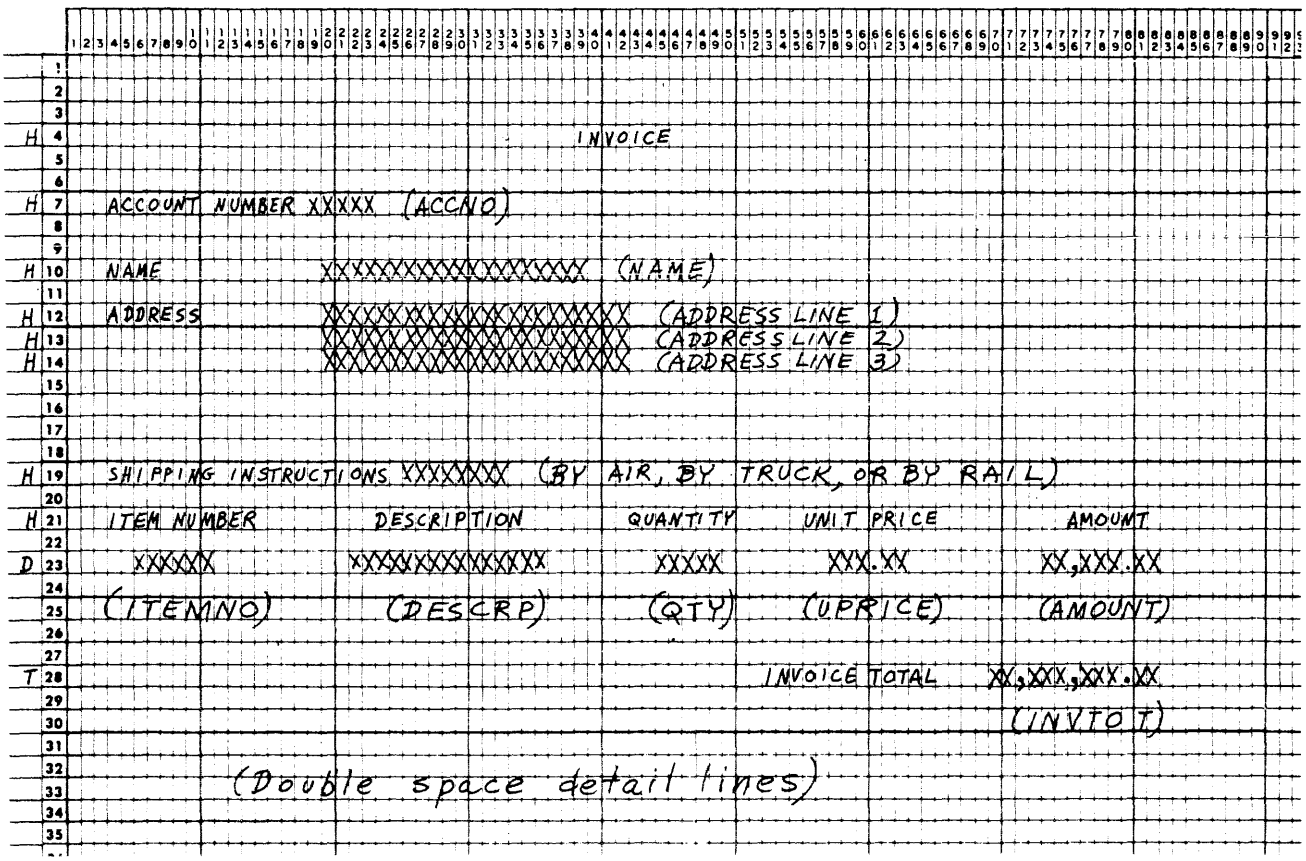
Name TRANSACTION RECORD



The input file is organized so that all transaction records for a customer follow the customer's name/address record. Each customer has one name/address record but might have one or more transaction records.



Standard computer paper is to be used for printing the invoices. Each invoice should be formatted like this:



Your first step is to analyze the problem and decide what processing must be done to get the desired results. Always keep in mind how things are done using RPG. In your analysis of the program, you would probably think of these points:

- Information for the first part of the invoice is taken from the name/address record. Information for the second part (list of transactions) is taken from transaction records.

- Before shipping instructions can be printed, the shipping code recorded in the record must be determined:

  1 = By truck

  2 = By rail

  3 = By air

- AMOUNT and INVTOT (invoice total) must be calculated because this information is not in the input records. These calculations must be done for all transaction records:

  QTY x UPRICE = AMOUNT

  AMOUNT + INVTOT = INVTOT

- INVTOT should be printed only after all transaction records for one account have been processed.

- The invoice for each customer must be on a separate page. This means that forms must advance each time a new customer name/address record is found. It is possible that one customer has purchased so many items that they cannot be listed on one page. In this case, forms should advance when the end of a page is reached. When an invoice includes more than one page, headings should be printed on all pages.

DETERMINE RPG SPECIFICATIONS NEEDED FOR THE PROGRAM

After you have carefully analyzed the requirements, determine what RPG specifications and program cycle operations you need. For example, consider the following:

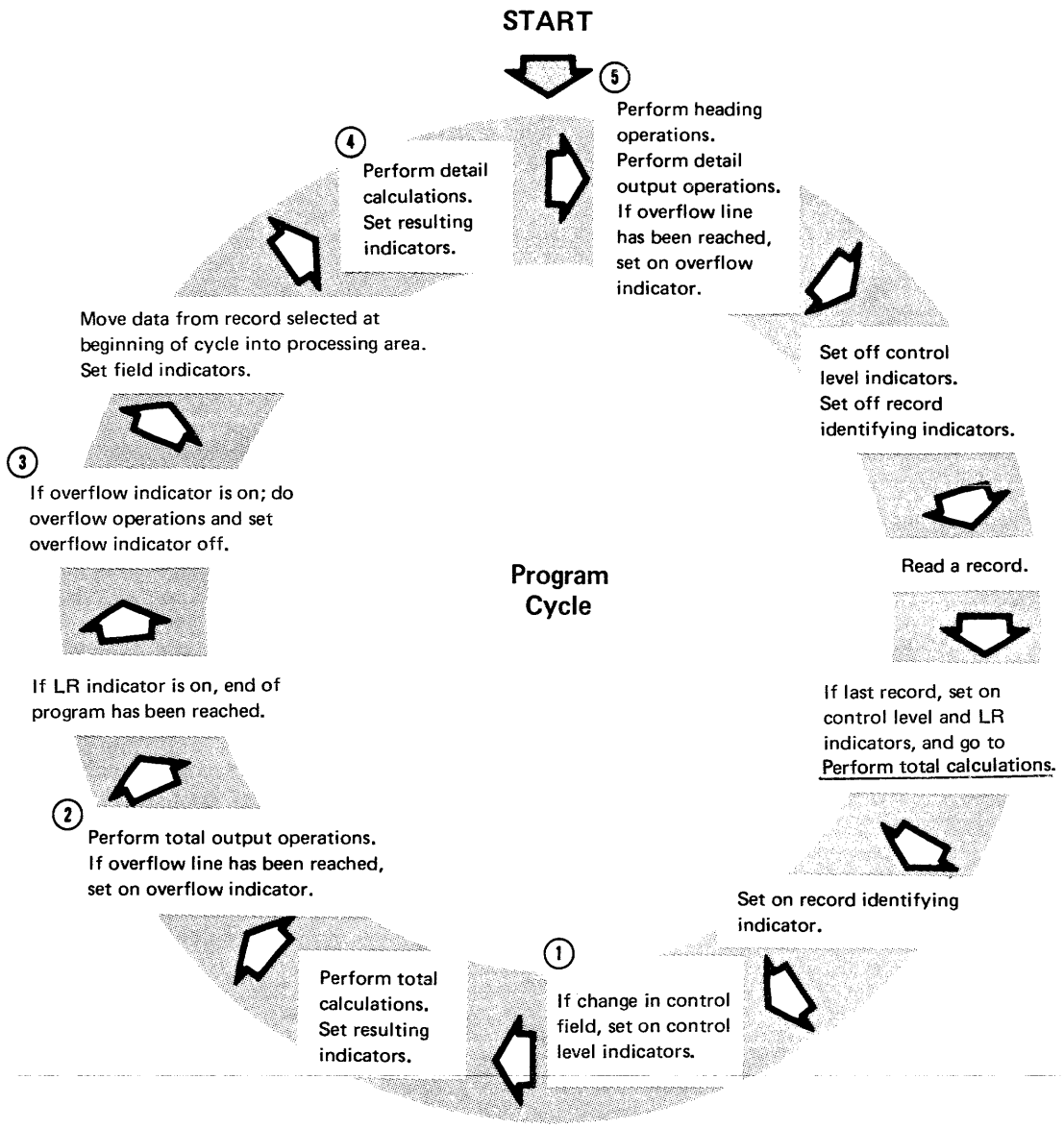
- Several kinds of records are in the file. This means that record identifying indicators must be specified to tell what to do for each record.

- The shipping code must be determined. One way to do this is to compare the shipping code to 2. Through the use of resulting indicators, you can determine if the code is less than 2 (1), greater than 2 (3), or equal to 2.

- INVTOT is printed only after all transaction records for one account have been processed. This is a total operation, done only after a group of records has been processed. Therefore, control fields and control level indicators must be used to do a total operation. The account number field can be used as the control field.

- Forms should advance each time a different name/address record is encountered or whenever overflow occurs. Thus, heading lines must be conditioned by a record identifying indicator and the OV indicator.

If the indicators and steps just listed are used, the RPG program cycle would include the steps shown in Figure 21.



(1) Did account number change?

(2) Print total only after all transaction records for one customer have been processed.

(3) Print all heading lines if overflow occurs.

(4) The operation to find shipping instructions can be done only for name/address records. The operations to find AMOUNT and INVTOT can be done only for transaction records.

(5) Headings for the invoice can be printed only when name/address records are read, and detail lines can be printed only when transaction records are read.

Figure 21. Program Cycle Operations for Sample Job

WRITE THE SPECIFICATIONS

After you have analyzed the problem and determined how to solve it using RPG, you can write the specifications. Figure 22 shows the specifications for the program.

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092 UM 050*
Printed in U.S.A.

IBM International Business Machines Corporation

Program: Invoice Preparation | Keying Instruction: Graphic / Key | Card Electro Number
Programmer: John Doe | Date: 1/10/--
Page: 01 of __ | Program Identification: INVOCE

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	Reserved	Currency Symbol	Date Format	Date Edit	Inverted Print	Reserved	Number of Print Positions	Alternate Collating Sequence	Reserved	Inquiry	Reserved	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Reserved	Table Load -halt	Shared I/O	Field Print	Formatted Dump	RPG to RPG II Conversion	Number of Formats	S/3 Conversion	Subprogram	CICS/DL/I	Transparent Literal
01	H																																

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D/F	E	A/D	F/V/S/M/D/E	Block Length	Record Length	L/R	A/P/I/K	I/X/D/T/R/ 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, UC
								External Record Name																						
02	F	NAMADD	I	P			F		128																					
03	F	INVOICE	O				F		120					OV			PRINTER													
04	F																													
05	F																													
06	F																													
07	F																													
08	F																													
09	F																													
10	F																													
	F																													
	F																													

Figure 22 (Part 1 of 4). Specifications for Invoice Preparation Program

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM/050* Printed in U.S.A.

Program: Invoice Preparation | Keying Instruction | Graphic | Key | Card Electro Number
Programmer: John Doe | Date: 1/10/-- | Page: 02 of | Program Identification: INVOCE

Line	Form Type	Filename or Record Name	Data Structure Name	O/A	R/N	D	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / 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	To	Data Structure: Occurs n Times	Length	Decimal Positions	RPG Field Name	Control Level (L1 L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	NAMADD					01	L			1		C	N																						
02	I																										2	6		0	ACCNO	L1				
03	I																										7	26			NAME					
04	I																										27	49			ADDR1					
05	I																										50	72			ADDR2					
06	I																										73	95			ADDR3					
07	I																										96	96		0	SHPCD					
08	I						02	N		20	1		C	T																						
09	I																										2	6		0	ACCNO	L1				
10	I																										9	14		0	ITEMNO					
11	I																										15	29			DESCRP					
12	I																										30	34		0	QTY					
13	I																										35	39		2	UPRICE					

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050* Printed in U.S.A.

IBM International Business Machines Corporation

Program: Invoice Preparation | Keying Instruction | Graphic | Key | Card Electro Number
Programmer: John Doe | Date: 1/10/-- | Page: 03 of | Program Identification: INVOCE

Line	Form Type	Control Level (L0 L9, LR, SR, AN/OR)	Indicators: And 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		10			SHPCD	COMP	2											97	98	99	IS CODE 1, 2, 3 ?
02	C		20			QTY	MULT	UPRICE	AMOUNT	7	2											FIND ITEM TOTAL
03	C		20			AMOUNT	ADD	INVTOT	INVTOT	10	2											FIND INV TOTAL
04	C																					
05	C																					
06	C																					
07	C																					
08	C																					

Figure 22 (Part 2 of 4). Specifications for Invoice Preparation Program

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM'050*
Printed in U.S.A.

Program: Invoice Preparation
Programmer: John Doe
Date: 1/10/--
Keying Instruction
Graphic Key
Card Electro Number
Page 04 of
Program Identification: INVOCE

O

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space	Skip	Output Indicators	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	>/B/L/R	Constant or Edit Word
				Before	After	Before / After / And / And / Not	*AUTO				
01	O	INVOICE	H	3	04	1Ø					
02	O		OR			OVN1Ø					
03	O								45		'INVOICE'
04	O		H	3		1Ø					
05	O		OR			OVN1Ø					
06	O								17		'ACCOUNT NUMBER'
07	O						ACCNO		23		
08	O		H	2		1Ø					
09	O		OR			OVN1Ø					
10	O								7		'NAME'
11	O						NAME		39		
12	O		H	1		1Ø					
13	O		OR			OVN1Ø					
14	O								1Ø		'ADDRESS'
15	O						ADDR1		42		
16	O		H	1		1Ø					
17	O		OR			OVN1Ø					
18	O						ADDR2		42		
19	O		H	2		1Ø					
20	O		OR			OVN1Ø					
	O										
	O										
	O										
	O										
	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

5 9 User Defined

Figure 22 (Part 3 of 4). Specifications for Invoice Preparation Program

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050* Printed in U.S.A.

Program: Invoice Preparation	Keying Instruction	Graphic		Card Electro Number	
Programmer: John Doe	Date: 1/10/--	Key			

Page 05 of

Program Identification: INVOCE

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; 5 - 9 = User Defined

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Stkr #/Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O										ADDR3		42		
02	O		H		3	2			1Ø						
03	O		OR						OVN1Ø						
04	O												24		'SHIPPING INSTRUCTIONS'
05	O									97			33		'BY AIR'
06	O									98			33		'BY TRUCK'
07	O									99			33		'BY RAIL'
08	O		H		2				1Ø						
09	O		OR						OVN1Ø						
10	O												14		'ITEM NUMBER'
11	O												34		'DESCRIPTION'
12	O												5Ø		'QUANTITY'
13	O												65		'UNIT PRICE'
14	O												81		'AMOUNT'
15	O		D		2				2Ø						
16	O										ITEMNOZ		11		
17	O										DESCRP		36		
18	O										QTY Z		49		
19	O										UPRICE1		63		
20	O										AMOUNT1		82		
	O		T		3	2			L1						
	O												65		'INVOICE TOTAL'
	O										INVTOT1B		82		
	O														
	O														

Figure 22 (Part 4 of 4). Specifications for Invoice Preparation Program

DOCUMENT THE PROGRAM

An important part of every programmer's job is to explain his program. This documentation provides information for people who will run the program and for programmers who may later need to alter or update it. Documentation is also useful to you. It is not always easy to remember what every program you wrote does. Reading documentation is a much easier way to recall the program than figuring out each instruction.

Documentation consists of:

1. Telling generally what the program does.

2. Describing input and output. (Record layout forms and printer spacing charts are an excellent means of describing input and output.) File names and field names should be meaningful.

3. Explaining the coding.

4. Telling the operator how to run the program, what to do if the system stops because of an error, and what to do when the program ends.

Not all documentation can be done at the time you write specifications. However, when writing your specifications, you can also write an explanation for a line or lines of coding on the specifications forms. You have probably noticed positions labeled Comments on the specifications sheets. Here is where you write an explanation for your coding.

In addition to using the comment positions on the coding forms, you can use comment lines. A comment line is indicated by an * (asterisk) in position 7 of the coding form:

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM/050* Printed in U.S.A.

Program | Keying Instruction: Graphic / Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

Programmer | Date

I

Line	Form Type	Filename or Record Name	Date Structure Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, ** or DS	External Field Name / 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	Data Structure: Occurs n Times	Data Structure: Length	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	
01	I																																	
02	I																																	
03	I																																	
04	I																																	
05	I																																	
06	I																																	
07	I																																	
08	I																																	
09	I																																	

```
I*A COMMENT LINE IS INDICATED BY AN * IN COLUMN 7. A COMMENT LINE CAN
I*INCLUDE ANY INFORMATION YOU WANT.
I**********************************************************************
```

Comment lines can be used anywhere on any specifications form. There is no limit to the number you can use. The RPG compiler does not regard comments and comment lines as part of the program. Therefore, the compiler does not translate the comments into instructions; however, the lines are printed as part of the source program listing.

PREPARE FOR COMPILATION

After completing your source program, you must prepare it for compilation.

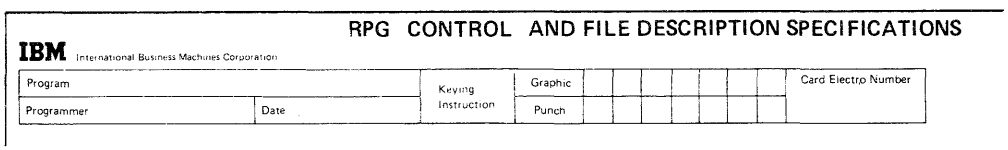
Specifications Form Order

Your specifications forms must be in this order:

1. Control and File Description Specifications form
2. Input Specifications form
3. Calculation Specifications form
4. Output Specifications form

Number the forms in positions 1 and 2. At this time, you might also check to see that the top part of each form is completely filled in.

If you are planning to give these specifications to someone to key, it is a good idea to fill in the box labeled Keying Instructions:

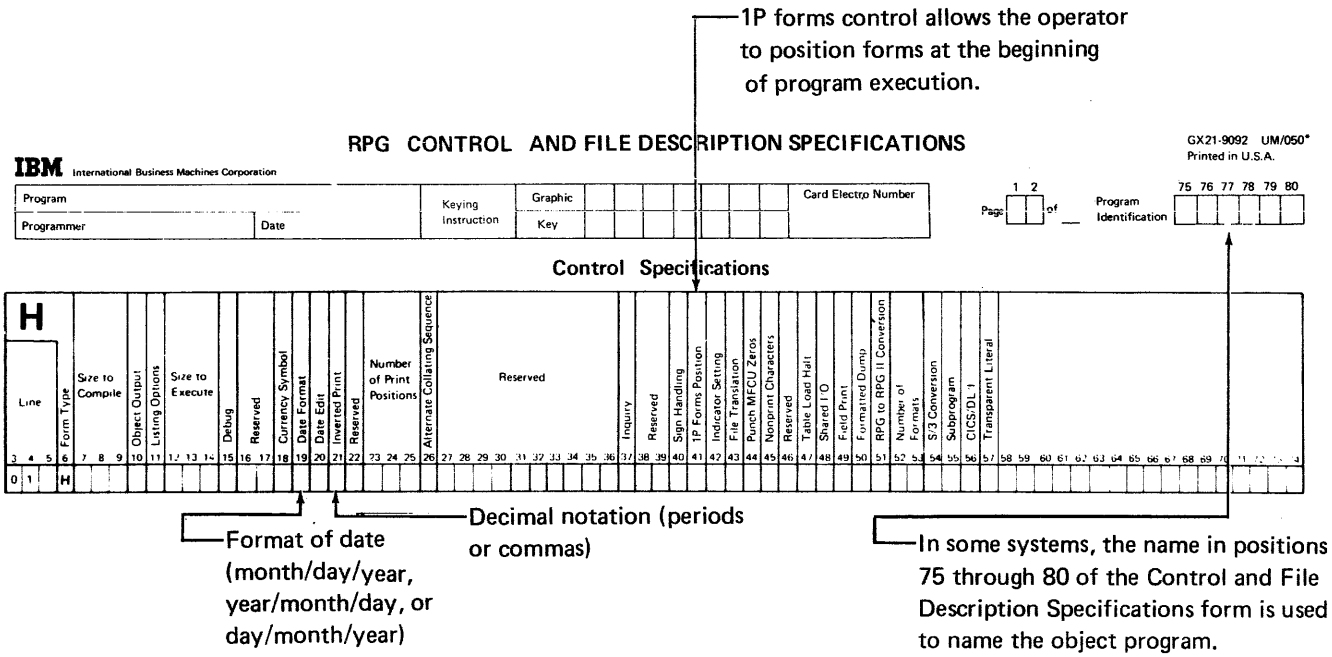


You indicate in this box the graphic symbols you are using and their meaning. Some printed letters and numbers are easily confused. For example, it is sometimes difficult to differentiate between the number 0 and the letter O and between the number 2 and the letter Z. You may, therefore, devise a graphic symbol that you use for certain letters. Some people use Ø for zero, Ƶ for the letter Z. Explain your symbols so that the operator will know what to key when the symbol appears on the coding forms.

Control Specifications Preparation

Some systems require control specifications. If yours does, you have to fill out the control specifications at the top of the Control and File Description Specifications form.

Control specifications give the compiler information about the system and tell whether any special RPG functions are used in the program. The following entries are typical.



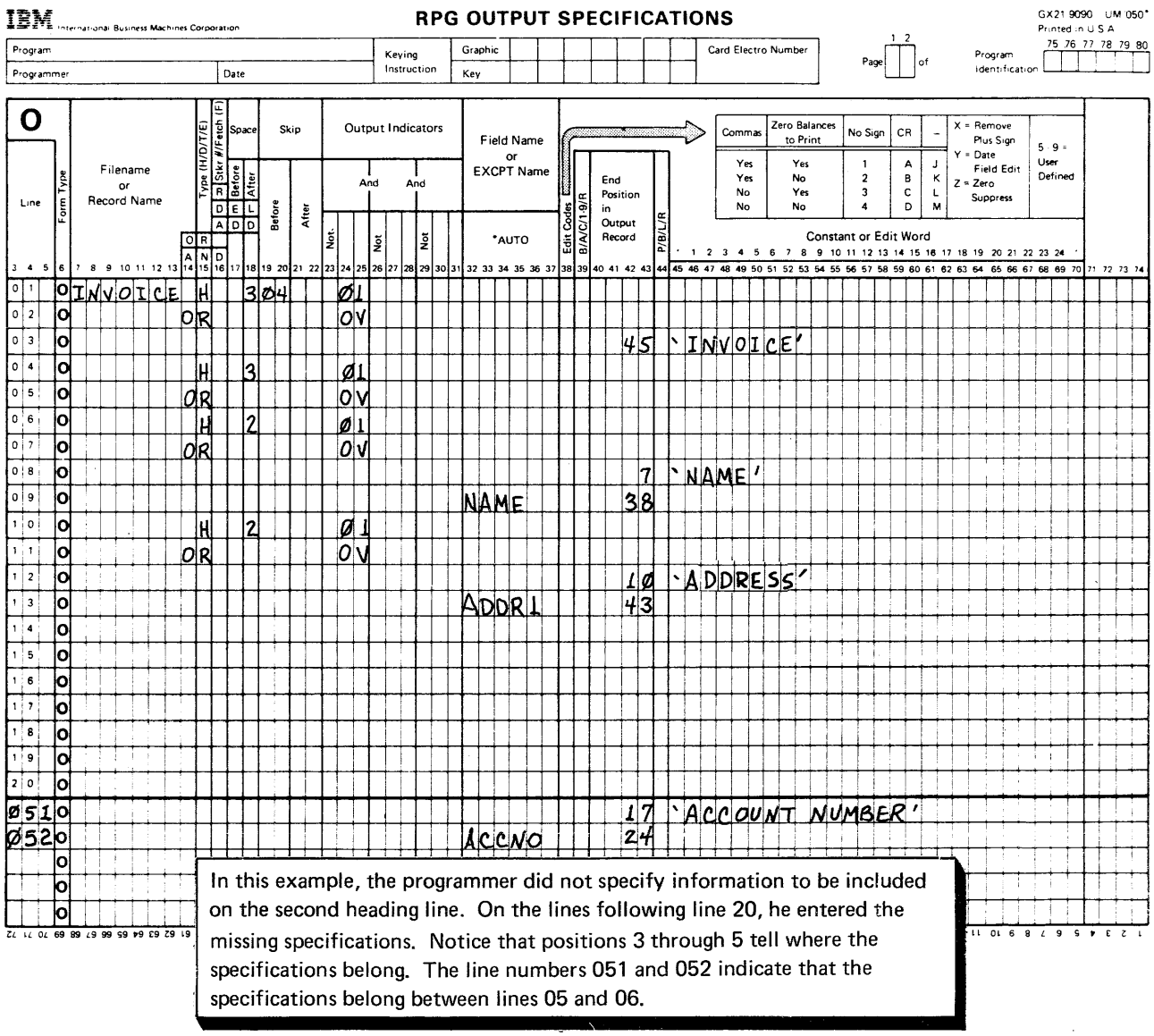
Checking the Specifications

Desk checking is a good way to reduce the number of potential program errors. Desk checking means carefully checking through your specifications to see whether you have:

- Placed entries in appropriate positions
- Used correct entries in positions
- Spelled the same field and file names identically throughout your program
- Used your indicators correctly

If you find that you have omitted a specification (for example, did not name an input field or an output field or did not enter a calculation), you can enter it on a line following line 20 (line 10 on the Control and File Description Specifications form under File Description).

Notice that no line numbers have been entered in positions 3 through 5 of the specification lines located below line 20. You can place numbers in these positions to tell where the missing specification belongs:



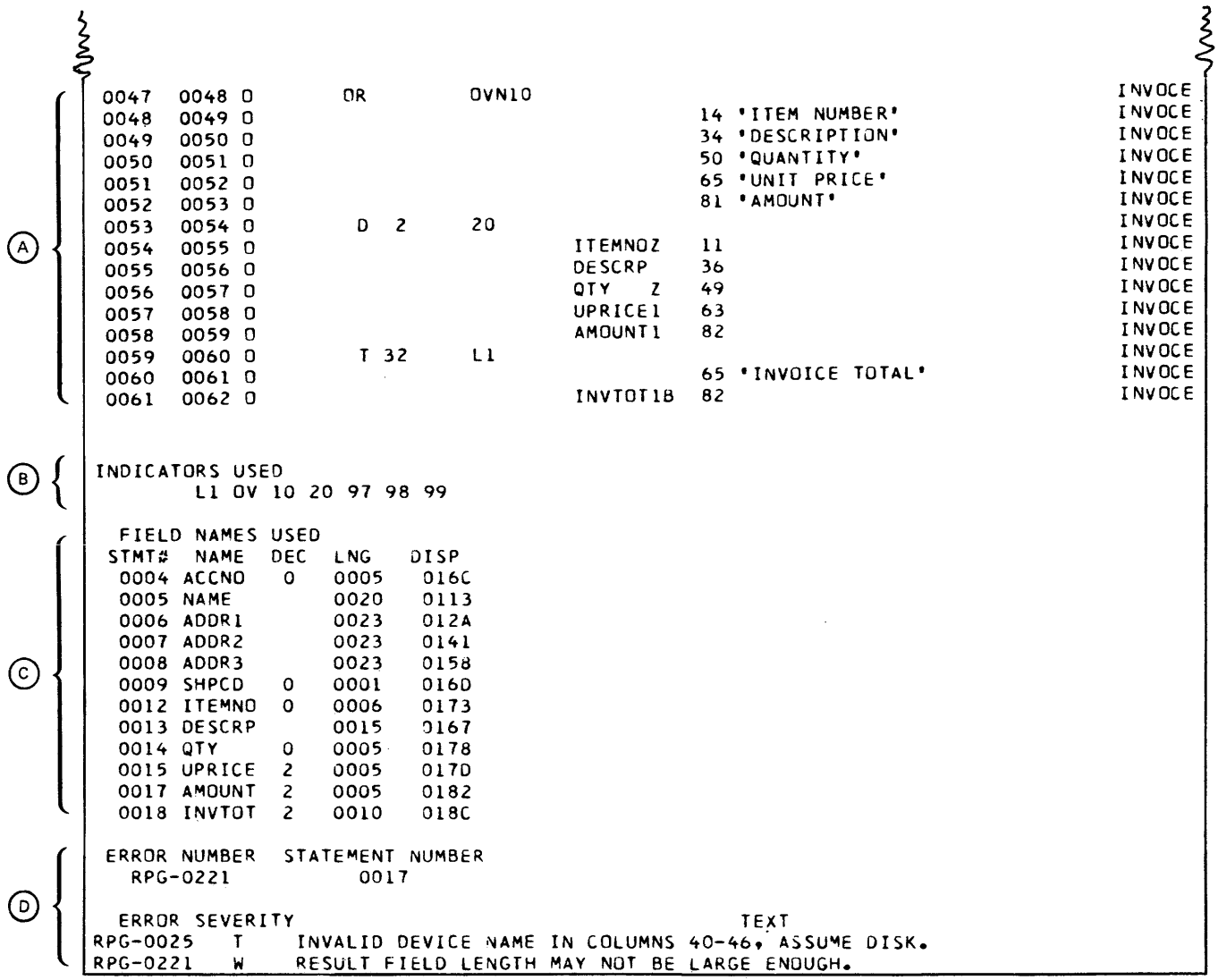
In this example, the programmer did not specify information to be included on the second heading line. On the lines following line 20, he entered the missing specifications. Notice that positions 3 through 5 tell where the specifications belong. The line numbers 051 and 052 indicate that the specifications belong between lines 05 and 06.

If your specifications are being keypunched, the out-of-order cards must be inserted in the appropriate place. If the source program is being entered directly into the system through a keyboard, the missing specifications will have to be inserted in the appropriate place when the specifications are keyed.

COMPILE THE SOURCE PROGRAM

When you think your source program is free of errors, it can be keyed according to the source entry method for your system. You can then compile your source program. The important part of compilation is, of course, translating the source program into machine language. But in addition to this, the compiler also produces a program listing similar to the following:

```
      0001 H     014                                                        INVOCE



 0001  0002 FNAMADD  IP     128 128            DSIK                         INVOCE
(D)→RG 025
 0002  0003 FINVOICE O      120 120     OV     PRINTER                      INVOCE



 0003  0004 INAMADD  011 10    1 CN                                         INVOCE
 0004  0005 I                                    2   60ACCNO L1             INVOCE
 0005  0006 I                                    7   26 NAME                INVOCE
 0006  0007 I                                   27   49 ADDR1               INVOCE
 0007  0008 I                                   50   72 ADDR2               INVOCE
 0008  0009 I                                   73   95 ADDR3               INVOCE
 0009  0010 I                                   96   960SHPCD               INVOCE
 0010  0011 I         02N 20    1 CT                                         INVOCE
 0011  0012 I                                    2   60ACCNO L1             INVOCE
 0012  0013 I                                    9   140ITEMNO              INVOCE
 0013  0014 I                                   15   29 DESCRP              INVOCE
 0014  0015 I                                   30   340QTY                 INVOCE
 0015  0016 I                                   35   392UPRICE              INVOCE



 0016  0017 C   10      SHPCD     COMP 2                    979899IS CODE 1,2,3,?INVOCE
 0017  0018 C   20      QTY       MULT UPRICE    AMOUNT  52       FIND ITEM TOTALINVOCE
 0018  0019 C   20      AMOUNT    ADD  INVTOT    INVTOT 102       FIND INV TOTAL INVOCE



 0019  0020 OINVOICE H  304   10                                             INVOCE
 0020  0021 O        OR       OVN10                                          INVOCE
(A)0021  0022 O                                 45 'INVOICE'                 INVOCE
 0022  0023 O         H  3    10                                             INVOCE
 0023  0024 O        OR       OVN10                                          INVOCE
 0024  0025 O                                 17 'ACCOUNT NUMBER'           INVOCE
 0025  0026 O                          ACCNO  23                             INVOCE
 0026  0027 O         H  2    10                                             INVOCE
 0027  0028 O        OR       OVN10                                          INVOCE
 0028  0029 O                                  7 'NAME'                     INVOCE
 0029  0030 O                          NAME   39                             INVOCE
 0030  0031 O         H  1    10                                             INVOCE
 0031  0032 O        OR       OVN10                                          INVOCE
 0032  0033 O                                 10 'ADDRESS'                  INVOCE
 0033  0034 O                          ADDR1  42                             INVOCE
 0034  0035 O         H  1    10                                             INVOCE
 0035  0036 O        OR       OVN10                                          INVOCE
 0036  0037 O                          ADDR2  42                             INVOCE
 0037  0038 O         H  2    10                                             INVOCE



 0038  0039 O        OR       OVN10                                          INVOCE
 0039  0040 O                          ADDR3  42                             INVOCE
 0040  0041 O         H 32    10                                             INVOCE
 0041  0042 O        OR       OVN10                                          INVOCE
 0042  0043 O                                 24 'SHIPPING INSTRUCTIONS'    INVOCE
 0043  0044 O                      97         33 'BY AIR'                   INVOCE
 0044  0045 O                      98         33 'BY TRUCK'                 INVOCE
 0045  0046 O                      99         33 'BY RAIL'                  INVOCE
 0046  0047 O         H  2    10                                             INVOCE
```



```
0047  0048 O        OR          OVN10                                    INVOCE
0048  0049 O                                          14 'ITEM NUMBER'   INVOCE
0049  0050 O                                          34 'DESCRIPTION'   INVOCE
0050  0051 O                                          50 'QUANTITY'      INVOCE
0051  0052 O                                          65 'UNIT PRICE'    INVOCE
0052  0053 O                                          81 'AMOUNT'        INVOCE
0053  0054 O         D  2      20                                        INVOCE
0054  0055 O                             ITEMNOZ    11                  INVOCE
0055  0056 O                             DESCRP     36                  INVOCE
0056  0057 O                             QTY    Z   49                  INVOCE
0057  0058 O                             UPRICE1    63                  INVOCE
0058  0059 O                             AMOUNT1    82                  INVOCE
0059  0060 O         T 32      L1                                        INVOCE
0060  0061 O                                          65 'INVOICE TOTAL' INVOCE
0061  0062 O                             INVTOT1B   82                  INVOCE

INDICATORS USED
        L1 OV 10 20 97 98 99

  FIELD NAMES USED
 STMT#  NAME   DEC  LNG   DISP
  0004 ACCNO   0   0005   016C
  0005 NAME        0020   0113
  0006 ADDR1       0023   012A
  0007 ADDR2       0023   0141
  0008 ADDR3       0023   0158
  0009 SHPCD   0   0001   016D
  0012 ITEMNO  0   0006   0173
  0013 DESCRP      0015   0167
  0014 QTY     0   0005   0178
  0015 UPRICE  2   0005   017D
  0017 AMOUNT  2   0005   0182
  0018 INVTOT  2   0010   018C

 ERROR NUMBER   STATEMENT NUMBER
   RPG-0221           0017

  ERROR SEVERITY                                        TEXT
RPG-0025    T     INVALID DEVICE NAME IN COLUMNS 40-46, ASSUME DISK.
RPG-0221    W     RESULT FIELD LENGTH MAY NOT BE LARGE ENOUGH.
```

The most important parts of the program listing[1] are:

(A) A printout of source specifications including comment lines. Notice the number at the left of each line. This is the sequence number the compiler assigns to the specification.

(B) A list of all indicators used in the program.

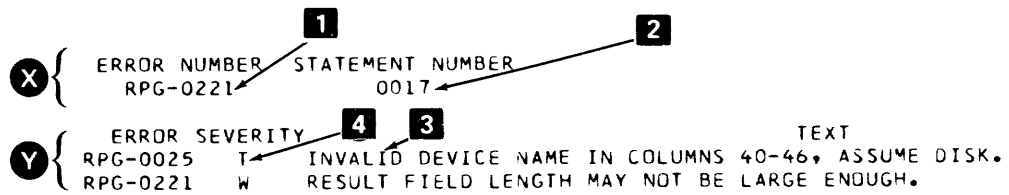
(C) A list of all fields used in the program. Included in the list is the storage location assigned to each field and a description of each field as indicated in your specifications. Some compilers also provide this information for constants used in the program.

(D) Diagnostic messages indicating the types of errors made and the statement in which they occur.

[1] The sample listing shown is the program listing for the invoice job (INVOCE). This program listing was produced by the IBM System/34 RPG II compiler (program 5726-RG1). Other compiler listings provide similar information. See the RPG reference manual for your system for a sample program listing for your system.

If the compiler finds any errors in your source specifications, it prints diagnostic messages telling you what errors were made. Various types of messages are printed: warning, terminal, or informative. A warning message is an indication that something may be wrong. If a warning message is printed, check the questioned specification. If the specification is all right for your program, you may not need to make any changes. If you get a terminal message, however, something is wrong with your coding. You must fix the specification and recompile the program before the compiler will actually translate your specifications.

The following example shows diagnostics from the IBM System/34 RPG II compiler listing. (See the RPG reference manual for your system for a sample program listing for your system.) The diagnostic message section of the program listing contains two basic parts: a list of messages (X) and an explanation of each message (Y).



Each error message in the list is identified by a message number 1. Next to the message number is either a statement number identifying the specification in which the error appears, or a field name or constant associated with the error 2. Following the list of messages is an explanation of each error 3 and an indication of the severity of each error (W = warning; T = terminal) 4.

The sample shown above shows diagnostic messages printed for the invoice program. Note that message 221 is a warning. A warning is an indication that something may be wrong. Therefore, you must check the specification noted. If you find that the specification is correct, you may not need to change it.

Checking the message in the listing, you would find that the warning points to the AMOUNT field in statement 0017:

```
0017  0018 C   20      QTY       MULT UPRICE    AMOUNT  52       FIND ITEM TOTALINVOCE
```

The AMOUNT field is specified as five characters with two decimal positions. In checking your specifications, you find that the amount should have been seven digits with two decimals, and that it was incorrectly keyed in the source data.

Message 0025 is a terminal error. This error number appears below the statement at which the error occurred. Checking the listing, you find that the error number appears at statement 1, where the device entry DISK was miskeyed as DSIK. You must correct this error. (Check the RPG reference manual for your system for the correct entries for the Device positions.)

TEST THE PROGRAM

It is good practice to test your program before using it for an actual job. To do this, make up test data representing all possible situations that could arise during an actual job. Run your program using that data to see if your program will really handle the situations you think it does. If you get the wrong results when testing, you know your program is not doing what you thought it would. You can usually find your errors by using actual input data and doing the operations specified yourself, step by step, in the order the system would do them. When doing this, you have to follow closely your specifications and the program cycle operations taken by your program. After you test your program and the results show it can handle all situations, your job is complete.

Glossary

address: A number identifying a location in storage.

alphabetic: In general usage, any of the letters A through Z. In RPG programming, any of the letters A through Z and special characters @, #, and $.

AND relationship: The specifying of conditioning indicators so that the operation is performed only when all conditions are met.

arithmetic operation: An operation such as addition, subtraction, multiplication, and division performed in the processing unit.

ascending sequence: The arrangement of records in a file from low to high, based on the contents of a specified field in each record.

blank after: An output specification that changes the contents of a field so that it contains only zeros or blanks after that field has been written to the output record.

Calculation Specifications form: An RPG coding form that specifies the type and order of calculations to be performed on the input data.

card: In data processing, a card containing combinations of holes representing data to a system.

card file: A group of related punched-card records.

card layout form: A chart for planning the design and format of cards.

card punch: A device that records information on a card in the form of combinations of holes representing characters.

card reader: A device that electronically senses information on punched cards and transfers that information to the processing unit.

character: Any individual data item that can be represented in printed form; that is, a letter, a digit, or a special character.

character constant: A constant used to represent any of the characters in the data character set.

character field: A field that contains any of the characters in the data character set.

coding: Making entries on RPG specification forms.

comments: Words or statements in a program that serve as documentation rather than as instructions to the compiler.

compile: To translate a source program (such as RPG specifications) into an object program (machine language program) by using the compiler.

compiler: A program that translates a source program into a machine language program.

conditioning: Using indicators to control when calculations or output operations are done.

constant: The actual data (not the name of a field containing the data) to be used in processing. A constant does not change during execution of a program, but the contents of a field can. For example, COST is a name representing a field containing data that changes, whereas the constant 100 is actual data that does not change.

Control and File Description Specifications form: An RPG coding form that gives, for a particular job, information needed for control of the system and a description of the files used.

control break: A change in the contents of a control field. It indicates that all records from a particular control group have been read and that a new control group is starting.

control field: One or more fields that are compared from record to record to determine when the information in the fields changes. When the information changes, the control level indicator (L1 through L9) assigned to a control field is set on.

control group: A set of records all having the same control field information.

control level indicator: An indicator (L1 through L9) used to specify certain fields as control fields and to condition which calculation or output operations to perform at detail or total time.

control unit: An area inside the processing unit that determines from instructions what has to be done. It directs other units or devices to perform the required functions.

data: A collection of facts, numbers, letters, and symbols that can be processed or produced by a system.

data character set: All of the 256 EBCDIC characters.

descending sequence: The arrangement of records in a file from high to low, based on the contents of a specific field in each record.

detail record: An output record produced during the detail output operation of the RPG program cycle.

detail time: An operation in the RPG program cycle in which calculation and output operations are performed for each record read.

diagnostic message: An output message that identifies RPG specification errors and their severity.

digit: One of the characters 0 through 9.

disk: A flat, circular plate with a magnetic surface on which data can be stored.

disk drive: A device that reads data from or writes data on a disk.

disk file: A group of related records stored on disk.

diskette: A thin, flexible magnetic disk permanently enclosed in a semirigid protective jacket.

documentation: A written explanation of a program, its use, function, and operations.

edit: To punctuate a numeric field by suppressing zeros and inserting commas, decimal points, dollar signs, or other constant information.

edit code: A number or letter indicating that editing of a numeric field should be done according to a predefined pattern.

eighty-column card: A punch card with 80 vertical columns representing 80 characters.

end of file: The end of records in a file.

error message: See *diagnostic message.*

execute: To process input data files according to machine language instructions to produce the desired output.

factor: In RPG programming, a field name or constant used in a calculation operation.

field: One or more adjacent record positions that contain a particular item of information.

field indicator: An indicator used to show whether a given field in an input record is plus, minus, zero, or blank.

field length: The number of positions allowed for a given field, determined by the maximum length of information that will be entered in the field.

field name: In RPG programming, a combination of no more than six alphabetic or numeric characters (the first of which must be alphabetic) that identifies a field.

file: An organized collection of related records treated as a unit.

file name: In RPG programming, a combination of no more than eight alphabetic or numeric characters (the first of which must be alphabetic) that identifies a file.

first page indicator: An indicator used to specify which lines (such as headings) should be printed on the first page only.

half adjust: A method of rounding off a number by adjusting the last digit to be kept. When the number to the right of the last digit to be retained is 5 or greater, 1 is added to the last retained digit. For example, 2.475 half adjusted to two decimal places becomes 2.48, but 2.474 becomes 2.47.

heading: A constant, usually printed at the top of a page, identifying the information or report on that page.

indicator: (1) A 2-character entry on an RPG specification form used to test a field or record or to tell when certain calculation or output operations are to be performed. (2) An internal switch used by the object program to remember when a certain event occurs and what to do when the event occurs.

input: Data that is to be operated on (processed) by the system.

input file: A set of records a program uses as a source of data.

Input Specifications form: A coding form used to identify the different types of records in each input file and to describe the fields in each record.

instruction: A statement that specifies an operation to be performed by the system and the locations in storage of all data involved in that operation.

keyboard: A device used to enter data into a system.

keypunch: A device, similar to a typewriter, used for punching information into cards.

last record indicator: An indicator that specifies when the last data record has been processed.

machine language: A language that can be interpreted and used by a system.

ninety-six-column card: A punch card with 96 vertical columns representing 96 characters. The columns are divided horizontally into thirds: the columns in the upper third are numbered 1 through 32; in the middle third, 33 through 64; and in the lower third, 65 through 96.

numeric: Any combination of the digits 0 through 9.

numeric constant: A constant used to represent a number. The constant can consist of a decimal point, a sign, and the digits 0 through 9.

object program: A set of instructions in machine language. The object program is produced by the compiler from the source program.

operation: A defined action performed on one or more data items, such as adding, multiplying or comparing.

operation code: A word or abbreviation specified on the Calculation Specifications form to identify an operation such as SUB for subtract or ADD for addition.

OR relationship: The specifying of conditioning indicators so that the operation conditioned is performed when either one or both of the conditions are met.

output: Data transferred from storage to an external medium such as printed form, punched cards, or disk.

output file: A set of records that is written, punched, or printed by the system to an external medium.

Output Specifications form: An RPG coding form used to specify the records to be written in each output file and the format of the records.

overflow: The condition that occurs when the last line to be printed on the page has been passed.

overflow indicator: An indicator that specifies when the last line on a page has been printed or passed. It can also be used to specify which lines are to be printed on the next page.

overflow line: The line specified as the last line printed on a page.

overflow page: The new page after an overflow has occurred.

primary file: The main file from which a program first reads records.

printer: An output device that records information on paper in the form of printed characters.

printer spacing chart: A form used to plan the location of data in the printer output file.

processing: Performing operations on data from an input record.

processing unit: The part of a system that controls the system and its attached devices, provides storage area for the programs and data, and performs the operations specified in the program.

program: A set of instructions that (when stored) tells the system which operations to do and how to do them.

program cycle: A series of operations performed by the system for each record read.

program listing: A printout that gives information about the source program, such as source statements, diagnostic messages, indicators used, storage addresses of fields, and constants used.

punched card: See *card*.

record: A group of related fields or data items treated as a unit. A record is the basic unit of data transferred between a file and a program.

record identification code: Characters placed in a record to identify that record type.

record identifying indicator: An indicator that identifies the type of record just read.

record length: The total number of positions in a record.

record types: The classification of records in a file. Records are classified according to a specific field or fields within each record. Records of the same type have the same fields in the same order and identical record identification codes.

result field: The name of a field where the outcome of arithmetic calculations is kept.

resulting indicator: An indicator that can specify whether the result of a calculation is plus, minus, zero, or blank; whether a field is greater than, less than, or equal to another field; or whether an operation was successfully completed.

secondary file: Any file other than the primary file used in multifile processing.

source program: A set of instructions representing a particular job as defined by the programmer. These instructions are written in a programming language, such as RPG.

special character: A character other than a digit, a letter, or @, #, and $. For example, *, +, and % are special characters.

specification forms: Forms on which an RPG program is coded and described. The four specification forms described in this manual are the Control and File Description Specifications form, the Input Specifications form, the Calculation Specifications form, and the Output Specifications form.

storage: An area inside the processing unit where instructions and data are stored.

total operations: Operations performed only after a group of records has been processed.

total time: That part of the RPG progam cycle in which calculation and output operations specified for a group of records are done.

zero suppression: The elimination of leading zeros in a number. For example, 00057, when zero suppressed, becomes ƀƀƀ57 (ƀ represents one blank space).

Index

READER'S COMMENT FORM

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Name _____

Address _____

● **No postage necessary if mailed in the U.S.A.**

GC21-7514-3

GC21-7514-3

Cut Along Line

Fold

Fold

FIRST CLASS
PERMIT NO. 40
ARMONK, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold

Fold

IBM

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

Introduction to RPG II and RPG III (File No. S34-28) Printed in U.S.A. GC21-7514-3

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)